COMP 430
Intro. to Database Systems
Normal Forms

Slides use ideas from Chris Ré.
What makes a **good** design?

During design process:

- Choose attributes & types appropriately.
- Pick entity sets appropriately.
- Associate attributes with entity sets appropriately.

Follow common patterns.

Any **objective** criteria for evaluating our decisions?
Normal forms

- Domain/key normal form
- 6\textsuperscript{th} normal form
- 5\textsuperscript{th} normal form
- 4\textsuperscript{th} normal form
- Boyce-Codd normal form = 3.5\textsuperscript{th} normal form
- 3\textsuperscript{rd} normal form
- 2\textsuperscript{nd} normal form
- 1\textsuperscript{st} normal form

Based on functional dependencies, i.e., what attributes depend upon.

Goal: Prevent DB anomalies.
Intuition

(3NF) “[Every] non-key field must provide a fact about the key, the whole key, and nothing but the key.” – Bill Kent

(BCNF) “Each attribute must represent a fact about the key, the whole key, and nothing but the key.” – Chris Date

Course(crn, dept_code, course_number, title)
Student(student_id, first_name, last_name)
Enrollment(crn, student_id, grade)
Denormalization

We’ll later see that there are also reasons to **not** use 3NF / BCNF.
1st normal form

Table represents a mathematical relation:
- Each record unique.
- Single value per record & attribute.
Anomalies

Update, delete, insert
Update anomalies caused by redundancy

Assume no separate Student or Course tables.

<table>
<thead>
<tr>
<th>student</th>
<th>course</th>
<th>room</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>COMP 130</td>
<td>BRK 101</td>
</tr>
<tr>
<td>Jane</td>
<td>COMP 130</td>
<td>BRK 101</td>
</tr>
<tr>
<td>John</td>
<td>COMP 430</td>
<td>DCH 1070</td>
</tr>
<tr>
<td>Mary</td>
<td>COMP 430</td>
<td>DCH 1070</td>
</tr>
<tr>
<td>Sue</td>
<td>COMP 430</td>
<td>DCH 1070</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Updating one room results in inconsistency = update anomaly.
Delete anomalies caused by poor attribute grouping

<table>
<thead>
<tr>
<th>student</th>
<th>course</th>
<th>room</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

No enrolled students results in no information about course = delete anomaly.

Assume no separate Student or Course tables.
Insert anomalies caused by poor attribute grouping

<table>
<thead>
<tr>
<th>student</th>
<th>course</th>
<th>room</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>COMP 130</td>
<td>BRK 101</td>
</tr>
<tr>
<td>Jane</td>
<td>COMP 130</td>
<td>BRK 101</td>
</tr>
<tr>
<td>John</td>
<td>COMP 430</td>
<td>DCH 1070</td>
</tr>
<tr>
<td>Mary</td>
<td>COMP 430</td>
<td>DCH 1070</td>
</tr>
<tr>
<td>Sue</td>
<td>COMP 430</td>
<td>DCH 1070</td>
</tr>
<tr>
<td>…</td>
<td>COMP 600</td>
<td>DCH 1070</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

Assume no separate Student or Course tables.

Need an enrolled student to reserve a room = insert anomaly.
Functional dependencies
FDs identify relationships between attributes

Process:
1. Start with some relational schema.
2. Identify its functional dependences.
3. Use these to design a better schema.
Roadmap

- Define FDs.
- Define closures to find all FDs.
- Define superkeys to determine what should be key.
- Apply definitions to 3NF, BCNF.
- Glimpse at what’s beyond 3NF, BCNF.

Intuition:

Attribute X (room) depends on Y (course).

→

X (room) should be a non-key attribute in a table with Y (course) as key.

With examples & activities, of course!
FD – definition

Let $S$, $T$ be sets of attributes. Let $R$ be a relation.

$S$ functionally determines $T$ ($S \rightarrow T$) holds on $R$ iff for all valid tuples $t_1$, $t_2$ in $R$, if $t_1[S] = t_2[S]$, then $t_1[T] = t_2[T]$.

S → T is a functional dependency.

