Relational Database Design Theory (II)

CS348 Spring 2023

Instructor: Sujaya Maiyya

Sections: 002 & 004 only
Outline For Today

1. Application Constraints and Decompositions
2. Functional Dependencies
3. Boyce-Codd Normal Form (BCNF) & BCNF Decomposition Alg.
4. Dependency Preservation and 3\textsuperscript{rd} Normal Form

This lecture
A Parts/Suppliers database example

- Each type of part has a name and an identifying number and may be supplied by zero or more suppliers.
- Each supplier has an identifying number, a name, and a contact location for ordering parts.
- Each supplier may offer the part at a different price.

![Database Diagram]

- Sno
- Sname
- Sno
- Sname
- Supplies
- Price
- Pno
- Pname
- Part
- Supplier
- City
Single table?

Supplied_Items

<table>
<thead>
<tr>
<th>Sno</th>
<th>Sname</th>
<th>City</th>
<th>Pno</th>
<th>Pname</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Magna</td>
<td>Ajax</td>
<td>P1</td>
<td>Bolt</td>
<td>0.50</td>
</tr>
<tr>
<td>S1</td>
<td>Magna</td>
<td>Ajax</td>
<td>P2</td>
<td>Nut</td>
<td>0.25</td>
</tr>
<tr>
<td>S1</td>
<td>Magna</td>
<td>Ajax</td>
<td>P3</td>
<td>Screw</td>
<td>0.30</td>
</tr>
<tr>
<td>S2</td>
<td>Budd</td>
<td>Hull</td>
<td>P3</td>
<td>Screw</td>
<td>0.40</td>
</tr>
</tbody>
</table>
Decomposed tables?

• An instance

<table>
<thead>
<tr>
<th>Suppliers</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sno</td>
<td>Sname</td>
<td>City</td>
</tr>
<tr>
<td>S1</td>
<td>Magna</td>
<td>Ajax</td>
</tr>
<tr>
<td>S2</td>
<td>Budd</td>
<td>Hull</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parts</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Pno</td>
<td>Pname</td>
</tr>
<tr>
<td>P1</td>
<td>Bolt</td>
</tr>
<tr>
<td>P2</td>
<td>Nut</td>
</tr>
<tr>
<td>P3</td>
<td>Screw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supplies</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sno</td>
<td>Pno</td>
<td>Price</td>
</tr>
<tr>
<td>S1</td>
<td>P1</td>
<td>0.50</td>
</tr>
<tr>
<td>S1</td>
<td>P2</td>
<td>0.25</td>
</tr>
<tr>
<td>S1</td>
<td>P3</td>
<td>0.30</td>
</tr>
<tr>
<td>S2</td>
<td>P3</td>
<td>0.40</td>
</tr>
</tbody>
</table>
Schema decomposition

• Let $R$ be a relation schema (= set of attributes).
• The collection \( \{R_1, \ldots, R_n\} \) of relations is a decomposition of $R$ if $R = R_1 \cup \cdots \cup R_n$

• What is a good decomposition?
Is this a good decomposition?

- Example 1

<table>
<thead>
<tr>
<th>Student</th>
<th>Assignment</th>
<th>Group</th>
<th>Mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ann</td>
<td>A1</td>
<td>G1</td>
<td>80</td>
</tr>
<tr>
<td>Ann</td>
<td>A2</td>
<td>G3</td>
<td>60</td>
</tr>
<tr>
<td>Bob</td>
<td>A1</td>
<td>G2</td>
<td>60</td>
</tr>
</tbody>
</table>

But computing the natural join of SGM and AM, we get **extra data** (spurious tuples).

We would therefore **lose information** if we were to replace Marks by SGM and AM.
“Good” Schema Decomposition

• Lossless-join decompositions
  • We should be able to **construct the instance** of the original table from the instances of the tables in the decomposition

A decomposition \( \{R_1, R_2\} \) of \( R \) is **lossless** iff the common attributes of \( R_1 \) and \( R_2 \) form a superkey for either schema,

\[
R_1 \cap R_2 \rightarrow R_1 \text{ or } R_1 \cap R_2 \rightarrow R_2
\]

*If \( X \) is a superkey of \( R \), then \( X \rightarrow R \) (all the attributes)  [last lecture]*
Is this a lossless join decomposition?

