CS460: Intro to Database Systems

Class 7: Decomposition & Schema Normalization

Instructor: Manos Athanassoulis

https://bu-disc.github.io/CS460/
Review: Database Design

Requirements Analysis
user needs; what must database do?

Conceptual Design
high level description (often done w/ ER model)

Logical Design
translate ER into DBMS data model

Schema Refinement
consistency, normalization

Physical Design
indexes, disk layout
Why schema refinement

what is a bad schema?

*a schema with redundancy!*

why?

redundant storage & insert/update/delete anomalies

how to fix it?

*normalize* the schema by decomposing

normal forms: BCNF, 3NF, ...
Motivating Example

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Salary</th>
<th>Telephone</th>
</tr>
</thead>
<tbody>
<tr>
<td>987-00-8761</td>
<td>John</td>
<td>65K</td>
<td>857-555-1234</td>
</tr>
<tr>
<td>987-00-8761</td>
<td>John</td>
<td>65K</td>
<td>857-555-8800</td>
</tr>
<tr>
<td>123-00-9876</td>
<td>Anna</td>
<td>80K</td>
<td>617-555-9876</td>
</tr>
<tr>
<td>787-00-4321</td>
<td>John</td>
<td>25K</td>
<td>617-555-3761</td>
</tr>
</tbody>
</table>

SSN $\rightarrow$ Name, Salary
Motivating Example

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Salary</th>
<th>Telephone</th>
</tr>
</thead>
<tbody>
<tr>
<td>987-00-8761</td>
<td>John</td>
<td>65K</td>
<td>857-555-1234</td>
</tr>
<tr>
<td>987-00-8761</td>
<td>John</td>
<td>65K</td>
<td>857-555-8800</td>
</tr>
<tr>
<td>123-00-9876</td>
<td>Anna</td>
<td>80K</td>
<td>617-555-9876</td>
</tr>
<tr>
<td>787-00-4321</td>
<td>John</td>
<td>25K</td>
<td>617-555-3761</td>
</tr>
</tbody>
</table>

SSN → Name, Salary

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>987-00-8761</td>
<td>John</td>
<td>65K</td>
</tr>
<tr>
<td>123-00-9876</td>
<td>Anna</td>
<td>80K</td>
</tr>
<tr>
<td>787-00-4321</td>
<td>John</td>
<td>25K</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>Telephone</th>
</tr>
</thead>
<tbody>
<tr>
<td>987-00-8761</td>
<td>857-555-1234</td>
</tr>
<tr>
<td>987-00-8761</td>
<td>857-555-8800</td>
</tr>
<tr>
<td>123-00-9876</td>
<td>617-555-9876</td>
</tr>
<tr>
<td>787-00-4321</td>
<td>617-555-3761</td>
</tr>
</tbody>
</table>
## Motivating Example 2

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
<th>color</th>
<th>price</th>
<th>department</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPhone</td>
<td>smartphone</td>
<td>black</td>
<td>600</td>
<td>phones</td>
</tr>
<tr>
<td>Lenovo Yoga</td>
<td>laptop</td>
<td>grey</td>
<td>800</td>
<td>computers</td>
</tr>
<tr>
<td>unifi</td>
<td>networking</td>
<td>white</td>
<td>150</td>
<td>computers</td>
</tr>
<tr>
<td>unifi</td>
<td>cables</td>
<td>white</td>
<td>10</td>
<td>stationary</td>
</tr>
<tr>
<td>OnePlus</td>
<td>smartphone</td>
<td>silver</td>
<td>450</td>
<td>phones</td>
</tr>
</tbody>
</table>

name, category $\rightarrow$ price, color  
category $\rightarrow$ department
Motivating Example 2

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
<th>color</th>
<th>price</th>
<th>department</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPhone</td>
<td>smartphone</td>
<td>black</td>
<td>600</td>
<td>phones</td>
</tr>
<tr>
<td>Lenovo Yoga</td>
<td>laptop</td>
<td>grey</td>
<td>800</td>
<td>computers</td>
</tr>
<tr>
<td>unifi</td>
<td>networking</td>
<td>white</td>
<td>150</td>
<td>computers</td>
</tr>
<tr>
<td>unifi</td>
<td>cables</td>
<td>white</td>
<td>10</td>
<td>stationary</td>
</tr>
<tr>
<td>OnePlus</td>
<td>smartphone</td>
<td>silver</td>
<td>450</td>
<td>phones</td>
</tr>
</tbody>
</table>

name, category $\rightarrow$ price, color $\leftarrow$ category $\rightarrow$ department
Reminder: Reasoning for FDs