I.e., all the tuples that fit the intended meaning of $R$. 
**FD – an illustration**

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>...</th>
<th>$S_m$</th>
<th>$T_1$</th>
<th>...</th>
<th>$T_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>John</td>
<td>Smith</td>
<td>Jr.</td>
<td>dog</td>
<td>blue</td>
<td>17</td>
</tr>
<tr>
<td>$t_2$</td>
<td>John</td>
<td>Smith</td>
<td>Jr.</td>
<td>dog</td>
<td>blue</td>
<td>17</td>
</tr>
</tbody>
</table>

If $t_1, t_2$ agree here... ...they also agree here!
## FD example

<table>
<thead>
<tr>
<th>emp_id</th>
<th>name</th>
<th>phone</th>
<th>position</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0045</td>
<td>Smith</td>
<td>1234</td>
<td>Clerk</td>
</tr>
<tr>
<td>E3542</td>
<td>Mike</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E1111</td>
<td>Smith</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E9999</td>
<td>Mary</td>
<td>1234</td>
<td>Lawyer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>emp_id</th>
<th>name</th>
<th>phone</th>
<th>position</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0045</td>
<td>Smith</td>
<td>1234</td>
<td>Clerk</td>
</tr>
<tr>
<td>E3542</td>
<td>Mike</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E1111</td>
<td>Smith</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E9999</td>
<td>Mary</td>
<td>1234</td>
<td>Lawyer</td>
</tr>
</tbody>
</table>

\[
\{\text{position}\} \rightarrow \{\text{phone}\}
\]

\[
\text{Not } \{\text{phone}\} \rightarrow \{\text{position}\}
\]

That a FD holds for one instance doesn’t imply that does for all instances.
Two ways FDs are used

Let $F$ be a set of FDs. Let $R$ be a relation.

• $F$ can specify a set of constraints on $R$’s tuples. Does $F$ hold on $R$?
  • Considered part of $R$’s schema.

• To test whether $R$’s tuples are valid under $F$. Does $R$ satisfy $F$?
Superkey – definition

Let S be a set of attributes. Let R be a relation.

S is a superkey of R iff

S → R,

e.g., for all valid tuples t₁, t₂ in R, if t₁[S]=t₂[S], then t₁[R]=t₂[R].

Intuition: R’s attributes should be dependent on exactly the primary key.
Activity – Find FDs in this instance

Find at least three FDs which hold on this instance:

• \{\}\rightarrow\{\}
• \{\}\rightarrow\{\}
• \{\}\rightarrow\{\}

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>
Problems finding FDs

• Can’t necessarily show a FD holds on all instances.

• Potentially large number – How to find them all?
  • Start with subproblem – Given a set of FDs, what other FDs must hold?
Logical implication of FDs – example

<table>
<thead>
<tr>
<th>name</th>
<th>color</th>
<th>category</th>
<th>dept</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Green</td>
<td>Gadget</td>
<td>Toys</td>
<td>49</td>
</tr>
<tr>
<td>Widget</td>
<td>Black</td>
<td>Gadget</td>
<td>Toys</td>
<td>59</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Green</td>
<td>Whatsit</td>
<td>Garden</td>
<td>99</td>
</tr>
</tbody>
</table>

Given:

\{\text{name}\} \rightarrow \{\text{color}\}
\{\text{category}\} \rightarrow \{\text{dept}\}
\{\text{color, category}\} \rightarrow \{\text{price}\}

Logically implies:

\{\text{name, category}\} \rightarrow \{\text{price}\}
Logical implication rules

Armstrong’s Axioms

• Reflexivity: If \( B \subseteq A \), then \( A \rightarrow B \).
• Augmentation: If \( A \rightarrow B \), then \( AC \rightarrow BC \).
• Transitivity: If \( A \rightarrow B \) and \( B \rightarrow C \), then \( A \rightarrow C \).

