DATABASE DESIGN

How do we design a “good” database?

We want to ensure the integrity of the data.

We also want to get good performance.
TODAY'S AGENDA

Normal Forms
NoSQL Denormalization
NORMAL FORMS

Now that we know how to derive more FDs, we can then:
→ Search for “bad” FDs
→ If there are such, then decompose the table into two tables, repeat for the sub-tables.
→ When done, the database schema is normalized.
NORMAL FORMS

A normal form is a characterization of a decomposition in terms of the properties that satisfies when putting the relations back together. → Also called the "universal relation"

Loseless Joins
Dependency Preservation
Redundancy Avoidance
DECOMPOSITION
SUMMARY

Lossless Joins

→ Motivation: Avoid information loss.
→ Goal: No noise introduced when reconstituting universal relation via joins.
→ Test: At each decomposition \( R=(R_1 \cup R_2) \), check whether \((R_1 \cap R_2) \Rightarrow R_1\) or \((R_1 \cap R_2) \Rightarrow R_2\).
DECOMPOSITION SUMMARY

Dependency Preservation

→ Motivation: Efficient FD assertions.
→ Goal: No global integrity constraints that require joins of more than one table with itself.
→ Test: $R = (R_1 \cup \ldots \cup R_n)$ is dependency preserving if closure of FD's covered by each $R_1 = \text{closure of FD's covered by } R = F$. 
DECOMPOSITION SUMMARY

Redundancy Avoidance

→ Motivation: Avoid update, delete anomalies.
→ Goal: Avoid update anomalies, wasted space.
→ Test: For an $X \rightarrow Y$ covered by $R_n$, $X$ should be a super key of $R_n$. 
HISTORY

Ted Codd introduced the concept of normalization and the **first normal form** in 1970.

Codd went on to define the **second normal form** and **third normal form** in 1971.

Codd and Raymond Boyce defined the **Boyce-Codd normal form** in 1974.
NORMAL FORMS

1\textsuperscript{st} Normal Form (1NF) \(\rightarrow\) All Tables are Flat

2\textsuperscript{nd} Normal Form (2NF) \(\rightarrow\) "Good Enough"

3\textsuperscript{rd} Normal Form (3NF) \(\rightarrow\) Most Common

Boyce-Codd Normal Form (BCNF) \(\rightarrow\) Most Common

4\textsuperscript{th} & 5\textsuperscript{th} Normal Forms \(\rightarrow\) See textbook

6\textsuperscript{th} Normal Form \(\rightarrow\) Most (normal) people never need this.
MORE NORMAL FORMS

Domain-Key Normal Form (1981)
Elementary Key Normal Form (1982)
Inclusion Normal Form (1992)
Key-Complete Normal Form (1998)
Inclusion Dependency Normal Form (2000)
THE UNIVERSE OF RELATIONS

All Relations

1NF

2NF

3NF

BCNF

4NF

5NF
## FIRST NORMAL FORM

All types must be atomic.
No repeating groups.

`loans(bname, assets, cname, loanId, amt)`

<table>
<thead>
<tr>
<th>bname</th>
<th>assets</th>
<th>cname</th>
<th>loanId</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pittsburgh</td>
<td>$9M</td>
<td>[Andy, DJ Snake]</td>
<td>L-17</td>
<td>$1000</td>
</tr>
<tr>
<td>Pittsburgh</td>
<td>$9M</td>
<td>Obama</td>
<td>L-23</td>
<td>$2000</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>$2M</td>
<td>Andy</td>
<td>L-93</td>
<td>$500</td>
</tr>
</tbody>
</table>
FIRST NORMAL FORM

All types must be atomic.
No repeating groups.

loans(bname, assets, c1, c2, ..., loanId, amt)

<table>
<thead>
<tr>
<th>bname</th>
<th>assets</th>
<th>c1</th>
<th>c2</th>
<th>c...</th>
<th>loanId</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pittsburgh</td>
<td>$9M</td>
<td>Andy</td>
<td>DJ Snake</td>
<td></td>
<td>L-17</td>
<td>$1000</td>
</tr>
<tr>
<td>Pittsburgh</td>
<td>$9M</td>
<td>Obama</td>
<td></td>
<td></td>
<td>L-23</td>
<td>$2000</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>$2M</td>
<td>Andy</td>
<td></td>
<td></td>
<td>L-93</td>
<td>$500</td>
</tr>
</tbody>
</table>
FIRST NORMAL FORM