Assume a relation R with attributes A, B, C

(1) reflexivity e.g., $AB \rightarrow B$

(2) augmentation e.g., if $A \rightarrow B$ then $AC \rightarrow BC$

(3) transitivity e.g., if $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$

(4) union e.g., if $A \rightarrow B$ and $A \rightarrow C$ then $A \rightarrow BC$

(5) decomposition e.g., if $A \rightarrow BC$ then $A \rightarrow B$ and $A \rightarrow C$

FD closure of $F$, $F^+$: is the set of all FDs that are implied by $F$

attr. closure of $X$: the set of all attributes that are determined by $X$

minimal cover: subset $S$ of $F^+$ such that $S^+ = F^+$
“chopping the relation into pieces using FDs”

DECOMPOSITION
Decomposition

Formally

we decompose $R(A_1, \ldots, A_n)$ by creating:

$R_1(B_1, \ldots, B_m)$

$R_2(C_1, \ldots, C_k)$

where $\{B_1, \ldots, B_m\} \cup \{C_1, \ldots, C_k\} = \{A_1, \ldots, A_n\}$

the instance of $R_1$ is the projection of $R$ onto $B_1, \ldots, B_m$

the instance of $R_2$ is the projection of $R$ onto $C_1, \ldots, C_k$
“Good” Decomposition

(1) minimize redundancy

(2) avoid information loss (lossless-join)

(3) preserve FDs (dependency preserving)

(4) ensure good query performance
Information Loss

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Salary</th>
<th>Telephone</th>
</tr>
</thead>
<tbody>
<tr>
<td>987-00-8761</td>
<td>John</td>
<td>65K</td>
<td>857-555-1234</td>
</tr>
<tr>
<td>987-00-8761</td>
<td>John</td>
<td>65K</td>
<td>857-555-8800</td>
</tr>
<tr>
<td>123-00-9876</td>
<td>Anna</td>
<td>80K</td>
<td>617-555-9876</td>
</tr>
<tr>
<td>787-00-4321</td>
<td>John</td>
<td>25K</td>
<td>617-555-3761</td>
</tr>
</tbody>
</table>

Decompose into:
- $R_1(\text{SSN, Name, Salary})$
- $R_2(\text{Name, Telephone})$

**can we reconstruct R?**
Lossless Decomposition

\[ R(A,B,C) \]

\[ \text{decompose} \]

\[ R_1(A,B) \]

\[ R_2(B,C) \]

\[ R'(A,B,C) \]

\[ \text{recover (join on } B) \]

the decomposition is lossless-join if for any initial instance \( R, R = R' \)
Lossless Criterion

given:
• a relation $R(A)$
• a set $F$ of FDs
• a decomposition of $R$ into $R_1(A_1)$ and $R_2(A_2)$

the decomposition is lossless-join if and only if
at least one of the following FDs is in $F^+$ (closure of $F$):
(1) $A_1 \cap A_2 \rightarrow A_1$
(2) $A_1 \cap A_2 \rightarrow A_2$
Example

Relation $R(A, B, C, D)$
FD $A \rightarrow BC$

$what \ is \ the \ F^+$?

$lossy$

decomposition into $R_1(A, B, C)$ and $R_2(D)$

$lossless\text{-}join$

decomposition into $R_1(A, B, C)$ and $R_2(A, D)$

$A_1 \cap A_2$ empty set

$A_1 \cap A_2 = A$ and $A_1 = A, B, C$

$A \rightarrow ABC$ is in $F^+$
Dependency Preserving

given $R$ and a set of FDs $F$, we decompose $R$ into $R_1$ and $R_2$. Suppose:

- $R_1$ has a set of FDs $F_1$
- $R_2$ has a set of FDs $F_2$

$F_1$ and $F_2$ are computed from $F$

it is dependency preserving if by enforcing $F_1$ over $R_1$ and $F_2$ over $R_2$, we can enforce $F$ over $R$
(Good) Example

**Person** (SSN, name, age, canDrink)

SSN → name, age

age → canDrink

What is a **dependency preserving** decomposition?

\[ R_1(\text{SSN}, \text{name}, \text{age}) \text{ and } R_2(\text{age}, \text{canDrink}) \]

<table>
<thead>
<tr>
<th>SSN</th>
<th>name</th>
<th>age</th>
<th>canDrink</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>name</td>
<td>age</td>
<td>canDrink</td>
</tr>
<tr>
<td>age</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Is it also lossless-join?

