CSE 344

AUGUST 3RD

NORMALIZATION

ADMINISTRIVIA

- **WQ6 due Monday**
  - DB design
- **HW7 due next Wednesday**
  - DB design
  - normalization

DATABASE DESIGN PROCESS

Conceptual Model:

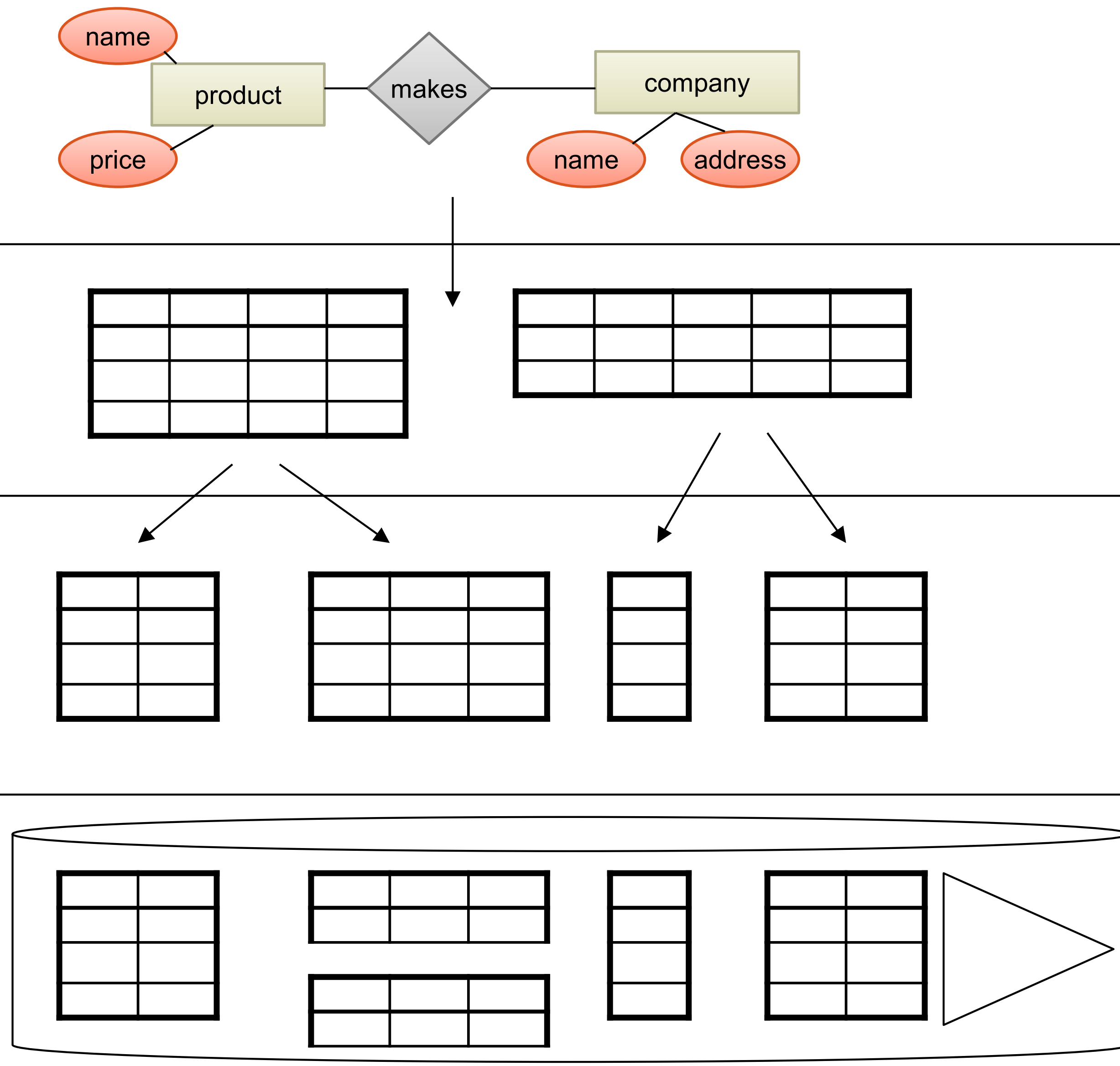
Relational Model:
Tables + constraints
And also functional dep.