All types must be atomic.
No repeating groups.

loans(bname,assets,cname,loanId,amt)

<table>
<thead>
<tr>
<th>bname</th>
<th>assets</th>
<th>cname</th>
<th>loanId</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pittsburgh</td>
<td>$9M</td>
<td>Andy</td>
<td>L-17</td>
<td>$1000</td>
</tr>
<tr>
<td>Pittsburgh</td>
<td>$9M</td>
<td>Obama</td>
<td>L-23</td>
<td>$2000</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>$2M</td>
<td>Andy</td>
<td>L-93</td>
<td>$500</td>
</tr>
<tr>
<td>Pittsburgh</td>
<td>$9M</td>
<td>DJ Snake</td>
<td>L-17</td>
<td>$1000</td>
</tr>
</tbody>
</table>

✓ Valid 1NF
SECOND NORMAL FORM

1NF and non-key attributes fully depend on the candidate key.

```
loans(bname, assets, cname, loanId, amt)
```

<table>
<thead>
<tr>
<th>bname</th>
<th>assets</th>
<th>cname</th>
<th>loanId</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pittsburgh</td>
<td>$9M</td>
<td>Andy</td>
<td>L-17</td>
<td>$1000</td>
</tr>
<tr>
<td>Pittsburgh</td>
<td>$9M</td>
<td>Obama</td>
<td>L-23</td>
<td>$2000</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>$2M</td>
<td>Andy</td>
<td>L-93</td>
<td>$500</td>
</tr>
<tr>
<td>Pittsburgh</td>
<td>$9M</td>
<td>DJ Snake</td>
<td>L-17</td>
<td>$1000</td>
</tr>
</tbody>
</table>

Provided FDs

bname → assets
loanId → amt,bname
SECOND NORMAL FORM

1NF and non-key attributes fully depend on the candidate key.

Provided FDs
\[ \text{bname} \rightarrow \text{assets} \]
\[ \text{loanId} \rightarrow \text{amt, bname} \]

\[ R_1(\text{bname}, \text{assets}, \text{cname}, \text{loanId}) \]

<table>
<thead>
<tr>
<th>bname</th>
<th>assets</th>
<th>cname</th>
<th>loanId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pittsburgh</td>
<td>$9M</td>
<td>Andy</td>
<td>L-17</td>
</tr>
<tr>
<td>Pittsburgh</td>
<td>$9M</td>
<td>Obama</td>
<td>L-23</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>$2M</td>
<td>Andy</td>
<td>L-93</td>
</tr>
<tr>
<td>Pittsburgh</td>
<td>$9M</td>
<td>DJ Snake</td>
<td>L-17</td>
</tr>
</tbody>
</table>

\[ R_2(\text{loanId}, \text{bname}, \text{amt}) \]

<table>
<thead>
<tr>
<th>loanId</th>
<th>bname</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-17</td>
<td>Pittsburgh</td>
<td>$1000</td>
</tr>
<tr>
<td>L-23</td>
<td>Pittsburgh</td>
<td>$2000</td>
</tr>
<tr>
<td>L-93</td>
<td>Los Angeles</td>
<td>$500</td>
</tr>
</tbody>
</table>
1NF and non-key attributes fully depend on the candidate key.

**SECOND NORMAL FORM**

Provided FDs:
- \( bname \rightarrow assets \)
- \( loanId \rightarrow amt, bname \)

\[ R_1(bname, assets, cname, loanId) \]

<table>
<thead>
<tr>
<th>bname</th>
<th>assets</th>
<th>cname</th>
<th>loanId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pittsburgh</td>
<td>$9M</td>
<td>Andy</td>
<td>L-17</td>
</tr>
<tr>
<td>Pittsburgh</td>
<td>$9M</td>
<td>Obama</td>
<td>L-23</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>$2M</td>
<td>Andy</td>
<td>L-93</td>
</tr>
<tr>
<td>Pittsburgh</td>
<td>$9M</td>
<td>DJ Snake</td>
<td>L-17</td>
</tr>
</tbody>
</table>

\[ R_2(loanId, bname, amt) \]

<table>
<thead>
<tr>
<th>loanId</th>
<th>bname</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-17</td>
<td>Pittsburgh</td>
<td>$1000</td>
</tr>
<tr>
<td>L-23</td>
<td>Pittsburgh</td>
<td>$2000</td>
</tr>
<tr>
<td>L-93</td>
<td>Los Angeles</td>
<td>$500</td>
</tr>
</tbody>
</table>
SECOND NORMAL FORM

1NF and non-key attributes fully depend on the candidate key.