Yes! \( A_1 \cap A_2 = \text{age} \) and \( A_2 = \text{age, canDrink} \)

age → age, canDrink is in \( F^+ \)
(Bad) Example

\[ R (A, B, C) \]
\[ A \rightarrow B \]
\[ BC \rightarrow A \]

not dependency preserving

\[ R_1(A, B) \quad \text{and} \quad R_2(A, C) \]
\[ A \rightarrow B \quad \text{no FDs!} \]
Normal Forms

How “good” is a schema design?
follows normal forms

1NF
2NF
3NF
BCNF
4NF
...

flat tables
atomic values

more restrictive
Normal Forms

How “good” is a schema design? follows normal forms

1NF
2NF
3NF
BCNF
4NF
...

flat tables
atomic values

more restrictive
Boyce-Codd Normal Form (BCNF)

given a relation $R(A_1,\ldots,A_n)$,
a set of FDs $F$, and $X \subseteq \{A_1,\ldots,A_n\}$

$R$ is in BCNF if $\forall X \rightarrow A$ one of the two holds:

- $A \in X$ (that is, it is a trivial FD)
- $X$ is a superkey

in other words: $\forall \text{non-trivial FD } X \rightarrow A, X$ is a superkey in $R$
BCNF - Example

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Salary</th>
<th>Telephone</th>
</tr>
</thead>
<tbody>
<tr>
<td>987-00-8761</td>
<td>John</td>
<td>65K</td>
<td>857-555-1234</td>
</tr>
<tr>
<td>987-00-8761</td>
<td>John</td>
<td>65K</td>
<td>857-555-8800</td>
</tr>
<tr>
<td>123-00-9876</td>
<td>Anna</td>
<td>80K</td>
<td>617-555-9876</td>
</tr>
<tr>
<td>787-00-4321</td>
<td>John</td>
<td>25K</td>
<td>617-555-3761</td>
</tr>
</tbody>
</table>

SSN → Name, Salary

key: {SSN, Telephone}

FD is not trivial!

so, is SSN a superkey?

no! it is not in BCNF
BCNF - Example 2

SSN ➝ Name, Salary
key: \{SSN\}

FD is not trivial!
so, is SSN a superkey?
yes! it is in **BCNF**
BCNF - Example 3

<table>
<thead>
<tr>
<th>SSN</th>
<th>Telephone</th>
</tr>
</thead>
<tbody>
<tr>
<td>987-00-8761</td>
<td>857-555-1234</td>
</tr>
<tr>
<td>987-00-8761</td>
<td>857-555-8800</td>
</tr>
<tr>
<td>123-00-9876</td>
<td>617-555-9876</td>
</tr>
<tr>
<td>787-00-4321</td>
<td>617-555-3761</td>
</tr>
</tbody>
</table>

key: \{SSN, Telephone\} the relation is in **BCNF**

why? Is it possible a binary relation to **not** be in **BCNF**?

no FDs
Binary Relations always BCNF

\( R \ (A,B) \)

excluding all trivial FDs, there are three cases:

1. \( R \) has no FD
2. \( R \) has one FD, either \( A \rightarrow B \) or \( B \rightarrow A \), or,
3. \( R \) has two FDs, \( A \rightarrow B \) and \( B \rightarrow A \)

(1) trivially in BCNF
(2) in either LHS is the key (hence, superkey)
(3) both, A and B candidate keys
BCNF Decomposition Algorithm

(1) find a FD that violates BCNF:
\[ A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m \]

(2) decompose \( R \) to \( R_1 \) and \( R_2 \)

\[ R_1(A_1, \ldots, A_n, B_1, \ldots, B_m) \]
\[ R_2(A_1, \ldots, A_n, \text{all other attributes of } R) \]

(3) repeat until no BCNF violations are left
(in new tables as well)
Our favorite example!

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Salary</th>
<th>Telephone</th>
</tr>
</thead>
<tbody>
<tr>
<td>987-00-8761</td>
<td>John</td>
<td>65K</td>
<td>857-555-1234</td>
</tr>
<tr>
<td>987-00-8761</td>
<td>John</td>
<td>65K</td>
<td>857-555-8800</td>
</tr>
<tr>
<td>123-00-9876</td>
<td>Anna</td>
<td>80K</td>
<td>617-555-9876</td>
</tr>
<tr>
<td>787-00-4321</td>
<td>John</td>
<td>25K</td>
<td>617-555-3761</td>
</tr>
</tbody>
</table>

**SSN \rightarrow Name, Salary** violates BCNF

$A_1 = SSN, B_1 = Name, B_2 = Salary$

Split in two relations:

$R_1(\text{SSN, Name, Salary})$

$R_2(\text{SSN, Telephone})$
Our favorite example!

