Schema Refinement and Normal Forms: Review
Functional Dependencies (FDs)

- A **functional dependency** \( X \rightarrow Y \) holds over relation \( R \) if for every instance \( r \) of \( R \)
  - \( t_1, t_2 \in r \), \( \pi_X (t_1) = \pi_X (t_2) \) implies \( \pi_Y (t_1) = \pi_Y (t_2) \)
  - given two tuples in \( r \), if the \( X \) values agree, \( Y \) values must also agree
  - In \( R \rightarrow W \) example, all emps with rating 8 have wage 10, etc.
- FD is a statement about **all** allowable relations.
  - Identified based on semantics of application (business logic)
  - Given an instance \( r \) of \( R \), we can check if it violates some FD \( f \), but we cannot tell if \( f \) holds over \( R \)!
Checking Functional Dependencies (FDs)

- Check FD $AB \rightarrow C$ on this instance:
  - Look for rows that are same on $A$ and $B$:
  - First two, and for these, $C$ values are the same
  - No more pairs of rows are the same on $A$ and $B,
  - So $AB \rightarrow C$ holds for this instance

![Table](image)

Figure 19.3  An Instance that Satisfies $AB \rightarrow C$
Sample Relation

Hourly_Emps \((ssn, name, lot, rating, wage, hrs\_worked)\)

- Denote relation schema by attribute initial: \(SNLRWH\)

- Constraints (functional dependencies, or FDs for short)
  - \(ssn\) is the key: \(S \rightarrow SNLRWH\)
  - \(rating\) determines \(wage\): \(R \rightarrow W\)
    - E.g., worker with rating \(10\) receives 20$/hr
Anomalies

- Problems due to $R \rightarrow W$ being in Hourly_Emps
  - **Update anomaly:** Change value of $W$ only in a tuple, end up with a dependency violation
  - **Insertion anomaly:** How to insert employee if we don’t know hourly wage for that rating?
  - **Deletion anomaly:** If we delete all employees with rating 5, we lose the information about the wage for rating 5!

```
S  N        L  R  W  H
123-22-3666  Attishoo  48  8  10  40
231-31-5368  Smiley    22  8  10  30
131-24-3650  Smethurst 35  5  7  30
434-26-3751  Guldu     35  5  7  32
612-67-4134  Madayan   35  8  10  40
```

- Note redundancy: multiple $R=8, W=10$ rows, also $R=5, W=7$ rows
Removing Anomalies

**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>

Create 2 smaller tables!

- Updating rating of employee will result in the wage “changing” accordingly
  - Note that there is no physical change of W, just a “pointer change”
- Deleting employee does not affect rating-wages data
- This process is called **decomposition**.
Reasoning About FDs

- Given FD set $F$, we can usually infer additional FDs:
  - $F^+ =$ closure of $F$ is the set of all FDs that are implied by $F$
- 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 $X \rightarrow Z$

- Theorems: Union and decomposition (splitting)
  - $X \rightarrow Y$ and $X \rightarrow Z \implies X \rightarrow YZ$
  - $X \rightarrow YZ \implies X \rightarrow Y$ and $X \rightarrow Z$

- You need to know these, but not how to do multiline proofs using them. They are used in attribute closure...
Attribute Closure: what X determines

- **Attribute closure** of X (denoted X⁺) wrt FD set F:
  - Set of all attributes A such that $X \rightarrow A$ is in $F⁺$
  - Set of all attributes that can be determined starting from attributes in X and using FDs in F

- **Algorithm:**
  $$X⁺ = X$$
  Repeat
  $$Y = X⁺$$ (remember starting $X⁺$ value for this pass)
  Search all FDs in F with LHS completely included in $X⁺$
  Add RHS of each such FD to $X⁺$
  Until $Y = X⁺$ (same as at start of this pass)
Attribute Closure

\[ X^+ = X \]

Repeat

\[ Y = X^+ \] (remember starting \( X^+ \) value)

Find FDs in \( F \) with LHSs completely included in \( X^+ \)

Add RHS of each such FD to \( X^+ \)

Until \( Y = X^+ \) (same as at start of this pass)

- \( C \) is the key: \( C \rightarrow CSJDPQV \) (i.e., \( C \rightarrow R \))
- Project purchases each part using single contract: \( JP \rightarrow C \)
- Dept purchases at most one part from a supplier: \( SD \rightarrow P \)

Example: Compute \( JPD^+ \)

\( JPD^+ = JPD \), look for FDs with LHS in \( JPD \), find \( JP \rightarrow C \)

\( JPD^+ = JPDC \), look again for FDs with LHS in \( JPDC \), find \( C \rightarrow R \)

\( JPD^+ = CSJDPQV \), all attributes, so done

- with abbreviated notation, after numbering the FDs:
  \( JPD^+ = JPD \), = \( JPDC \) by (2), = \( JPDCR \) by (1), = \( R \)
Verifying if attribute set is a key