• Example 1
  • $R = \{\text{Student, Assignment, Group, Mark}\}$

<table>
<thead>
<tr>
<th>Student</th>
<th>Assignment</th>
<th>Group</th>
<th>Mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ann</td>
<td>A1</td>
<td>G1</td>
<td>80</td>
</tr>
<tr>
<td>Ann</td>
<td>A2</td>
<td>G3</td>
<td>60</td>
</tr>
<tr>
<td>Bob</td>
<td>A1</td>
<td>G2</td>
<td>60</td>
</tr>
</tbody>
</table>

$\mathcal{F}$ includes: $\text{Student, Assignment} \rightarrow \text{Group, Mark}$

• $R_1 = \{\text{Student, Group, Mark}\}, R_2 = \{\text{Assignment, Mark}\}$

$R_1 \cap R_2 = \{\text{Mark}\}$ is not a superkey of either $R_1$ or $R_2$

$\Rightarrow$ This decomposition is lossy
Which one is a better decomposition?

• Example 2: a table for a company database
  • \( R = \{\text{Proj, Dept, Div}\} \)

\[
\mathcal{F} \text{ includes:} \\
\text{FD1: } \text{Proj} \rightarrow \text{Dept} \\
\text{FD2: } \text{Dept} \rightarrow \text{Div} \\
\text{FD3: } \text{Proj} \rightarrow \text{Div}
\]

• Consider 2 decompositions

\[
D_1 = \left\{ R_1\{\text{Proj, Dept}\}, \\
R_2\{\text{Dept, Div}\} \right\} \\
D_2 = \left\{ R_1\{\text{Proj, Dept}\}, \\
R_2\{\text{Proj, Div}\} \right\}
\]

• Both are lossless. (Why?) \( R_1 \cap R_2 \rightarrow R_1 \text{ or } R_2 \)

• However, testing FDs is easier on one of them. (Which?)
Testing FDs

• Example 2: a table for a company database
  • $R = \{Proj, Dept, Div\}$

  $\mathcal{F}$ includes:
  
  FD1: $Proj \rightarrow Dept$  
  FD2: $Dept \rightarrow Div$  
  FD3: $Proj \rightarrow Div$

• Consider 2 decompositions

  $D_1 = \{R_1\{Proj, Dept\}, \quad R_2\{Dept, Div\}\}$

  $D_2 = \{R_1\{Proj, Dept\}, \quad R_2\{Proj, Div\}\}$

  • FD1 (in R1)
  • FD2 (in R2)
  • FD3 (join R1 and R2?)
  • $\rightarrow$ No need, if FD1 and FD2 hold, then FD3 hold
Testing FDs

• Example 2: a table for a company database
  • $R = \{Proj, Dept, Div\}$

  $\mathcal{F}$ includes:
  - FD1: $Proj \rightarrow Dept$
  - FD2: $Dept \rightarrow Div$
  - FD3: $Proj \rightarrow Div$

• Consider 2 decompositions

  $D_1 = \{R_1\{Proj, Dept\}, \ R_2\{Dept, Div\}\}$
  $D_2 = \{R_1\{Proj, Dept\}, \ R_2\{Proj, Div\}\}$

  • FD1 (in R1)
  • FD2 (in R2)
  • FD3 (join R1 and R2?)
  • $\rightarrow$ No need, if FD1 and FD2 hold, then FD3 hold

  • FD1 (in R1)
  • FD3 (in R2)
  • FD2 (join R1 and R2?)
  • $\rightarrow$ Yes. FD1 and FD3 are not sufficient to guarantee FD2
Testing FDs

• Example 2: a table for a company database
  • \( R = \{ \text{Proj, Dept, Div} \} \)

\[ \mathcal{F} \text{ includes:} \]
- FD1: \( \text{Proj} \rightarrow \text{Dept} \)
- FD2: \( \text{Dept} \rightarrow \text{Div} \)
- FD3: \( \text{Proj} \rightarrow \text{Div} \)