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Salary</th>
<th>Telephone</th>
</tr>
</thead>
<tbody>
<tr>
<td>987-00-8761</td>
<td>John</td>
<td>65K</td>
<td>857-555-1234</td>
</tr>
<tr>
<td>987-00-8761</td>
<td>John</td>
<td>65K</td>
<td>857-555-8800</td>
</tr>
<tr>
<td>123-00-9876</td>
<td>Anna</td>
<td>80K</td>
<td>617-555-9876</td>
</tr>
<tr>
<td>787-00-4321</td>
<td>John</td>
<td>25K</td>
<td>617-555-3761</td>
</tr>
</tbody>
</table>

**SSN** | **Name** | **Salary** | **Telephone** |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>987-00-8761</td>
<td>John</td>
<td>65K</td>
<td>857-555-1234</td>
</tr>
<tr>
<td>987-00-8761</td>
<td>John</td>
<td>65K</td>
<td>857-555-8800</td>
</tr>
<tr>
<td>123-00-9876</td>
<td>Anna</td>
<td>80K</td>
<td>617-555-9876</td>
</tr>
<tr>
<td>787-00-4321</td>
<td>John</td>
<td>25K</td>
<td>617-555-3761</td>
</tr>
</tbody>
</table>
BCNF Decomposition Properties

removes [certain types of] redundancy

is lossless-join

is not always dependency preserving
BCNF – Lossless Join

Example

\[ R \ (A, B, C) \text{ and } \text{FD: } A \rightarrow B \]

superkey(s) of the relation?

\{A, C\}^+, \{A, B, C\}^+ = \{A, B, C\}

\[ A \rightarrow B \text{ violates BCNF (A is not a superkey) } \]

so, the BCNF decomposition is:

\[ R_1(A, B) \text{ and } R_2(A, C) \]

we can reconstruct it!
BCNF – not dependency preserving

Example

R (A, B, C), FDs: \( A \rightarrow B \) and \( BC \rightarrow A \)

superkey(s) of the relation?

\( \{A, C\}^+, \{B, C\}^+, \{A, B, C\}^+ = \{A, B, C\} \)

\( BC \rightarrow A \) is ok, but \( A \rightarrow B \) violates BCNF

so, the BCNF decomposition is:

\( R_1 (A, B) \) and \( R_2 (A, C) \)

\( A \rightarrow B \) is preserved in \( R_1 \)

\( BC \rightarrow A \) is not preserved!
**BCNF Decomposition Examples**

**Books** (author, gender, booktitle, genre, price)

- author $\rightarrow$ gender
- booktitle $\rightarrow$ genre, price

candidate key(s)?

{author, booktitle} is the only one

Is it in BCNF? No, because LHS of both FD are not a superkey!
BCNF Decomposition Examples

**Books** (author, gender, booktitle, genre, price)

- \( author \rightarrow gender \)
- \( booktitle \rightarrow genre, price \)

Splitting using: \( author \rightarrow gender \)

**AuthorInfo** (author, gender)

- \( FD \) \( author \rightarrow gender \) *(in BCNF!)*

**Book2** (author, booktitle, genre, price)

- \( FD \) \( booktitle \rightarrow genre, price \)

Is booktitle a superkey? No! \{booktitle, author\} is.

So not in BCNF!
BCNF Decomposition Examples

Books (author, gender, booktitle, genre, price)

author $\rightarrow$ gender
booktitle $\rightarrow$ genre, price

AuthorInfo (author, gender)
Further splitting with booktitle $\rightarrow$ genre, price

Book2 (author, booktitle, genre, price)

BookInfo (booktitle, genre, price)

FD booktitle $\rightarrow$ genre, price in BCNF!

is booktitle a superkey? Yes!

BookAuthor (booktitle, author) binary is in BCNF!
what if not dependency preserving?

in some cases BCNF decomposition is not dependency preserving

how to address this?

relax the normalization requirements
Third Normal Form (3NF)

given a relation $R (A_1,\ldots,A_n)$, a set of FDs $F$, and $X \subseteq \{A_1,\ldots,A_n\}$

$R$ is in 3NF if $\forall X \rightarrow A$ one of the three holds:

- $A \in X$ (that is, it is a trivial FD)
- $X$ is a superkey
- $A$ is part of some candidate key for $R$

is a relation in 3NF also in BCNF?