$R_1(bname, assets)$

- Pittsburgh: $9M$
- Los Angeles: $2M$

$R_2(loanId, bname, amt)$

- L-17: Pittsburgh, $1000$
- L-23: Pittsburgh, $2000$
- L-93: Los Angeles, $500$

$R_3(bname, cname, loanId)$

- Pittsburgh, Andy, L-17
- Pittsburgh, Obama, L-23
- Los Angeles, Andy, L-93
- Pittsburgh, DJ Snake, L-17

✓ Valid 2NF

Provided FDs:
- $bname \rightarrow assets$
- $loanId \rightarrow amt, bname$
BOYCE-CODD NORMAL FORM

BCNF guarantees no redundancies and no lossless joins (but not DP).

A relation $R$ with FD set $F$ is in BCNF if for all non-trivial $X \rightarrow Y$ in $F+$:

$\rightarrow X \rightarrow R$ (i.e., $X$ is a super key)
BOYCE-CODD NORMAL FORM (EX.1)

Is \( R \) in BCNF?

Consider the non-trivial dependencies in \( F^+ \):

- \( A \rightarrow B \), \( A \rightarrow R \) (\( A \) is a super key)
- \( A \rightarrow C \), \( A \rightarrow R \) (\( A \) is a super key)
- \( B \rightarrow C \), \( B \nrightarrow A \) (\( B \) is not a super key)

\[ R(A, B, C) \]
\[ F = \{ A \rightarrow B, \ B \rightarrow C \} \]

Not Valid BCNF
BOYCE-CODD NORMAL FORM (EX.2)

Is $R_1$ and $R_2$ in BCNF?

Step #1 – Test $R_1$
$A \rightarrow B$, $A \rightarrow R_1$ ($A$ is a super key)

Step #2 – Test $R_2$
$B \rightarrow C$, $B \rightarrow R_2$ ($B$ is a super key)

$F = \{A \rightarrow B, \ B \rightarrow C\}$

$R_1(A, B)$ $R_2(B, C)$

✓ Valid BCNF
BOYCE-CODD NORMAL FORM

Given a schema $R$ and a set of FDs $F$, we can always decompose $R$ into \{R_1, \ldots, R_n\} such that
\[
\rightarrow \{R_1, \ldots, R_n\} \text{ are in BCNF}
\rightarrow \text{The decompositions are lossless.}
\]

But some BCNF decompositions might lose dependencies.
BCNF DECOMPOSITION ALGORITHM

Given a relation $R$ and a FD set $F$:
Step #1 – Compute $F^+$
Step #2 – $\text{Result} \leftarrow \{R\}$
Step #3 – While $R_i \in \text{Result}$ not in BCNF, do:
  $\rightarrow$ (a) Choose $(X \rightarrow Y) \in F^+$ such that $(X \rightarrow Y)$ is covered by $R_i$ and $X \Rightarrow R_i$
  $\rightarrow$ (b) Decompose $R_i$ on $(X \rightarrow Y)$:

\[
\begin{align*}
R_{i,1} & \leftarrow X \cup Y & (R_{i,1} \text{ includes } Y) \\
R_{i,2} & \leftarrow R_i - Y & (R_{i,2} \text{ does not include } Y)
\end{align*}
\]

$\text{Result} \leftarrow (\text{Result} - \{R_i\}) \cup \{R_{i,1}, R_{i,2}\}$
BOYCE-CODD NORMAL FORM (EX.3)

Step #1 – Compute Closure

$\rightarrow F^+ \leftarrow \{ \text{ssn} \rightarrow \text{name, city}, \text{ssn} \rightarrow \text{name, city} \}$

$F = \{ \text{ssn} \rightarrow \text{name, city} \}$

<table>
<thead>
<tr>
<th>name</th>
<th>ssn</th>
<th>phone</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andy</td>
<td>123-45-6789</td>
<td>555-555-5555</td>
<td>Pittsburgh</td>
</tr>
<tr>
<td>Andy</td>
<td>123-45-6789</td>
<td>666-666-6666</td>
<td>Pittsburgh</td>
</tr>
<tr>
<td>Lil' Fame</td>
<td>987-65-4321</td>
<td>777-777-7777</td>
<td>Brooklyn</td>
</tr>
<tr>
<td>Lil' Fame</td>
<td>987-65-4321</td>
<td>888-888-8888</td>
<td>Brooklyn</td>
</tr>
</tbody>
</table>
BOYCE-CODD NORMAL FORM (EX.3)