• Consider 2 decompositions

\[ D_1 = \{ R_1\{\text{Proj, Dept}\}, R_2\{\text{Dept, Div}\} \} \]
\[ D_2 = \{ R_1\{\text{Proj, Dept}\}, R_2\{\text{Proj, Div}\} \} \]

- FD1 (in R1)
- FD2 (in R2)
- FD3 (join R1 and R2?)
  \[ \rightarrow \] No need, if FD1 and FD2 hold, then FD3 hold

\[ \text{(i) Equivalent to } \mathcal{F} \]
\[ \text{(ii) Not interrelational} \]

- FD1 (in R1)
- FD2 (join R1 and R2?)
  \[ \rightarrow \] Yes. FD1 and FD3 are not sufficient to guarantee FD2
“Good” Schema Decomposition

• Lossless-join decompositions
• Dependency-preserving decompositions

Given a schema $R$ and a set of FDs $\mathcal{F}$, decomposition of $R$ is dependency preserving if there is an equivalent set of FDs $\mathcal{F}'$, none of which is interrelational in the decomposition.

• Next, how to obtain such decompositions?
  • BCNF $\rightarrow$ guaranteed to be a lossless join decomposition!

Boyce-Codd Normal Form (BCNF)

- A relation $R$ is in BCNF iff whenever $(X \rightarrow Y) \in \mathcal{F}^+$ and $XY \subseteq R$, then either
  - $(X \rightarrow Y)$ is trivial (i.e., $Y \subseteq X$), or
  - $X$ is a super key of $R$ (i.e., $X \rightarrow R$)
    - That is, all non-trivial FDs follow from “key → other attributes”

- Example: $R = \{\text{Sno, Sname, City, Pno, Pname, Price}\}$

  $\mathcal{F}$ includes:
  - **FD1: Sno → Sname, City**
  - **FD2: Pno → Pname**
  - **FD3: Sno, Pno → Price**

  - The schema is not in BCNF because, for example, Sno determines Sname, City, is non-trivial but is not a superkey of $R$
BCNF decomposition algorithm

• Find a BCNF violation
  • That is, a non-trivial FD $X \rightarrow Y$ in $\mathcal{F}^+$ of $R$ where $X$ is not a super key of $R$
    • Example: $R = \{\text{Sno}, \text{Sname}, \text{City}, \text{Pno}, \text{Pname}, \text{Price}\}$

$\mathcal{F}$ includes:

  - FD1: $\text{Sno} \rightarrow \text{Sname, City}$
  - FD2: $\text{Pno} \rightarrow \text{Pname}$
  - FD3: $\text{Sno, Pno} \rightarrow \text{Price}$

• Decompose $R$ into $R_1$ and $R_2$, where
  • $R_1$ has attributes $X \cup Y$;
  • $R_2$ has attributes $X \cup Z$, where $Z$ contains all attributes of $R$ that are in neither $X$ nor $Y$

$R = \{\text{Sno, Sname, City, Pno, Pname, Price}\}$

BCNF violation: $\text{Sno} \rightarrow \text{Sname, City}$

• Repeat (till all are in BCNF)

  R2{\text{Sno, Pno, Pname, Price}}  R1{\text{Sno, Sname, City}}
**BCNF decomposition example**

- \( R = \{\text{Sno, Sname, City, Pno, Pname, Price}\} \)

\( \mathcal{F} \text{ includes:} \)
- \( \text{FD1: Sno } \rightarrow \text{ Sname, City} \)
- \( \text{FD2: Pno } \rightarrow \text{ Pname} \)
- \( \text{FD3: Sno, Pno } \rightarrow \text{ Price} \)

\( \{\text{Sno, Sname, City, Pno, Pname, Price}\} \)

BCNF violation: \( \text{Sno } \rightarrow \text{ Sname, City} \)

R2\{Sno, Pno, Pname, Price\}   R1\{Sno, Sname, City\}

- \( \text{Pno } \rightarrow \text{ Pname} \)
- \( \text{Sno, Pno } \rightarrow \text{ Price} \)

BCNF violation: \( \text{Pno } \rightarrow \text{ Pname} \)

R2b\{Sno, Pno, Price\}   R2a\{Pno, Pname\}

BCNF: \( \text{Sno, Pno } \rightarrow \text{ Price} \)