Normalization:
Eliminates anomalies

Conceptual Schema

Physical storage details

Physical Schema



RELATIONAL SCHEMA DESIGN

Name	StudentID	PhoneNumber	City
Fred	123456789	206-555-1234	Seattle
Fred	123456789	206-555-6543	Seattle
Joe	987654321	908-555-2121	Westfield

One person may have multiple phones, but lives in only one city

Primary key is thus (StudentID, PhoneNumber)

What is the problem with this schema?

RELATIONAL SCHEMA DESIGN

Name	StudentID	PhoneNumber	City
Fred	123456789	206-555-1234	Seattle
Fred	123456789	206-555-6543	Seattle
Joe	987654321	908-555-2121	Westfield

Anomalies:

- Redundancy = repeat data
- Update anomalies = what if Fred moves to “Bellevue”?
- Deletion anomalies = what if Joe deletes his phone number?

RELATION DECOMPOSITION

Break the relation into two:

Name	StudentID	PhoneNumber	City
Fred	123456789	206-555-1234	Seattle
Fred	123456789	206-555-6543	Seattle
Joe	987654321	908-555-2121	Westfield

Name	StudentID	City
Fred	123456789	Seattle
Joe	987654321	Westfield

StudentID	PhoneNumber
123456789	206-555-1234
123456789	206-555-6543
987654321	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 ?)

RELATIONAL SCHEMA DESIGN
(OR LOGICAL DESIGN)

How do we do this systematically?

Start with some relational schema

Find out its *functional dependencies* (FDs)

Use FDs to *normalize* the relational schema

FUNCTIONAL DEPENDENCIES (FDS)

Definition

If two tuples agree on the attributes

$A_1, A_2, \ldots, A_n$

then they must also agree on the attributes

$B_1, B_2, \ldots, B_m$

Formally:

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

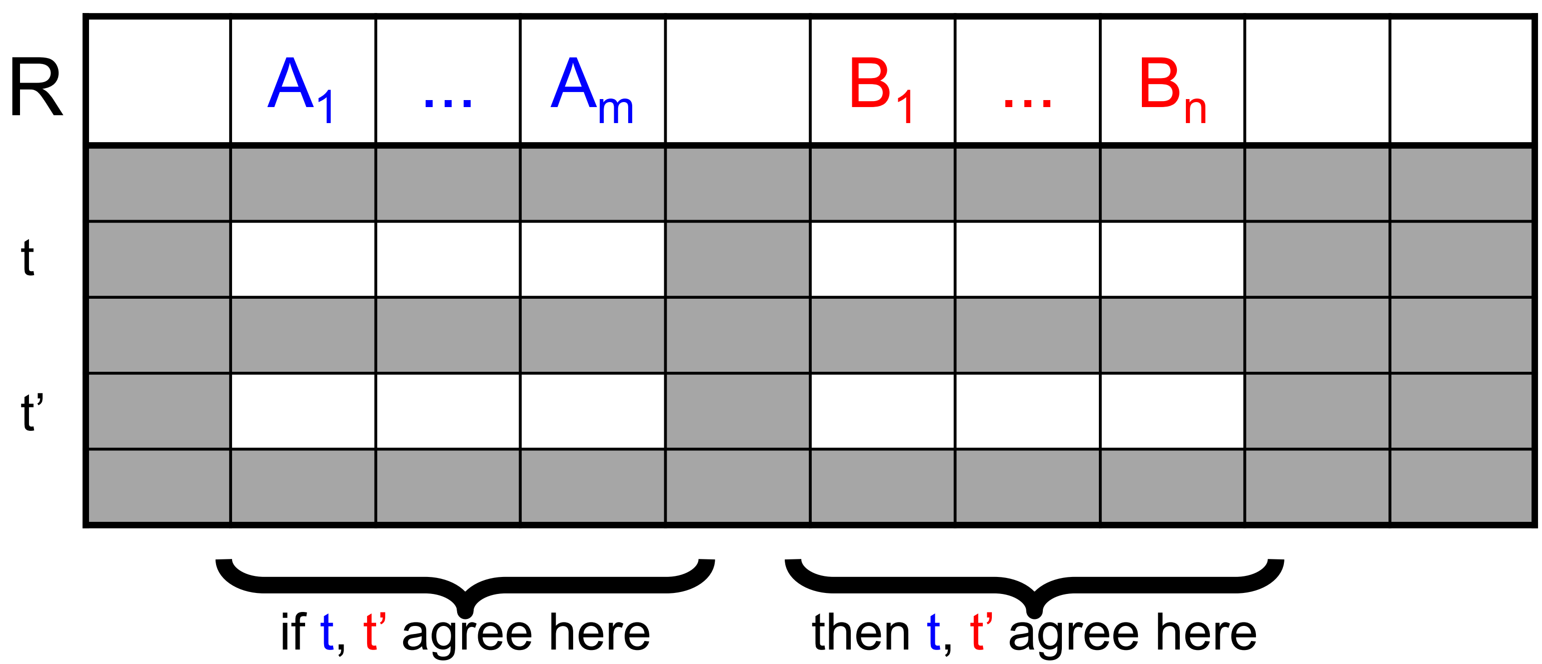
$A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m$