Derivable rules:

• Union: If \( A \rightarrow B \) and \( A \rightarrow C \), then \( A \rightarrow BC \).
• Decomposition: If \( A \rightarrow BC \), then \( A \rightarrow B \) and \( A \rightarrow C \).
• Pseudo-transitivity: If \( A \rightarrow B \) and \( BC \rightarrow D \), then \( AC \rightarrow D \).

\[\text{Sound: Only imply correct FDs.}\]
\[\text{Complete: Imply all correct FDs.}\]

Standard notation for union of FD sets.
Activity – using FD implication rules

<table>
<thead>
<tr>
<th>name</th>
<th>color</th>
<th>category</th>
<th>dept</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Green</td>
<td>Gadget</td>
<td>Toys</td>
<td>49</td>
</tr>
<tr>
<td>Widget</td>
<td>Black</td>
<td>Gadget</td>
<td>Toys</td>
<td>59</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Green</td>
<td>Whatsit</td>
<td>Garden</td>
<td>99</td>
</tr>
</tbody>
</table>

**Given:**

{name} → {color}
{category} → {dept}
{color, category} → {price}

**Logical implication:**

{name, category} → {price}
{name, category} → {name}
{name, category} → {color}
{name, category} → {category}
{name, category} → {color, category}

**Rule(s)?**

Closures – all logically implied FDs

Let \( F \) be a set of FDs. Let \( A, B \) be sets of attributes.

The *closure* of \( F \) \((F^+)\) is the set of all FDs logically implied by \( F \).

The *closure* of \( A \) \((A^+)\) is the set of all attributes \( B \) such that \( A \rightarrow B \).
Closure of attribute sets

**Given:**

\{name\} → \{color\}
\{category\} → \{dept\}
\{color, category\} → \{price\}

**Some closures:**

\{name\}^+ \quad = \quad \{name, color\}
\{name, category\}^+ \quad = \quad \{name, category, color, dept, price\}
\{color\}^+ \quad = \quad \{color\}
Algorithm for closure of attribute sets

\[ \text{Closure}(A) = \]
\[ \text{Result} = A \]
\[ \text{Repeat until } A \text{ doesn't change:} \]
\[ \text{For each } \text{FD } B \rightarrow C: \]
\[ \text{if } B \subseteq \text{Result,} \]
\[ \text{then add } C \text{ to Result} \]
\[ \text{Return Result} \]

**Given:**
\{name\} \rightarrow \{color\}
\{category\} \rightarrow \{dept\}
\{color, category\} \rightarrow \{price\}

**Compute:**
\{name, category\}^+
Activity – use closure algorithm

Closure(A) =

Result = A

Repeat until A doesn’t change:

For each FD B → C:

if B ⊆ Result,
then add C to Result

Return Result

Given:

{a, b} → {c}
{a, d} → {e}
{b} → {d}
{a, f} → {b}

Compute:

{a, b}⁺ =
{a, f}⁺ =
Closure of FD set

Let A, B be sets of attributes:

Closure(F) =

Result = ∅

For each subset A:

For each $A \rightarrow B$ provable:

Add $A \rightarrow B$ to Result.

For each $B \subseteq A^+$:

Add $A \rightarrow B$ to Result.

Return Result.
Closure of FD set – example

Given:
\{a, b\} \rightarrow \{c\}
\{a, d\} \rightarrow \{b\}
\{b\} \rightarrow \{d\}

Compute attribute set closures:
\{a\}^+ = \{a\}
\{b\}^+ = \{b, d\}
\{c\}^+ = \{c\}
\{d\}^+ = \{d\}
\{a, b\}^+ = \{a, b, c, d\}
\{a, c\}^+ = \{a, c\}
\{a, d\}^+ = \{a, b, c, d\}
\{b, c\}^+ = \{b, c, d\}
\{b, d\}^+ = \{b, d\}
\{c, d\}^+ = \{c, d\}
\{a, b, c\}^+ = \{a, b, c, d\}
\{a, b, d\}^+ = \{a, b, c, d\}
\{a, c, d\}^+ = \{a, b, c, d\}
\{b, c, d\}^+ = \{b, c, d\}
\{a, b, c, d\}^+ = \{a, b, c, d\}