Step #3 – R is not in BCNF
→ 3(a): We choose \( \text{ssn} \rightarrow \text{name}, \text{city} \) as the FD to split on because \( \text{ssn} \) does not get us the phone (i.e., it is not the super key).
→ 3(b): Split \( R \) based on \( \text{ssn} \rightarrow \text{name}, \text{city} \) such that \( R_1 = (\text{name}, \text{ssn}, \text{city}) \) and \( R_2 = (\text{ssn}, \text{phone}) \)

\[
R(\text{name}, \text{ssn}, \text{phone}, \text{city}) \\
F = \{\text{ssn} \rightarrow \text{name}, \text{city}\}
\]

<table>
<thead>
<tr>
<th>name</th>
<th>ssn</th>
<th>phone</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andy</td>
<td>123-45-6789</td>
<td>555-555-5555</td>
<td>Pittsburgh</td>
</tr>
<tr>
<td>Andy</td>
<td>123-45-6789</td>
<td>666-666-6666</td>
<td>Pittsburgh</td>
</tr>
<tr>
<td>Lil' Fame</td>
<td>987-65-4321</td>
<td>777-777-7777</td>
<td>Brooklyn</td>
</tr>
<tr>
<td>Lil' Fame</td>
<td>987-65-4321</td>
<td>888-888-8888</td>
<td>Brooklyn</td>
</tr>
</tbody>
</table>
Step #3: R is not in BCNF  
→ 3(c): The resulting schema is now 
R={R₁,R₂} 

<table>
<thead>
<tr>
<th>name</th>
<th>ssn</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andy</td>
<td>123-45-6789</td>
<td>Pittsburgh</td>
</tr>
<tr>
<td>Lil' Fame</td>
<td>987-65-4321</td>
<td>Brooklyn</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ssn</th>
<th>phone#</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-45-6789</td>
<td>555-555-5555</td>
</tr>
<tr>
<td>123-45-6789</td>
<td>666-666-6666</td>
</tr>
<tr>
<td>987-65-4321</td>
<td>777-777-7777</td>
</tr>
<tr>
<td>987-65-4321</td>
<td>888-888-8888</td>
</tr>
</tbody>
</table>
BOYCE-CODD NORMAL FORM (EX.3)

Step #3: Check whether \{R_1, R_2\} are not in BCNF
→ Lossless?
→ Anomalies?

\[ R_1(\text{name}, \text{ssn}, \text{city}) \]
\[ R_2(\text{ssn}, \text{phone}) \]
\[ F = \{\text{ssn} \rightarrow \text{name}, \text{city}\} \]

<table>
<thead>
<tr>
<th>name</th>
<th>ssn</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andy</td>
<td>123-45-6789</td>
<td>Pittsburgh</td>
</tr>
<tr>
<td>Lil' Fame</td>
<td>987-65-4321</td>
<td>Brooklyn</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ssn</th>
<th>phone#</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-45-6789</td>
<td>555-555-5555</td>
</tr>
<tr>
<td>123-45-6789</td>
<td>666-666-6666</td>
</tr>
<tr>
<td>987-65-4321</td>
<td>777-777-7777</td>
</tr>
<tr>
<td>987-65-4321</td>
<td>888-888-8888</td>
</tr>
</tbody>
</table>

✓ Valid BCNF
A PROBLEM WITH BCNF

\[ R(\text{item}, \text{comp}, \text{category}) \]
\[ F = \{ \text{item} \rightarrow \text{comp}, \ \text{comp}, \text{category} \rightarrow \text{item} \} \]

Super Key: (item, category)
A PROBLEM WITH BCNF

\[ R_1(\text{item, comp}) \rightarrow R_2(\text{item, category}) \]
\[ F = \{ \text{item} \rightarrow \text{comp, comp, category} \rightarrow \text{item} \} \]

<table>
<thead>
<tr>
<th>item</th>
<th>comp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basketball</td>
<td>Pavlo Inc.</td>
</tr>
<tr>
<td>Baseball Bat</td>
<td>Pavlo Inc.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>item</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basketball</td>
<td>Sports Equipment</td>
</tr>
<tr>
<td>Baseball Bat</td>
<td>Sports Equipment</td>
</tr>
</tbody>
</table>