BCNF: \( \text{Pno } \rightarrow \text{ Pname} \)

\( \{\text{SNo}\}^+ = \{\text{Sno, Sname, City}\} \rightarrow \text{ a superkey of } R1 \)
BCNF helps remove redundancy

<table>
<thead>
<tr>
<th>Sno</th>
<th>Sname</th>
<th>City</th>
<th>Pno</th>
<th>Pname</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Magna</td>
<td>K-W</td>
<td>P1</td>
<td>A</td>
<td>$25</td>
</tr>
<tr>
<td>S1</td>
<td>Magna</td>
<td>K-W</td>
<td>P2</td>
<td>B</td>
<td>$34</td>
</tr>
<tr>
<td>S1</td>
<td>Magna</td>
<td>K-W</td>
<td>P3</td>
<td>A</td>
<td>$20</td>
</tr>
<tr>
<td>S2</td>
<td>Box</td>
<td>London</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BCNF violation: $Sno \rightarrow Sname, City$

<table>
<thead>
<tr>
<th>Sno</th>
<th>Pno</th>
<th>Pname</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>P1</td>
<td>A</td>
<td>$25</td>
</tr>
<tr>
<td>S1</td>
<td>P2</td>
<td>B</td>
<td>$34</td>
</tr>
<tr>
<td>S1</td>
<td>P3</td>
<td>A</td>
<td>$20</td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sno</th>
<th>Sname</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Magna</td>
<td>K-W</td>
</tr>
<tr>
<td>S2</td>
<td>Box</td>
<td>London</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Another example

\[ \mathcal{F} \text{ includes:} \\
\text{uid} \rightarrow \text{uname, twittered} \\
\text{twitterid} \rightarrow \text{uid} \\
\text{uid, gid} \rightarrow \text{fromDate} \]

\text{UserJoinsGroup} (\text{uid, uname, twitterid, gid, fromDate})
Another example

UserJointsGroup (uid, uname, twitterid, gid, fromDate)

BCNF violation: \( uid \rightarrow uname, twitterid \)

User (uid, uname, twitterid)

BCNF
\[ \{uid\}^+ = \{uid, uname, twitterid\} \]
\[ \{twitterid\}^+ = \{uid, uname, twitterid\} \]

Member (uid, gid, fromDate)

BCNF
\[ \{uid, gid\}^+ = \{uid, gid, fromDate\} \]

\[ \mathcal{F} \text{ includes:} \]
uid \rightarrow uname, twitterid

\[ twitterid \rightarrow uid \]

uid, gid \rightarrow fromDate
Alt. solution

UserJoinsGroup \((uid, \text{uname}, \text{twitterid}, \text{gid}, \text{fromDate})\)

BCNF violation: \(\text{twitterid} \rightarrow \text{uid}\)

UserId \((\text{twitterid}, \text{uid})\)

BCNF

UserJoinsGroup \((\text{twitterid}, \text{uname}, \text{gid}, \text{fromDate})\)

BCNF violation: \(\text{twitterid} \rightarrow \text{uname}\)

UserName \((\text{twitterid}, \text{uname})\)

BCNF

Member \((\text{twitterid}, \text{gid}, \text{fromDate})\)

BCNF

\(\mathcal{F}\) includes:
- \(uid \rightarrow \text{uname}, \text{twitterid}\)
- \(\text{twitterid} \rightarrow \text{uid}\)
- \(\text{uid}, \text{gid} \rightarrow \text{fromDate}\)

But we need to check all the FDs in \(\mathcal{F}^+\)!!

