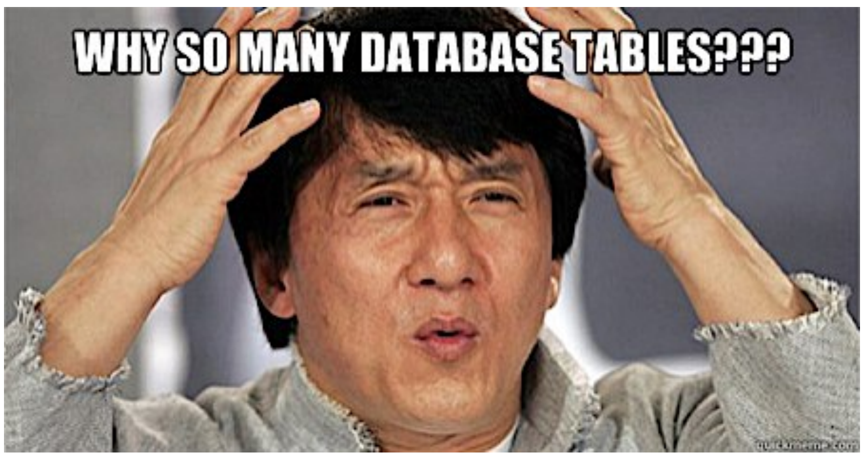
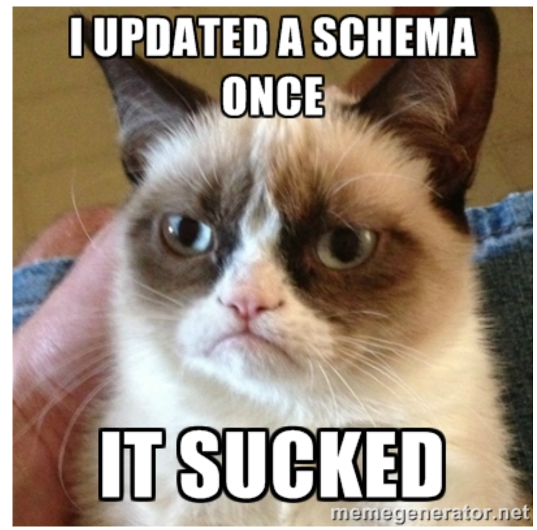
Introduction to Data Management
CSE 344

Lectures 18: BCNF

What makes good schemas?





Review: Relation Decomposition

Break the relation into two:

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield

Name	SSN	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Westfield

SSN	PhoneNumber
123-45-6789	206-555-1234
123-45-6789	206-555-6543
987-65-4321	908-555-2121

Anomalies have gone:

- No more repeated data
- Easy to move Fred to “Bellevue” (how ?)
- Easy to delete all Joe’s phone numbers (how ?)

Review: Functional Dependencies (FDs)

Definition

If two tuples agree on the attributes

$\boxed{A_1, A_2, \ldots, A_n}$

then they must also agree on the attributes

$\boxed{B_1, B_2, \ldots, B_m}$

Formally:

$A_1 \ldots A_n$ **determines** $B_1 .. B_m$

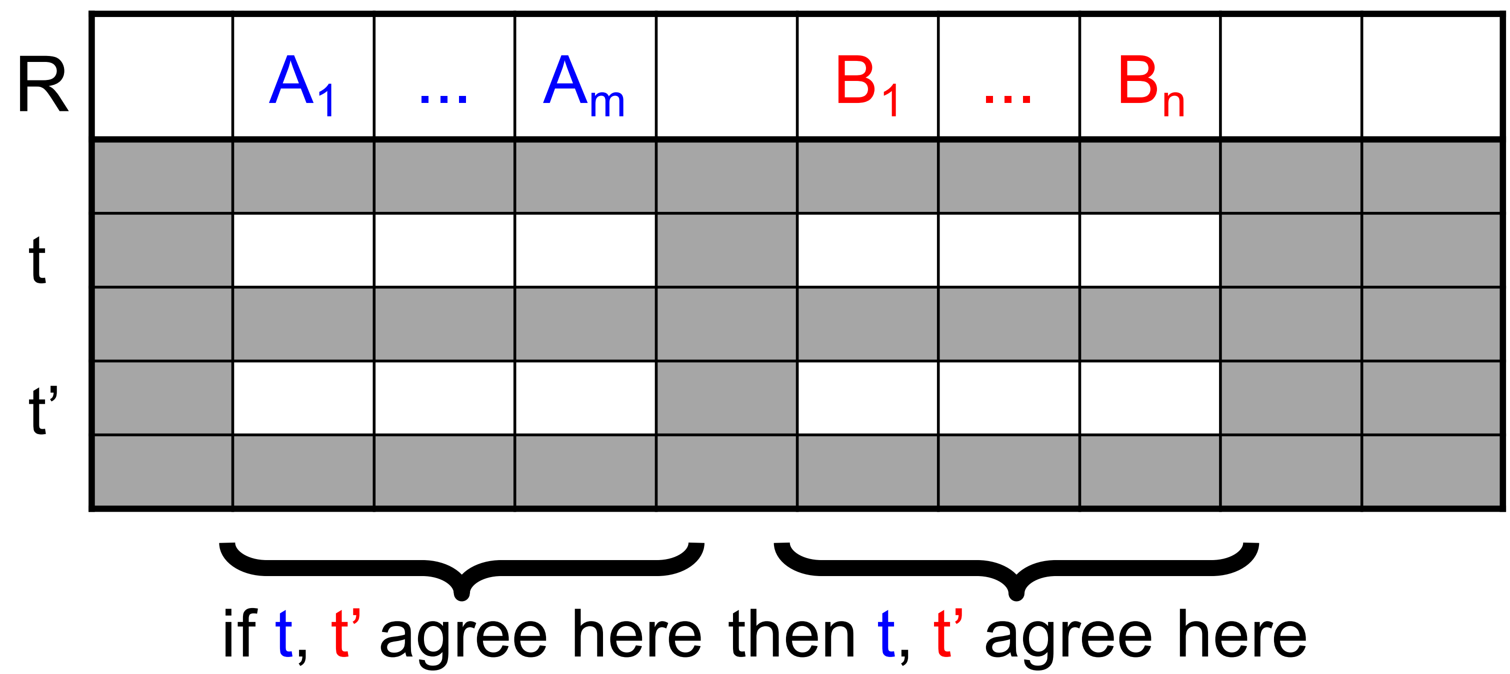
$\boxed{A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m}$

Review: Functional Dependencies (FDs)

Definition $A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n$ **holds** in R if:

$\forall t, t' \in R,$

$(t.A_1 = t'.A_1 \wedge \ldots \wedge t.A_m = t'.A_m \rightarrow t.B_1 = t'.B_1 \wedge \ldots \wedge t.B_n = t'.B_n)$



Review: An Interesting Observation

If all these FDs are true:

name → color
category → department
color, category → price

Then this FD also holds:

name, category → price

If we find out from application domain that a relation satisfies some FDs, it doesn’t mean that we found all the FDs that it satisfies!
There could be more FDs implied by the ones we have.

Review:
Closure of a set of Attributes

Given a set of attributes $A_1, \ldots, A_n$

The **closure** is the set of attributes B, notated $\{A_1, \ldots, A_n\}^+$,
s.t. $A_1, \ldots, A_n \rightarrow B$

Example:

1. name → color
2. category → department
3. color, category → price

Closures:

$\text{name}^+ = \{\text{name, color}\}$

$\{\text{name, category}\}^+ = \{\text{name, category, color, department, price}\}$

$\text{color}^+ = \{\text{color}\}$

Closure Algorithm

$X=\{A_1, \ldots, A_n\}$.

Repeat until X doesn't change **do**:
  if $B_1, \ldots, B_n \rightarrow C$ is a FD **and**
  $B_1, \ldots, B_n$ are all in X
  then add C to X.