FUNCTIONAL DEPENDENCIES (FDS)

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

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

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



FUNCTIONAL DEPENDENCIES (FDS)

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

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

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

Logically equivalent:

$$\neg \exists\ t, t' \in R,$$

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

EXAMPLE

An FD <u>holds</u>, or <u>does not hold</u> on an instance:

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

EmpID → Name, Phone, Position

Position → Phone

but not Phone → Position

EXAMPLE

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876 ←	Salesrep
E1111	Smith	9876 ←	Salesrep
E9999	Mary	1234	Lawyer

Position → Phone

EXAMPLE

EmpID	Name	Phone	Position
E0045	Smith	1234 →	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234 →	Lawyer

But not Phone → Position

EXAMPLE

name → color
category → department
color, category → price

name	category	color	department	price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Green	Toys	99

Do all the FDs hold on this instance?

EXAMPLE

name → color
category → department
color, category → price

name	category	color	department	price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Green	Toys	49
Gizmo	Stationary	Green	Office-supp.	59

What about this one ?

BUZZWORDS

FD holds or does not hold on an instance

If we can be sure that *every instance of R* will be one in which a given FD is true, then we say that R satisfies the FD

If we say that R satisfies an FD, we are stating a constraint on R

WHY BOTHER WITH FDS?

Name	StudentID	PhoneNumber	City
Fred	123456789	206-555-1234	Seattle
Fred	123456789	206-555-6543	Seattle
Joe	987654321	908-555-2121	Westfield

FD: StudentID -> Name, City

Anomalies:

- Redundancy = repeat data
- Update anomalies = what if Fred moves to “Bellevue”?
- Deletion anomalies = what if Joe deletes his phone number?

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.

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:

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

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

$color^+ = \{color\}$

CLOSURE ALGORITHM

X={A1, …, An}.

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: name, category → 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 the key of R?

PRACTICE AT HOME

Find all FD's implied by:

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

PRACTICE AT HOME

Find all FD's implied by:

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

Step 1: Compute X^+, for every X:

$A^+ = A$, $B^+ = BD$, $C^+ = C$, $D^+ = D$

$AB^+ = ABCD$, $AC^+ = AC$, $AD^+ = ABCD$,
$BC^+ = BCD$, $BD^+ = BD$, $CD^+ = CD$

$ABC^+ = ABD^+ = ACD^+ = ABCD$ (no need to compute– why ?)

