Comp 115: Databases

Decomposition & Schema Normalization

Instructor: Manos Athanassoulis

http://www.cs.tufts.edu/comp/115/
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?
\textit{a schema with redundancy!}

why?
redundant storage & insert/update/delete anomalies

how to fix it?
\textit{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 → 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 → price, color

category → 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>

<table>
<thead>
<tr>
<th>category</th>
<th>department</th>
</tr>
</thead>
<tbody>
<tr>
<td>laptop</td>
<td>computers</td>
</tr>
<tr>
<td>networking</td>
<td>computers</td>
</tr>
<tr>
<td>cables</td>
<td>stationary</td>
</tr>
<tr>
<td>smartphone</td>
<td>phones</td>
</tr>
</tbody>
</table>
“chopping the relation into pieces using FDs”

DECOMPOSITION
Decomposition

Formally
we decompose $R(A_1, ..., A_n)$ by creating:

$R_1(B_1, ..., B_m)$

$R_2(C_1, ..., C_k)$

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

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

the instance of $R_2$ is the projection of $R$ onto $C_1, ..., C_k$
## 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>

---

<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>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>
“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>
</tr>
</thead>
<tbody>
<tr>
<td>987-00-8761</td>
<td>John</td>
<td>65K</td>
</tr>
<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>

Decompose into:

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

Can we reconstruct $R$?
The decomposition is lossless-join if for any initial instance $R$, $R = R'$.

**Graph:***

- $R(A, B, C)$
- $R_1(A, B)$
- $R_2(B, C)$
- $R'(A, B, C)$

Decompose

Recover (join on $B$)
Lossless Criterion

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

the decomposition is \textit{lossless-join if and only if} at least one of the 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 $A \rightarrow B, C$

**lossy**

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

$A_1 \cap A_2$ empty set

**lossless-join**

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

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

$A \rightarrow A, B, C$ 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 $\rightarrow$ name, age

age $\rightarrow$ canDrink

what is a **dependency preserving** decomposition?

$R_1$(SSN, name, age) and $R_2$(age, canDrink)

SSN $\rightarrow$ name, age

age $\rightarrow$ canDrink

Is it also lossless-join?

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

age $\rightarrow$ age, canDrink *is in* $F^+$
(Bad) Example

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

*not dependency preserving*

\( R_1(A, \ B) \) and \( R_2(A, \ C) \)
\[ A \rightarrow B \]

*no FDs!*

the table violates
\[ B, \ C \rightarrow A \]
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
...

more restrictive

flat tables
atomic values
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

[alternatively, $\forall$ non-trivial FD $X \rightarrow A$, $X$ is a superkey in $R$]
BCNF - Example

### Table

<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

<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>

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?  
no FDs

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

Find a FD that violates BCNF:

\[ A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m \]

Decompose \( R \) to \( R_1 \) and \( R_2 \)

\[
R_1(A_1, \ldots, A_n, B_1, \ldots, B_m)
\]

\[
R_2(A_1, \ldots, A_n, \text{all other attributes})
\]

continue 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>

<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>
removes [certain types of] redundancy

is lossless-join

is not always dependency preserving
BCNF – Lossless Join

Example

\( R \ (A, B, C) \) and FD: \( A \rightarrow B \)

superkey(s) of the relation?

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

\( A \rightarrow B \) violates BCNF (A is not a superkey)

so, the BCNF decomposition is:

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

we can reconstruct it!
BCNF – not dependency preserving

Example

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

superkey(s) of the relation?

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

\( B, C \rightarrow A \) 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 \( R_1 \)

\( B, C \rightarrow A \) 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 → gender
booktitle → genre, price

AuthorInfo (author, gender)

Further splitting with booktitle → genre, price

Book2 (author, booktitle, genre, price)

BookAuthor (booktitle, author) binary is in BCNF!

BookInfo (booktitle, genre, price) in BCNF!

FD booktitle → genre, price

is booktitle a superkey? Yes!
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 $R (A_1,...,A_n)$,
a set of FDs $F$, and $X \subseteq \{A_1,...,A_n\}$
$R$ 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 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), FDs C → A and A, B → C
is in 3NF but not in BCNF. Why?

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

candidate keys?
{A, B} 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)

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

\( A \rightarrow D \) \quad \text{superkeys?} \quad \{A\} \ \{D\} \ \{A, B\} \ \{A, D\}, \ldots

\( B \rightarrow C \) \quad \text{not} \ \{B\}

**Step 1:** find a BCNF decomposition

\( R_1 \) (B, C)

\( R_2 \) (A, B, D)
3NF algorithm example

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

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

**Step 2:** find a minimal cover

\[
\begin{align*}
A & \rightarrow D \\
B & \rightarrow C \\
D & \rightarrow A \\
D & \rightarrow B
\end{align*}
\]
### 3NF algorithm example

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

- $A \rightarrow D$
- $A, B \rightarrow C$
- $A, D \rightarrow C$
- $B \rightarrow C$
- $D \rightarrow A, B$

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

- $A \rightarrow D$
- $B \rightarrow C$
- $D \rightarrow A$
- $D \rightarrow B$

$R_1$ (B, C)

$R_2$ (A, B, D)

all FD are preserved!

both are in BCNF!
Is Normalization Always Good?

**Example 1:** suppose A and B are always used together, but normalization says they should be 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 → S, J, D, P, Q, V
J, P → C
S, D → 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)  superkeys?
C → S, J, D, P, Q, V  \{C\}, \{J, P\}, \{D, J\}, ...
J, P → C  \text{not} \ \{S, D\}
S, D → P  superkeys of R_2 (C, S, J, D, Q, V)?
J → S  \{C\}, ... \text{not} \ \{J\}

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

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

Step 2: Minimal Cover
  C → J, C → D, C → Q, C → V
  J, P → C
S, D → P
J → S

R₁ (S, D, P)
R₂ (J, S)
R₃ (C, J, D, Q, V)
R₄ (J, P, C)

are they all preserved?

No!

Step 3: need to add R₄ (J, P, C)
Example

\[ R (C, S, J, D, P, Q, V) \]
\[ C \rightarrow S, J, D, P, Q, V \]
\[ J, P \rightarrow C \]
\[ S, D \rightarrow P \]
\[ J \rightarrow S \]

**Step 2: Minimal Cover**

\[ C \rightarrow J, C \rightarrow D, C \rightarrow Q, C \rightarrow V \]
\[ J, P \rightarrow C \]
\[ S, D \rightarrow P \]
\[ J \rightarrow S \]

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

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: execution of queries
  
  procedural & declarative