Example:

1. name → color
2. category → department
3. color, category → price

{name, category }$^+$ =
{ name, category, color, department, price }

Hence: $\boxed{\text{name, category} \rightarrow \text{color, department, price}}$

Example

In class:

R(A,B,C,D,E,F)

A, B	$\rightarrow$	C
A, D	$\rightarrow$	E
B	$\rightarrow$	D
A, F	$\rightarrow$	B

Compute $\{A, B\}^+$ X = {A, B, }

Compute $\{A, F\}^+$ X = {A, F, }

Example

In class:

R(A,B,C,D,E,F)

A, B	$\rightarrow$	C
A, D	$\rightarrow$	E
B	$\rightarrow$	D
A, F	$\rightarrow$	B

Compute $\{A, B\}^+$ X = {A, B, C, D, E }

Compute $\{A, F\}^+$ X = {A, F, }

Example

In class:

R(A,B,C,D,E,F)

A, B	$\rightarrow$	C
A, D	$\rightarrow$	E
B	$\rightarrow$	D
A, F	$\rightarrow$	B

Compute $\{A, B\}^+$ X = {A, B, C, D, E }

Compute $\{A, F\}^+$ X = {A, F, B, C, D, E }

Example

In class:

R(A,B,C,D,E,F)

$$A, B \rightarrow C$$
$$A, D \rightarrow E$$
$$B \rightarrow D$$
$$A, F \rightarrow B$$

Compute $\{A, B\}^+$ $X = \{A, B, C, D, E\}$

Compute $\{A, F\}^+$ $X = \{A, F, B, C, D, E\}$

What is a key of R?

Keys

- A **superkey** is a set of attributes $A_1, ..., A_n$ s.t. for any other attribute B in the same relation, we have $A_1, ..., A_n \rightarrow B$

- A **key** is a minimal superkey
  – A superkey and for which no subset is a superkey

Computing (Super)Keys

- For all sets X, compute X^+
- If X^+ = [all attributes], then X is a superkey
- Try reducing to the minimal X's to get the key

Example

Product(name, price, category, color)

name, category $\rightarrow$ price
category $\rightarrow$ color

What is the key ?

Example

Product(name, price, category, color)

$$\boxed{\begin{array}{l}\text{name, category} \rightarrow \text{price} \\ \text{category} \rightarrow \text{color}\end{array}}$$

What is the key ?

(name, category)$^+$ = { name, category, price, color }

Hence (name, category) is a key

Key or Keys ?

Can we have more than one key ?

Given R(A,B,C) define FD's s.t. there are two or more distinct keys

Key or Keys ?

Can we have more than one key ?

Given R(A,B,C) define FD's s.t. there are two or more distinct keys

$A \rightarrow B$ $B \rightarrow C$ $C \rightarrow A$	or	$AB \rightarrow C$ $BC \rightarrow A$	or	$A \rightarrow BC$ $B \rightarrow AC$

what are the keys here ?

Eliminating Anomalies

Main idea:

- X → A is OK if X is a (super)key
- X → A is not OK otherwise
  - Need to decompose the table, but how?

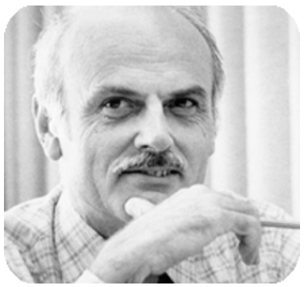
Boyce-Codd Normal Form

Boyce-Codd Normal Form

Dr. Raymond F. Boyce

Turing Awards in Data Management

Charles Bachman, 1973
IDS and CODASYL



Ted Codd, 1981
Relational model

Jim Gray, 1998
Transaction processing

Michael Stonebraker, 2014
INGRES and Postgres

Boyce-Codd Normal Form

There are no "bad" FDs:

Definition. A relation R is in BCNF if:

Whenever $X \rightarrow B$ is a non-trivial dependency, then X is a superkey.

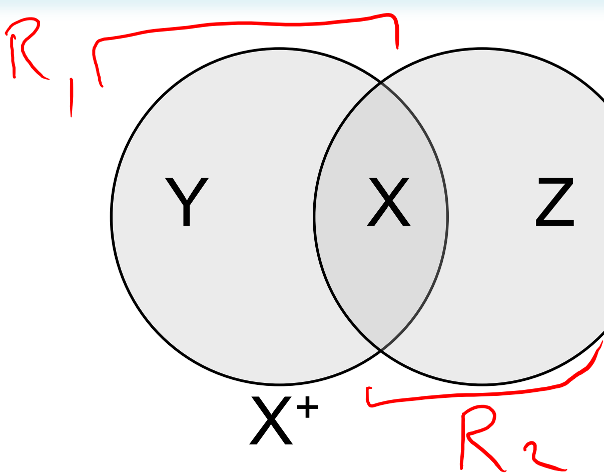
Equivalently:

Definition. A relation R is in BCNF if:

$\forall X$ in $X \rightarrow B$,
either $X^+ = X$ or $X^+ =$ [all attributes]

BCNF Decomposition Algorithm

Normalize(R)
  find X in X→B s.t.: X $\neq$ X^+ and $X^+ \neq$ [all attributes]
  if (not found) **then** R is in BCNF
  let Y = X^+ - X;   Z = [all attributes] - X^+
  decompose R into R1(X ∪ Y) and R2(X ∪ Z)
  Normalize(R1); Normalize(R2);

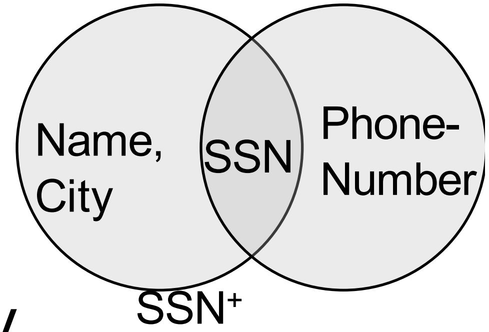


Want X in X→B s.t.: X = X^+ or X^+ = [all attributes]

Example

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield
Joe	987-65-4321	908-555-1234	Westfield

SSN → Name, City



The only key is: {SSN, PhoneNumber}
Hence SSN → Name, City is a "bad" dependency

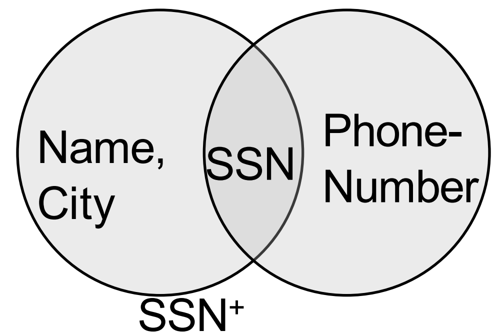
In other words:
SSN+ = SSN, Name, City and is neither SSN nor All Attributes

Want X in $X \rightarrow B$ s.t.: $X = X^+$ or X^+ = [all attributes]

Example BCNF Decomposition

Name	**SSN**	**City**
Fred	123-45-6789	Seattle
Joe	987-65-4321	Westfield

$$\overset{X}{\text{SSN}} \rightarrow \overbrace{\text{Name, City}}^{B}$$



SSN	**PhoneNumber**
123-45-6789	206-555-1234
123-45-6789	206-555-6543
987-65-4321	908-555-2121
987-65-4321	908-555-1234

Let's check anomalies:

- Redundancy ?
- Update ?
- Delete ?

Find X in X→B s.t.: X ≠ X^+ and X^+ ≠ [all attributes]

Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

SSN → name, age

age → hairColor

Find X in $X \rightarrow B$ s.t.: $X \neq X^+$ and $X^+ \neq$ [all attributes]

Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

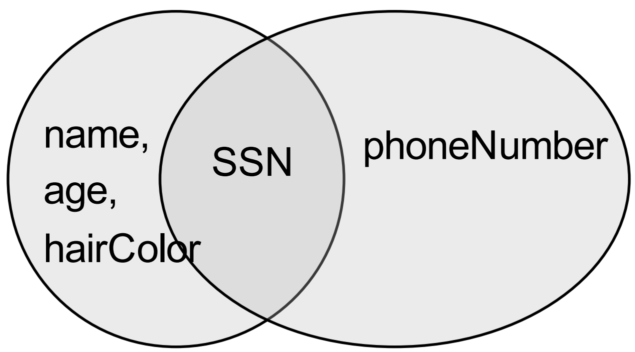
SSN → name, age

age → hairColor

Iteration 1: Person: SSN+ = SSN, name, age, hairColor

Decompose into: P(SSN, name, age, hairColor)

Phone(SSN, phoneNumber)



Find X in X→B s.t.: X $\neq X^+$ and $X^+ \neq$ [all attributes]

Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

SSN → name, age

age → hairColor

What are the keys ?