Compute FDs:
\{a\} \rightarrow \{a\}
\{b\} \rightarrow \{b\}, ... \rightarrow \{d\}, ... \rightarrow \{b, d\}
\{c\} \rightarrow \{c\}
\{d\} \rightarrow \{d\}
\{a, b\} \rightarrow \{a, b, c, d\}
\{a, c\} \rightarrow \{a, c\}
\{a, d\} \rightarrow \{a, b, c, d\}
\{b, c\} \rightarrow \{b, c, d\}
\{b, d\} \rightarrow \{b, d\}
\{c, d\} \rightarrow \{c, d\}
\{a, b, c\} \rightarrow \{a, b, c, d\}
\{a, b, d\} \rightarrow \{a, b, c, d\}
\{a, c, d\} \rightarrow \{a, b, c, d\}
\{b, c, d\} \rightarrow \{b, c, d\}
\{a, b, c, d\} \rightarrow \{a, b, c, d\}

If we take \(A \rightarrow B\) to be shorthand for \(A \rightarrow B',\) where \(B' \subseteq B.\)
## Closure of FD set – example

**Given:**

<table>
<thead>
<tr>
<th>{a, b} → {c}</th>
<th>{a} → {a}</th>
</tr>
</thead>
<tbody>
<tr>
<td>{a, d} → {b}</td>
<td>{b} → {b}, ... → {d}, ... → {b, d}</td>
</tr>
<tr>
<td>{b} → {d}</td>
<td>{c} → {c}</td>
</tr>
</tbody>
</table>

**Compute FDs:**

| \{a, b\} → {a, b, c, d} | \{a, b\} → {c, d} |
| \{a, c\} → {a, c}        | \{a, d\} → {b, c} |
| \{a, d\} → {a, b, c, d}  | \{b, c\} → {d}   |
| \{b, c\} → {b, c, d}     | \{b, d\} → {b, d} |
| \{b, d\} → {b, d}        | \{c, d\} → {c, d} |
| \{c, d\} → {c, d}        | \{a, b, c\} → {a, b, c, d} |
| \{a, b, d\} → {a, b, c, d} | \{a, b, d\} → {c} |
| \{a, c, d\} → {a, b, c, d} | \{a, c, d\} → {b} |
| \{b, c, d\} → {b, c, d}  | \{a, b, c, d\} → {a, b, c, d} |
| \{a, b, c, d\} → {a, b, c, d} | \{a, b, c, d\} → {a, b, c, d} |

**Shorthand version:**

| \{b\} → {d} |

Eliminating *trivial* FDs

\(A \rightarrow B\), where \(B \subseteq A\).

Replacing FDs \(A \rightarrow AB\) with \(A \rightarrow B\).
FD covers & equivalence

F covers G iff G can be inferred from F. I.e., \( G^+ \subseteq F^+ \).

F and G are equivalent iff \( F^+ = G^+ \).

Given:
\[
\begin{align*}
\{a, b\} & \rightarrow \{c\} \\
\{a, d\} & \rightarrow \{b\} \\
\{b\} & \rightarrow \{d\}
\end{align*}
\]

Compute FDs:
\[
\begin{align*}
\{a\} & \rightarrow \{a\} \\
\{b\} & \rightarrow \{b, d\} \\
\{c\} & \rightarrow \{c\} \\
\{d\} & \rightarrow \{d\} \\
\{a, b\} & \rightarrow \{a, b, c, d\} \\
\{a, c\} & \rightarrow \{a, c\} \\
\{a, d\} & \rightarrow \{a, b, c, d\} \\
\{b, c\} & \rightarrow \{b, c, d\} \\
\{b, d\} & \rightarrow \{b, d\} \\
\{c, d\} & \rightarrow \{c, d\} \\
\{a, b, c\} & \rightarrow \{a, b, c, d\} \\
\{a, b, d\} & \rightarrow \{a, b, c, d\} \\
\{a, c, d\} & \rightarrow \{a, b, c, d\} \\
\{b, c, d\} & \rightarrow \{b, c, d\} \\
\{a, b, c, d\} & \rightarrow \{a, b, c, d\}
\end{align*}
\]

Shorthand version:
\[
\begin{align*}
\{b\} & \rightarrow \{d\} \\
\{a, b\} & \rightarrow \{c, d\} \\
\{a, d\} & \rightarrow \{b, c\} \\
\{b, c\} & \rightarrow \{d\} \\
\{a, b, c\} & \rightarrow \{d\} \\
\{a, b, d\} & \rightarrow \{c\} \\
\{a, c, d\} & \rightarrow \{b\}
\end{align*}
\]
Superkeys & keys
Superkeys & keys (reminder)

Let $S$ be a set of attributes. Let $R$ be a relation.