We keep \text{item} \rightarrow \text{comp} but we lose \text{comp, category} \rightarrow \text{item}

At this point we don’t have any problems:
→ We’re in BCNF and all local FDs are satisfied.
A PROBLEM WITH BCNF

$R_1(\text{item,comp})$ $R_2(\text{item,category})$

$F = \{\text{item} \rightarrow \text{comp}, \text{comp,category} \rightarrow \text{item}\}$

<table>
<thead>
<tr>
<th>item</th>
<th>comp</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basketball</td>
<td>Pavlo Inc.</td>
<td>Sports Equipment</td>
</tr>
<tr>
<td>Baseball Bat</td>
<td>Pavlo Inc.</td>
<td>Sports Equipment</td>
</tr>
</tbody>
</table>

Violates $(\text{comp,product} \rightarrow \text{item})$
A PROBLEM WITH BCNF

We started with a relation $R$ and its dependency set $\text{FD}$.

We decomposed $R$ into BCNF relations $\{R_1, \ldots, R_n\}$ with their own $\{\text{FD}_1, \ldots, \text{FD}_n\}$.

We can reconstruct $R$ from $\{R_1, \ldots, R_n\}$.

But we cannot reconstruct $\text{FD}$ from $\{\text{FD}_1, \ldots, \text{FD}_n\}$.
THIRD NORMAL FORM

3NF preserves dependencies but may have some anomalies.

A relation $R$ with FD set $F$ is in 3NF if for every $X \rightarrow Y$ in $F+$:
- $X \rightarrow Y$ is trivial, or
- $X$ is a super key, or
- $Y$ is part of a candidate key
3NF DECOMPOSITION ALGORITHM

Given a relation \( R \) and a FD set \( F \):

Step #1: Compute \( F_c \)

Step #2: \( \text{Result} \leftarrow \emptyset \)

Step #3: For \( (X \rightarrow Y) \in F_c \), add a relation \( R_1(X,Y) \) to \( \text{Result} \)

Step #4: If \( \text{Result} \) is not lossless, add a relation with an appropriate key.
3NF EXAMPLE

Step #1: Compute canonical cover

→ $F_c \leftarrow \{A \rightarrow B, \ B \rightarrow C\}$

$R(A, B, C)$

$F = \{A \rightarrow B, \ B \rightarrow C\}$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>B_A1</td>
<td>C_B_A1</td>
</tr>
<tr>
<td>A2</td>
<td>B_A2</td>
<td>C_B_A2</td>
</tr>
<tr>
<td>A3</td>
<td>B_A3</td>
<td>C_B_A3</td>
</tr>
<tr>
<td>A2</td>
<td>B_A2</td>
<td>C_B_A2</td>
</tr>
</tbody>
</table>
3NF EXAMPLE

Step #3: Split $R$ based on its FDs

$R \rightarrow R_1(A, B)$ because $A \rightarrow B$

$R \rightarrow R_2(B, C)$ because $B \rightarrow C$

$R(A, B, C)$

$F = \{A \rightarrow B, \ B \rightarrow C\}$

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>B_A1</td>
<td>C_B_A1</td>
</tr>
<tr>
<td>A2</td>
<td>B_A2</td>
<td>C_B_A2</td>
</tr>
<tr>
<td>A3</td>
<td>B_A3</td>
<td>C_B_A3</td>
</tr>
<tr>
<td>A2</td>
<td>B_A2</td>
<td>C_B_A2</td>
</tr>
</tbody>
</table>
3NF EXAMPLE

Step #3: Split $R$ based on its FDs

$\rightarrow R_1(A, B)$ because $A \rightarrow B$

$\rightarrow R_2(B, C)$ because $B \rightarrow C$

$F = \{A \rightarrow B, B \rightarrow C\}$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>B_A1</td>
</tr>
<tr>
<td>A2</td>
<td>B_A2</td>
</tr>
<tr>
<td>A3</td>
<td>B_A2</td>
</tr>
<tr>
<td>A2</td>
<td>B_A2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>B_A1</td>
<td>C_B_A1</td>
</tr>
<tr>
<td>B_A2</td>
<td>C_B_A2</td>
</tr>
<tr>
<td>B_A3</td>
<td>C_B_A3</td>
</tr>
<tr>
<td>B_A2</td>
<td>C_B_A2</td>
</tr>
</tbody>
</table>
3NF EXAMPLE

Step #4: Check whether \( \{R_1, R_2\} \) is lossless.