Iteration 1: Person: SSN+ = SSN, name, age, hairColor

Decompose into: P(SSN, name, age, hairColor)
Phone(SSN, phoneNumber)

Iteration 2: P: age+ = age, hairColor

Decompose: People(SSN, name, age)
Hair(age, hairColor)
Phone(SSN, phoneNumber)

Find X in X→B s.t.: X ≠X^+ and X^+ ≠ [all attributes]

Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

SSN → name, age

age → hairColor

Note the keys!

Iteration 1: Person: SSN+ = SSN, name, age, hairColor

Decompose into: P(<u>SSN</u>, name, age, hairColor)
Phone(<u>SSN, phoneNumber</u>)

Iteration 2: P: age+ = age, hairColor

Decompose: People(<u>SSN</u>, name, age)
Hair(<u>age</u>, hairColor)
Phone(<u>SSN, phoneNumber</u>)

R(A,B,C,D)

$A \rightarrow B$
$B \rightarrow C$

Example: BCNF

R(A,B,C,D)

R(A,B,C,D)

A → B
B → C

Example: BCNF

Recall: find X s.t.
$X \neq X^+ \neq$ [all-attrs]

R(A,B,C,D)

R(A,B,C,D)

$A \rightarrow B$
$B \rightarrow C$

Example: BCNF

Recall: find X s.t.
$X \neq X^+ \neq$ [all-attrs]

R(A,B,C,D)
$A^+ = ABC \neq ABCD$

R(A,B,C,D)

Example: BCNF

$A \rightarrow B$
$B \rightarrow C$

Recall: find X s.t.
$X \neq X^+ \neq$ [all-attrs]

R(A,B,C,D)
$A^+ = ABC \neq ABCD$

$R_1(A,B,C)$

$R_2(A,D)$

R(A,B,C,D)

Example: BCNF

$A \rightarrow B$
$B \rightarrow C$

Recall: find X s.t.
$X \neq X^+ \neq$ [all-attrs]

R(A,B,C,D)
$A^+ = ABC \neq ABCD$

$R_1(A,B,C)$
$B^+ = BC \neq ABC$

$R_2(A,D)$

R(A,B,C,D)

Example: BCNF

A → B
B → C

Recall: find X s.t.
$X \neq X^+ \neq$ [all-attrs]

R(A,B,C,D)
$A^+ = ABC \neq ABCD$

$R_1(A,B,C)$
$B^+ = BC \neq ABC$

$R_2(A,D)$

$R_{11}(B,C)$

$R_{12}(A,B)$

What are the keys ?

What happens if in R we first pick B^+ ? Or AB^+ ?

Decompositions in General

$$R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_p)$$

$$S_1(A_1, \ldots, A_n, B_1, \ldots, B_m) \qquad S_2(A_1, \ldots, A_n, C_1, \ldots, C_p)$$

S_1 = projection of R on $A_1, \ldots, A_n, B_1, \ldots, B_m$
S_2 = projection of R on $A_1, \ldots, A_n, C_1, \ldots, C_p$

Lossless Decomposition

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Name	Price
Gizmo	19.99
OneClick	24.99
~~Gizmo~~	~~19.99~~

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Lossy Decomposition

What is lossy here?

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Price	Category
19.99	Gadget
24.99	Camera
19.99	Camera

Lossy Decomposition

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Price	Category
19.99	Gadget
24.99	Camera
19.99	Camera

Decomposition in General

$R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_p)$

$S_1(A_1, \ldots, A_n, B_1, \ldots, B_m)$ $S_2(A_1, \ldots, A_n, C_1, \ldots, C_p)$

Let: S_1 = projection of R on $A_1, \ldots, A_n, B_1, \ldots, B_m$
S_2 = projection of R on $A_1, \ldots, A_n, C_1, \ldots, C_p$

The decomposition is called *lossless* if $R = S_1 \bowtie S_2$

Fact: If $A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$ then the decomposition is lossless

It follows that every BCNF decomposition is lossless

Schema Refinements = Normal Forms

- 1st Normal Form = all tables are flat
- 2nd Normal Form = obsolete
- Boyce Codd Normal Form = no bad FDs
- 3rd and 4th Normal Form = see book
  - BCNF is lossless but can cause loss of ability to check some FDs (see book 3.4.4)
  - 3NF fixes that (is lossless and dependency-preserving), but some tables might not be in BCNF – i.e., they may have redundancy anomalies