Key verification can also be done with attribute closure

To verify if $X$ is a key, two conditions needed:
- $X^+ = R$
- $X$ is minimal

How to test minimality
- Removing an attribute from $X$ results in $X'$ such that $X'^+ <> R$: check if it is a key
Lossless Join Decompositions

- A *decomposition* of R replaces it by two or more relations
  - Each new relation schema contains a subset of the attributes of R
  - Every attribute of R is in some relation, in some cases in two.
  - E.g., SNLRWH decomposed into SNLRH and RW

- Decomposition of R into X and Y is *lossless-join* if:
  - $\pi_X(r) \Join \pi_Y(r) = r$
  - If not lossless, then join has extra rows

- *It is essential that all decompositions used to deal with redundancy be lossless!*
The decomposition of R into X and Y is lossless-join wrt F if and only if the closure of F contains:
- $X \cap Y \rightarrow X$, or
- $X \cap Y \rightarrow Y$

Note that $X \cap Y$ constitutes the join columns, so the join columns need to be a superkey of X or Y.

In particular, the decomposition of R into UV and $R - V$ is lossless-join if $U \rightarrow V$ holds over R.

Example: SNLRWH, $R \rightarrow W$ decomposed into SNLRH and RW
Dependency Preserving Decomposition

- Consider CSJDPQV, C is key, JP → C and SD → P.
  - Consider decomposition: CSJDQV and SDP
  - Problem: Checking JP → C requires a join!
  - We want J, P, and C all in one table of the decomposition
  - Then we'll be able to say we've "preserved" this dependency

- Dependency preserving decomposition (Intuitive):
  - If R is decomposed into X and Y, and we enforce the FDs that hold on X, Y then all FDs that were given to hold on R must also hold

- We get a dependency preserving decomposition if all the FDs lie in different tables of the decomposition, and in some additional cases that require a lot of machinery (math-like) that we don't have time to study.
  - E.g., SNLRWH decomposed into SNLRH and RW: S→NLRH in first, R→W in second, S→W by transitivity
Boyce-Codd Normal Form (BCNF)

- Relation $R$ with FDs $F$ is in **BCNF** if, for all $X \rightarrow A$ in $F^+$
  - $A \subseteq X$ (called a *trivial* FD), or
  - $X$ is a superkey of $R$.
- The only non-trivial FDs allowed are key constraints
- Every non-key column must depend on the whole key (2NF)
  - Because if non-key $B$ depended on part of the key, say $X'$, we would have FD $X' \rightarrow B$, where $B$ is not in $X'$ and $X'$ is not a superkey, so the table is not BCNF.
- No redundancy (provable by FDs), no update, insertion, or deletion anomalies are possible
- Note that BCNF is the same as 3NF unless there are multiple keys and at least one multicolumn key, so very close to our goal.
Decomposition into BCNF (and 3NF too)

- Consider relation $R$ with FDs $F$. If $X \rightarrow Y$ violates BCNF*, decompose $R$ into $R - Y$ and $XY$.
- Repeated application of this idea will give us a collection of relations that are in BCNF; lossless join decomposition, and guaranteed to terminate.
- e.g., CSJDPQV, key $C$, $JP \rightarrow C$, $SD \rightarrow P$, $J \rightarrow S$
- To deal with $SD \rightarrow P$, decompose into $SDP$, CSJDQV.
- To deal with $J \rightarrow S$, decompose CSJDQV into $JS$ and CJDQV

*or 3NF if doing decomposition to 3NF, defined soon.
In general, there may not be a dependency preserving decomposition into BCNF.

- e.g., $AB \rightarrow C$, $C \rightarrow A$
- Can’t decompose while preserving first FD; not in BCNF
- We will see that ABC is in 3NF.
Third Normal Form (3NF)

- Relation $R$ with FDs $F$ is in 3NF if, for all $X \rightarrow A$ in $F^+$
  - $A \in X$ (called a trivial FD), or
  - $X$ contains a key for $R$, or
  - $A$ is part of some key for $R$ ($A$ here is a single attribute)
- Minimality of a key is crucial in third condition above!
- If $R$ is in BCNF, it is also in 3NF.
- If $R$ is in 3NF, some redundancy is possible
  - compromise used when BCNF not achievable
  - Lossless-join, dependency-preserving decomposition of $R$ into a collection of 3NF relations always possible.
- 2NF: replace "$X$ contains a key" with "$X$ contains a key or is determined by a key" to allow transitive dependencies
Summary of Schema Refinement

- **BCNF**: relation is free of FD redundancies
  - Having only BCNF relations is desirable
  - If relation is not in BCNF, it can be decomposed to BCNF
    - Lossless join property guaranteed
    - But some FD may be lost (i.e., dependency may not be preserved)

- **3NF is a relaxation of BCNF**
  - Guarantees both lossless join and FD preservation, but allows a small amount of redundancy (only if multiple keys and a multicolumn key)

- **Decompositions may lead to performance loss**
  - *performance requirements* must be considered when using decomposition