$S$ is a **superkey** of $R$ iff $S \rightarrow R$.

Equivalently, iff $S^+ = R$.

A **key** is a minimal superkey.

We pick a key as *primary key*. 
Superkeys & keys – example

Product(name, price, category, color)

{name, category} → price
{category} → color

What are superkey(s)? Key(s)?
How can you search for them?
Activity

10a-keys.ipynb
Normal forms
FD-based table normalization

While there are “bad” FDs
    Decompose a table with “bad” FDs into sub-tables.
Boyce-Codd normal form (BCNF)

(BCNF) “Each attribute must represent a fact about the key, the whole key, and nothing but the key.” – Chris Date

Ignoring trivial FDs:

- Good: $X \rightarrow R$  
  $X$ is a (super)key

- Bad: $X \rightarrow A$  
  for $A \subseteq R$  
  $X$ is not a (super)key

“Bad” because $X$ isn’t the primary key, but it functionally determines some of the attributes. Thus, there is redundancy.
BCNF – definition

Relation R is in BCNF
iff
whenever $A \rightarrow B$ is a non-trivial FD in R, then A is a superkey for R.

I.e., each FD is trivial or “good”, not “bad”.
BCNF – example

<table>
<thead>
<tr>
<th>name</th>
<th>ssn</th>
<th>phone</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>123-45-6789</td>
<td>713-555-1234</td>
<td>Houston</td>
</tr>
<tr>
<td>Mary</td>
<td>123-45-6789</td>
<td>713-555-6543</td>
<td>Houston</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>512-555-2121</td>
<td>Austin</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>512-555-1234</td>
<td>Austin</td>
</tr>
</tbody>
</table>

Find a “bad” FD. \{ssn\} → \{name, city\}
**BCNF – example fixed**

Decompose table into sub-tables.

<table>
<thead>
<tr>
<th>name</th>
<th>ssn</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>123-45-6789</td>
<td>Houston</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>Austin</td>
</tr>
</tbody>
</table>

Now a “good” FD.

\[ \{\text{ssn}\} \rightarrow \{\text{name, city}\} \]

<table>
<thead>
<tr>
<th>ssn</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-45-6789</td>
<td>713-555-1234</td>
</tr>
<tr>
<td>123-45-6789</td>
<td>713-555-6543</td>
</tr>
<tr>
<td>987-65-4321</td>
<td>512-555-2121</td>
</tr>
<tr>
<td>987-65-4321</td>
<td>512-555-1234</td>
</tr>
</tbody>
</table>
**BCNF decomposition**

\[ \text{BCNF\_decomp}(R) = \]

Find attribute set \( X \) s.t. \( X^+ \neq X \) (not trivial) and \( X^+ \neq R \) (not superkey).
If no such \( X \), then return \( R \).

Let \( D = X^+ - X \). (attributes functionally determined by \( X \))
Let \( N = R - X^+ \). (attributes not functionally determined by \( X \))
Decompose \( R \) into \( R_1(X \cup D) \) and \( R_2(X \cup N) \).
Return \( \text{BCNF\_decomp}(R_1), \text{BCNF\_decomp}(R_2) \).
BCNF decomposition – example

BCNF_decomp(R) =

Find attribute set X s.t. $X^+ \neq X$ and $X^+ \neq R$.
If no such X, then return R.

Let D = $X^+ - X$.
Let N = $R - X^+$.
Decompose R into $R_1(X \cup D)$ and $R_2(X \cup N)$.
Return BCNF_decomp($R_1$), BCNF_decomp($R_2$).
BCNF decomposition – example

BCNF_decomp(R) =

Find attribute set X s.t. $X^+ \neq X$ and $X^+ \neq R$.
If no such X, then return R.

Let $D = X^+ - X$.
Let $N = R - X^+$.
Decompose R into $R_1(X \cup D)$ and $R_2(X \cup N)$.
Return BCNF_decomp($R_1$), BCNF_decomp($R_2$).
BCNF decomposition – example

BCNF_decomp(R) =

Find attribute set X s.t. $X^+ \neq X$ and $X^+ \neq R$.
If no such X, then return R.