No FDs in \(\mathcal{F}\) violate BCNF here! (as \(\text{uid}\) is missing in this relation)
“Good” Schema Decomposition

• Lossless-join decompositions
• Dependency-preserving decompositions
• BCNF $\rightarrow$ guaranteed to be a lossless join decomposition!
  • Depend on the sequence of FDs for decomposition
  • Not necessarily dependency preserving

Example: consider $R=\{A, B, C\}$

$\mathcal{F}$ includes: FD1: $AB \rightarrow C$   FD2: $C \rightarrow B$

BCNF violation: $C \rightarrow B$

$\{A, C\}$   $\{C, B\}$

$AB \rightarrow C$ is interrelational and cannot be tested directly
“Good” Schema Decomposition

• Lossless-join decompositions
• Dependency-preserving decompositions
• BCNF → guaranteed to be a lossless join decomposition!
  • Depend on the sequence of FDs for decomposition
  • Not necessarily dependency preserving
• 3NF → both lossless join and dependency preserving
Third normal form (3NF)

• A relation $R$ is in 3NF iff whenever $(X \rightarrow Y) \in \mathcal{F}^+$ and $XY \subseteq R$, then either
  • $(X \rightarrow Y)$ is trivial (i.e., $Y \subseteq X$), or
  • $X$ is a super key of $R$ (i.e., $X \rightarrow R$) or
  • Each attribute in $Y - X$ is contained in a candidate key of $R$

• Example: consider $R=$\{A, B, C\}
  • Satisfies 3NF, but not BCNF

• 3NF is looser than BCNF $\rightarrow$ Allows more redundancy
How to find a 3NF relation schemas?

• Lossless-join, dependency-preserving decomposition into 3NF relation schemas always exists.

• Step 1: Finding the minimal cover of the FD set $\mathcal{F}$

$$\mathcal{F} \xrightarrow{\mathcal{F}^+} \mathcal{F}' \xrightarrow{\mathcal{F}''^+} \Rightarrow \text{schema}$$

Given a set of FDs $\mathcal{F}$, we say $\mathcal{F}'$ is equivalent to $\mathcal{F}$ if their closures are the same: $\mathcal{F}^+ = \mathcal{F}''^+$.

• Step 2: Decompose based on the minimal cover (i.e., $\mathcal{F}'$ is minimal).
Minimal cover

• A set of FDs $\mathcal{F}$ is minimal if
  1. every right-hand side of a FD in $\mathcal{F}$ is a single attribute

• Example: $R = \{Sno, Sname, City, Pno, Pname, Price, PType\}$

$\mathcal{F}$: FD1: $Sno \rightarrow Sname, City$
FD2: $Pno \rightarrow Pname$
FD3: $Sno, Pno \rightarrow Price$
FD4: $Sno, Pname \rightarrow Price$
FD5: $Pno, Pname \rightarrow Ptype$

Fail condition 1
Minimal cover

- A set of FDs $\mathcal{F}$ is **minimal** if
  1. every right-hand side of a FD in $\mathcal{F}$ is a single attribute
  2. there does not exist $X \rightarrow A$, and $Z$ a proper subset of $X$, such that the set $(\mathcal{F} - \{X \rightarrow A\}) \cup \{Z \rightarrow A\}$ is equivalent to $\mathcal{F}$,
     English: no extraneous (redundant) attributes in the left-hand side of an FD in $\mathcal{F}$

- Example: $R = \{\text{Sno, Sname, City, Pno, Pname, Price, PType}\}$

\[\mathcal{F}:\]
- FD1: $\text{Sno} \rightarrow \text{Sname, City}$
- FD2: $\text{Pno} \rightarrow \text{Pname}$
- FD3: $\text{Sno, Pno} \rightarrow \text{Price}$
- FD4: $\text{Sno, Pname} \rightarrow \text{Price}$
- FD5: $\text{Pno, Pname} \rightarrow \text{Ptype}$

- \(\text{compute} X^+ (\{\text{Pno}\}, \{\text{FD1, FD2, FD3, FD4, FD5}\}) = \{\ldots, \text{Ptype}, \ldots\}\)

[visit Lecture 9 for how to compute closure]
Minimal cover

• A set of FDs $\mathcal{F}$ is minimal if

1. Every right-hand side of a FD in $\mathcal{F}$ is a single attribute
2. There does not exist $X \rightarrow A$ and $Z$ a proper subset of $X$, such that $(\mathcal{F} - \{X \rightarrow A\}) \cup \{Z \rightarrow A\}$ is equivalent to $\mathcal{F}$, English: no extraneous (redundant) attributes in the left-hand side of a FD in $\mathcal{F}$
3. There does not exist $X \rightarrow A$ in $\mathcal{F}$, such that $\mathcal{F} - \{X \rightarrow A\}$ equivalent to $\mathcal{F}$