No, but a relation in BCNF is always in 3NF!
Third Normal Form (3NF)

Example

\[ R \ (A, B, C), \text{ FDs } C \rightarrow A \\text{ and } AB \rightarrow C \]

is in 3NF but not in BCNF. Why?

superkeys?
\{A, B\}, \{B, C\}, \text{ and } \{A, B, C\}

candidate keys?
\{A, B\}\text{ and } \{B, C\}

Compromise: aim for BCNF but settle for 3NF
lossless-join & dependency preserving possible
3NF Algorithm

1. Apply BCNF until all relations are in 3NF

2. Compute a minimal cover $F'$ of $F$

3. For each non-preserved FD $X \rightarrow A$ in $F'$
   add a new relation $R (X, A)$
3NF algorithm example

Assume $R$ (A, B, C, D)

$A \rightarrow D$
$AB \rightarrow C$
$AD \rightarrow C$
$B \rightarrow C$
$D \rightarrow AB$

Step 1: find a BCNF decomposition

$R_1$ (B, C)
$R_2$ (A, B, D)
3NF algorithm example

Assume $R$ (A, B, C, D)

$A \rightarrow D$
$AB \rightarrow C$ can be expressed via: $AB \rightarrow AC$ which gives $AB \rightarrow A$ and $AB \rightarrow C$
$AD \rightarrow C$ can be expressed via: $D \rightarrow AB$, which gives $D \rightarrow A$ and $D \rightarrow B$ & $B \rightarrow C$
$B \rightarrow C$
$D \rightarrow AB$ which is simplified to $D \rightarrow A$ and $D \rightarrow B$

Step 2: find a minimal cover

$A \rightarrow D$
$B \rightarrow C$
$D \rightarrow A$
$D \rightarrow B$
3NF algorithm example

Assume \( R \) (A, B, C, D)

\[
\begin{align*}
A & \rightarrow D \\
AB & \rightarrow C \\
AD & \rightarrow C \\
B & \rightarrow C \\
D & \rightarrow AB
\end{align*}
\]

**Step 3:** add a new relation for not preserved FDs

\[
\begin{align*}
A & \rightarrow D \\
B & \rightarrow C \\
D & \rightarrow A \\
D & \rightarrow B \\
R_1 & (B, C) \\
R_2 & (A, B, D)
\end{align*}
\]

all FD are preserved!

both are in BCNF!
Is Normalization Always Good?

Example 1: suppose A and B are always used together, but normalization puts them in different tables (e.g., hours_worked and hourly_rate)

decomposition might produce unacceptable performance loss

Example 2: data warehouses
huge historical DBs, rarely updated after creation
joins expensive or impractical
[we want “flat” tables, a.k.a, denormalized]
**Example**

R (C, S, J, D, P, Q, V)

C → SJDPQV

JP → C

SD → P

J → S

**Step 1:**

R₁ (S, D, P)

R₂ (C, S, J, D, Q, V)

superkeys?

{C}, {J, P}, {D, J}, …

not {S, D}
Example

R (C, S, J, D, P, Q, V)
C \rightarrow SJDPQV
JP \rightarrow C
SD \rightarrow P
J \rightarrow S

**Step 1b:**
R_1 (S, D, P)
R_2 (C, S, J, D, Q, V)
R_2' (J, S)
R_3 (C, J, D, Q, V)

superkeys of \( R_2 \) (C, S, J, D, Q, V)?
\{C\}, ... not \{J\}
Example

R (C, S, J, D, P, Q, V)
C → SJDPQV
JP → C
SD → P
J → S

Step 2: Minimal Cover

C → J, C → D, C → Q, C → V
JP → C
SD → P
J → S

R_1 (S, D, P)
R_2' (J, S)
R_3 (C, J, D, Q, V)
R_4 (J, P, C)

are they all preserved?

No!

Step 3: need to add R_4 (J, P, C)
Example

R (C, S, J, D, P, Q, V)
C → SJDPQV
JP → C
SD → P
J → S

**Step 2: Minimal Cover**

\[ C → J, \ C → D, \ C → Q, \ C → V \]
\[ JP → C \]
SD → P
J → S

\( R_1 (S, D, P) \)
\( R_2' (J, S) \)
\( R_3 (C, J, D, Q, V) \)
\( R_4 (J, P, C) \)

are they all preserved?

**No!**

**Step 3: need to add** \( R_4 (J, P, C) \)

*did we just introduce redundancy?*
Lesson!

theory of normalization is a guide

cannot always give a “perfect” solution

redundancy
alternatives
query performance
Summary

fix bad schemas (redundancy) by decomposition

lossless-join

dependency preserving

Desired normal forms

**BCNF:** only superkey FDs

**3NF:** superkey FDs + dependencies to prime attributes in RHS

**Next:** SQL