Let $D = X^+ - X$.
Let $N = R - X^+$.

Decompose R into $R_1(X \cup D)$ and $R_2(X \cup N)$.
Return BCNF_decomp(R₁), BCNF_decomp(R₂).

R₁₁(c,d)
{a} → {b, c}
{c} → {d}

No such X.
BCNF decomposition – example

BCNF_decomp(R) =

Find attribute set X s.t. $X^+ \neq X$ and $X^+ \neq R$.
If no such X, then return R.

Let $D = X^+ - X$.
Let $N = R - X^+$.

Decompose R into $R_1(X \cup D)$ and $R_2(X \cup N)$.
Return BCNF_decomp($R_1$), BCNF_decomp($R_2$).
BCNF decomposition – example

BCNF_decomp(R) =

Find attribute set X s.t. \( X^+ \neq X \) and \( X^+ \neq R \).
If no such X, then return R.

Let \( D = X^+ - X \).
Let \( N = R - X^+ \).

Decompose R into \( R_1(X \cup D) \) and \( R_2(X \cup N) \).
Return BCNF_decomp(R_1), BCNF_decomp(R_2).
BCNF decomposition + keys – example

\[ R(a,b,c,d,e) \]
\[ \{a\} \rightarrow \{b, c\} \]
\[ \{c\} \rightarrow \{d\} \]

BCNF

\[ R_{11}(c,d) \]
\[ R_{12}(a,b,c) \]
\[ R_2(a,e) \]
\[ \{a\} \rightarrow \{b, c\} \]
\[ \{c\} \rightarrow \{d\} \]

keys

\[ R_{11}(c,d) \]
\[ R_{12}(a,b,c) \]
\[ R_2(a,e) \]
\[ \{a\} \rightarrow \{b, c\} \]
\[ \{c\} \rightarrow \{d\} \]
Activity – BCNF & keys

10b-bcnf.ipynb

<table>
<thead>
<tr>
<th>name</th>
<th>ssn</th>
<th>phone</th>
<th>city</th>
<th>zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>123-45-6789</td>
<td>713-555-1234</td>
<td>Houston</td>
<td>77005</td>
</tr>
<tr>
<td>Mary</td>
<td>123-45-6789</td>
<td>713-555-6543</td>
<td>Houston</td>
<td>77005</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>281-555-2121</td>
<td>Houston</td>
<td>77005</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>281-555-1234</td>
<td>Houston</td>
<td>77005</td>
</tr>
</tbody>
</table>

{city} → {zip}
{ssn} → {name, city}
Decompositions
BCNF decomposition – the good & the bad

The good: Algorithm to detect & remove redundancies.
          Standard practice.

The bad: Sometimes some subtle, undesirable side-effects.
Decompositions & joins

If decomposition is \textit{lossless}, a join restores the original relation.
Decomposition & join – lossless example

<table>
<thead>
<tr>
<th>name</th>
<th>price</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

Decomp.

Join

<table>
<thead>
<tr>
<th>name</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Camera</td>
</tr>
</tbody>
</table>
Decomposition & join – lossy example

<table>
<thead>
<tr>
<th>name</th>
<th>price</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

Loses the association between name and price.

<table>
<thead>
<tr>
<th>price</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Camera</td>
</tr>
</tbody>
</table>

Decomp.

Join
BCNF decompositions is lossless

\[ R(A_1, ..., A_n, B_1, ..., B_m, C_1, ..., C_p) \]

Decomp. \[ \rightarrow \]

Lossless iff
\[ \{A_1, ..., A_n\} \rightarrow \{B_1, ..., B_m\} \]

Join

R_1(A_1, ..., A_n, B_1, ..., B_m)
R_2(A_1, ..., A_n, C_1, ..., C_p)

Don’t need
\[ \{A_1, ..., A_n\} \rightarrow \{C_1, ..., C_p\} \]

Holds by definition of BCNF decomposition algorithm.
BCNF can lose FD information

R(name, company, category)
{category, company} → {name}
{name} → {company}

Keys: {category, company}, {category, name}

“Bad” FD

R₁(name, category)
R₂(name, company)
{category, company} → {name}
{name} → {company}

Can’t enforce “nonlocal” FD.