Example: $R = \{\text{Sno}, \text{Sname}, \text{City}, \text{Pno}, \text{Pname}, \text{Price}, \text{PType}\}$

$\mathcal{F}$: FD1: $\text{Sno} \rightarrow \text{Sname}, \text{City}$
FD2: $\text{Pno} \rightarrow \text{Pname}$
FD3: $\text{Sno}, \text{Pno} \rightarrow \text{Price}$
FD4: $\text{Sno}, \text{Pname} \rightarrow \text{Price}$
FD5: $\text{Pno}, \text{Pname} \rightarrow \text{Ptype}$

Fail condition 3: FD2+FD4 can give FD3 $(\mathcal{F} - \{\text{FD3}\})$ is equiv. to $\mathcal{F}$

$computeX^+\{\text{Sno, Pno}\}, \{\text{FD1, FD2, FD4, FD5}\}$
$= \{\ldots, \text{Price, } \ldots\}$
Finding minimal cover

• A minimal cover for $\mathcal{F}$ can be computed in 3 steps.
  1. Replace $X \rightarrow YZ$ with the pair $X \rightarrow Y$ and $X \rightarrow Z$
  2. Remove $A$ from the left-hand side of $X \rightarrow B$ in $\mathcal{F}$ if $B \in \text{compute}X^+(X - \{A\}, \mathcal{F})$
  3. Remove $X \rightarrow A$ from $\mathcal{F}$ if $A \in \text{compute}X^+(X, \mathcal{F} - \{X \rightarrow A\})$
• Note that each step must be repeated until it no longer succeeds in updating $\mathcal{F}$.

• Example: $R = \{\text{Sno}, \text{Sname}, \text{City}, \text{Pno}, \text{Pname}, \text{Price}, \text{PType} \}$

$\mathcal{F}$: FD1: Sno $\rightarrow$ Sname, City
FD2: Pno $\rightarrow$ Pname
FD3: Sno, Pno $\rightarrow$ Price
FD4: Sno, Pname $\rightarrow$ Price
FD5: Pno, Pname $\rightarrow$ Ptype

Remove FD3

Pno $\rightarrow$ Ptype
Computing 3NF decomposition

Efficient algorithm for computing a 3NF decomposition of $R$ with FDs $\mathcal{F}$:

1. Initialize the decomposition with empty set
2. Find a minimal cover for $\mathcal{F}$, let it be $\mathcal{F}^*$
3. For every $(X \rightarrow Y) \in \mathcal{F}^*$, add a relation $\{XY\}$ to the decomposition
4. If no relation contains a candidate key for $R$, then compute a candidate key $K$ for $R$, and add relation $\{K\}$ to the decomposition.
Example for 3NF decomposition

\[ R = \{ \text{Sno, Sname, City, Pno, Pname, Price} \} \]

\[ \mathcal{F}: \text{FD1: Sno} \rightarrow \text{Sname, City} \]
\[ \text{FD2: Pno} \rightarrow \text{Pname} \]
\[ \text{FD3: Sno, Pno} \rightarrow \text{Price} \]
\[ \text{FD4: Sno, Pname} \rightarrow \text{Price} \]

\[ \mathcal{F}^*: \text{FD1a: Sno} \rightarrow \text{Sname} \]
\[ \text{FD1b: Sno} \rightarrow \text{City} \]
\[ \text{FD2: Pno} \rightarrow \text{Pname} \]
\[ \text{FD4: Sno, Pname} \rightarrow \text{Price} \]

• Add relation for candidate key
• Optimization for this example: combine relations R1a and R1b
Summary

• Functional dependencies: provide clues towards elimination of (some) redundancies in a schema.
  • Closure of FDs (rules, e.g. Armstrong’s axioms)
  • Compute attribute closure

• Schema decomposition
  • Lossless join decompositions
  • Dependency preserving decompositions
  • Normal forms based on FDs
    • BCNF $\rightarrow$ lossless join decompositions
    • 3rd NF $\rightarrow$ lossless join and dependency-preserving decompositions with more redundancy