Nope!

Add \( R_3 \) based on join attribute \( A \rightarrow C \)

\[
\begin{array}{ccc}
A & B & C \\
A_1 & B_1 & C_1 \\
A_2 & B_2 & C_2 \\
A_3 & B_3 & C_3 \\
A_2 & B_2 & C_2 \\
\end{array}
\]

\[
R_1(A, B) \quad R_2(B, C) \\
F = \{ A \rightarrow B, \ B \rightarrow C \}
\]

\[
\begin{array}{ccc}
A & B \\
A_1 & B_1 \\
A_2 & B_2 \\
A_3 & B_3 \\
\end{array}
\]

\[
\begin{array}{ccc}
B & C \\
B_1 & C_1 \\
B_2 & C_2 \\
B_3 & C_3 \\
\end{array}
\]
3NF EXAMPLE

Step #4: Check whether \{R_1, R_2\} is lossless.

Nope!
Add \(R_3\) based on join attribute \(A \rightarrow C\)

\[R_1(A, B) \ R_2(B, C) \ R_3(A, C)\]
\[F = \{A \rightarrow B, \ B \rightarrow C\}\]

\[
\begin{array}{|c|c|}
\hline
A & B \\
\hline
A_1 & B_{_A1} \\
A_2 & B_{_A2} \\
A_3 & B_{_A3} \\
A_2 & B_{_A2} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
B & C \\
\hline
B_{_{A1}} & C_{_{B_{_A1}}} \\
B_{_{A2}} & C_{_{B_{_A2}}} \\
B_{_{A2}} & C_{_{B_{_A3}}} \\
B_{_{A2}} & C_{_{B_{_A2}}} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
A & C \\
\hline
A_1 & C_{_{B_{_A1}}} \\
A_2 & C_{_{B_{_A2}}} \\
A_3 & C_{_{B_{_A3}}} \\
\hline
\end{array}
\]

✓ Valid 3NF
BCNF VS. 3NF

BCNF:
→ No anomalies, but may lose some FDS.
→ In practice, this is what you want.

3NF:
→ Keeps all FDs, but may have some anomalies.
→ You usually get this when you convert an ER diagram to SQL.
CONFESSION

The normal forms is usually not how people design databases.

Instead, people usually think in terms of object-oriented programming.
THE RISE OF NOSQL

Prior to the early 2000s, few people needed a high-performance DBMS.

Key tenants of the NoSQL movement:
→ Joins are slow, so we will denormalize tables.
→ Transactions are slow and we need to be on-line 24/7, so let’s drop ACID.
Document Databases

Document Model = JSON / XML

MongoDB supports basic server-side joins. They instead promote "pre-joined" collections by embedding related documents inside of each other.
BCNF EXAMPLE

A customer has orders and each order has order items.

- $R_1(\text{custId}, \text{name}, \ldots)$
- $R_2(\text{orderId}, \text{custId}, \ldots)$
- $R_3(\text{itemId}, \text{orderId}, \ldots)$
A customer has orders and each order has order items.
BCNF EXAMPLE

A customer has orders and each order has order items.

```
{  
  "custId": 1234,
  "custName": "Andy",
  "orders": [
    {
      "orderId": 9999,
      "orderItems": [
        {
          "itemId": "XXXX",
          "price": 19.99
        },
        {
          "itemId": "YYYY",
          "price": 29.99
        }
      ]
    }
  ]
}
```
DENORMALIZATION EXAMPLE

No joins is not a by-product of using the document model, but it makes logical denormalization more “natural”.

Violates the separation between a database's logical layer and its physical layer.
The relational model also supports "nesting" at the physical storage level.
The relational model also supports "nesting" at the physical storage level.

```javascript
db.customers.find(
  {"orders.orderItems": "XXXX"} 
)
```

```sql
SELECT * FROM customers AS c,
   orders AS o,
   order_items AS oi
WHERE c.custId = o.custId
  AND o.orderId = oi.orderId
  AND oi.itemId = "XXXX"
```
CONCLUSION

You should know about normal forms. They exist.

There is no magic formula to determine what is the right amount of normalization for an application.
NEXT CLASS

Database Storage Management