<table>
<thead>
<tr>
<th>name</th>
<th>company</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>GizmoWorks</td>
<td>Gadget</td>
</tr>
<tr>
<td>GizmoPlus</td>
<td>GizmoWorks</td>
<td>Gadget</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>company</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>GizmoWorks</td>
</tr>
<tr>
<td>GizmoPlus</td>
<td>GizmoWorks</td>
</tr>
</tbody>
</table>
Three solutions

• Accept the BCNF tradeoff between avoiding redundancy/anomalies and preserving FDs.
  
  BCNF is most common choice.

• Take extra steps to enforce these FDs.
  
  E.g., join tables and then check.

• Weaken decomposition so that no such lost FDs.
  
  E.g., 3NF.
3\textsuperscript{rd} normal form (3NF)

**BCNF:**

Relation R is in BCNF iff whenever $A \rightarrow B$ is a non-trivial FD in R, then
- A is a superkey for R.

**3NF:**

Relation R is in 3NF iff whenever $A \rightarrow B$ is a non-trivial FD in R, then either:
- A is a superkey for R, or
- Every element of B is part of a key.
3NF avoids losing FD information

\[ R(\text{name, company, category}) \]
\[ \{\text{category, company}\} \rightarrow \{\text{name}\} \]
\[ \{\text{name}\} \rightarrow \{\text{company}\} \]

BCNF: “Bad”
3NF: “Good” because \text{company} part of a key.

Keys: \{\text{category, company}\},
\{\text{category, name}\}
3NF allows some redundancies/anomalies

<table>
<thead>
<tr>
<th>name</th>
<th>company</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>GizmoWorks</td>
<td>Gadget</td>
</tr>
<tr>
<td>Gizmo</td>
<td>GizmoWorks</td>
<td>Camera</td>
</tr>
<tr>
<td>GizmoPlus</td>
<td>GizmoWorks</td>
<td>Gadget</td>
</tr>
<tr>
<td>GizmoPlus</td>
<td>GizmoWorks</td>
<td>Camera</td>
</tr>
<tr>
<td>NewThing</td>
<td>NULL</td>
<td>Gadget</td>
</tr>
</tbody>
</table>

Redundancy. Repeating product’s company.

Insertion anomaly. Product not yet made by any company.
More about 3NF

• 3NF is still lossless!

• Requires somewhat more complicated decomposition algorithm.
Glimpse beyond BCNF/3NF/FDs

• 4NF – multi-valued dependencies (generalizes FDs)
• 5NF & 6NF – join dependencies
• DKNF – only domain & key constraints
Multi-value dependencies

FD:  \( A \rightarrow B \)  The value in A determines a value for B.
MVD:  \( A \rightarrow\!\!\!\!\!
\) The value in A determines a set of values for B.

<table>
<thead>
<tr>
<th>restaurant</th>
<th>menu_item</th>
<th>delivery_area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Papa John’s</td>
<td>Pizza</td>
<td>Rice Village</td>
</tr>
<tr>
<td>Papa John’s</td>
<td>Pizza</td>
<td>Rice University</td>
</tr>
<tr>
<td>Papa John’s</td>
<td>Pizza</td>
<td>Southampton</td>
</tr>
<tr>
<td>Domino’s</td>
<td>Pizza</td>
<td>Rice Village</td>
</tr>
<tr>
<td>Domino’s</td>
<td>Pizza</td>
<td>Rice University</td>
</tr>
<tr>
<td>Domino’s</td>
<td>Pasta</td>
<td>Rice Village</td>
</tr>
<tr>
<td>Domino’s</td>
<td>Pasta</td>
<td>Rice University</td>
</tr>
</tbody>
</table>

\{restaurant\} \rightarrow \{menu\_item\}
\{restaurant\} \rightarrow \{delivery\_area\}
Normal form summary

• Constraints on data/tables to limit redundancy

• Decomposition strategy/algorithm to meet constraints

• Different normal forms for different trade-offs