$BCD^+ = BCD$, $ABCD^+ = ABCD$

Step 2: Enumerate all FD's $X \rightarrow Y$, s.t. $X^+ = X \cup Y$ (with $X \cap Y = \varnothing$):

$AB \rightarrow CD$, $AD \rightarrow BC$, $ABC \rightarrow D$, $ABD \rightarrow C$, $ACD \rightarrow B$

KEYS

A superkey is a set of attributes $A_1, \ldots, A_n$ s.t. for any other attribute B, we have $A_1, \ldots, A_n \rightarrow B$

- I.e., for which closure = all attributes

A key is a minimal superkey

- A superkey and for which no subset is a superkey
- (A key already determines everything, so any superset of key does too.)

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 → price
category → color

What is the key ?

EXAMPLE

Product(name, price, category, color)

name, category → price
category → color

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

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, either $X^+ = X$ or $X^+ =$ [all attributes]

BCNF DECOMPOSITION ALGORITHM

Normalize(R)

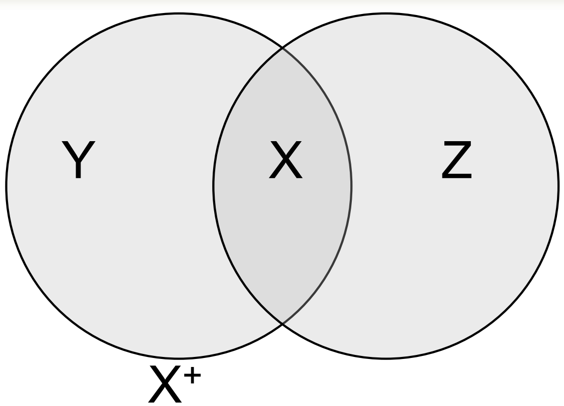
find X s.t.: $X \neq X^+$ and $X^+ \neq$ [all attributes]

<u>if</u> (not found) **<u>then</u>** "R is in BCNF"

<u>let</u> $Y = X^+ - X$; $Z =$ [all attributes] $- X^+$

decompose R into R1($X \cup Y$) and R2($X \cup Z$)

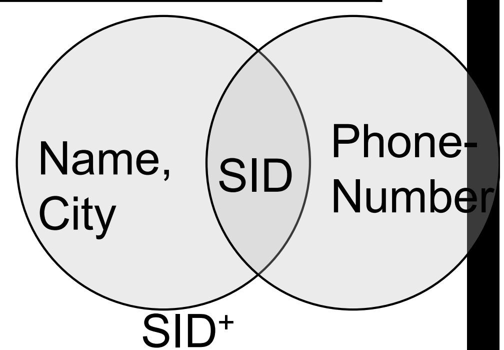
Normalize(R1); Normalize(R2);



EXAMPLE

Name	StudentID	PhoneNumber	City
Fred	123456789	206-555-1234	Seattle
Fred	123456789	206-555-6543	Seattle
Joe	987654321	908-555-2121	Westfield
Joe	987654321	908-555-1234	Westfield

StudentID → Name, City



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

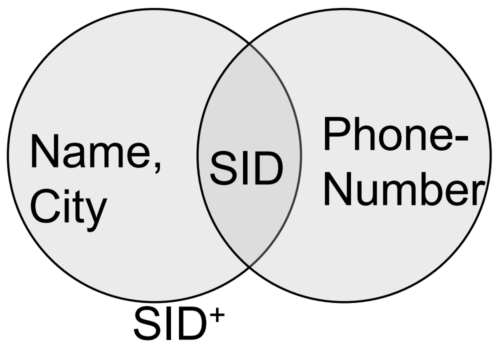
In other words:
StudentID+ = StudentID, Name, City and is neither StudentID nor All Attributes

EXAMPLE BCNF DECOMPOSITION

Name	**StudentID**	**City**
Fred	123456789	Seattle
Joe	987654321	Westfield

StudentID → Name, City

StudentID	**PhoneNumber**
123456789	206-555-1234
123456789	206-555-6543
987654321	908-555-2121
987654321	908-555-1234



Let's check anomalies:

- Redundancy ?
- Update ?
- Delete ?

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

EXAMPLE BCNF DECOMPOSITION

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

SSN → name, age

age → hairColor

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

EXAMPLE BCNF DECOMPOSITION

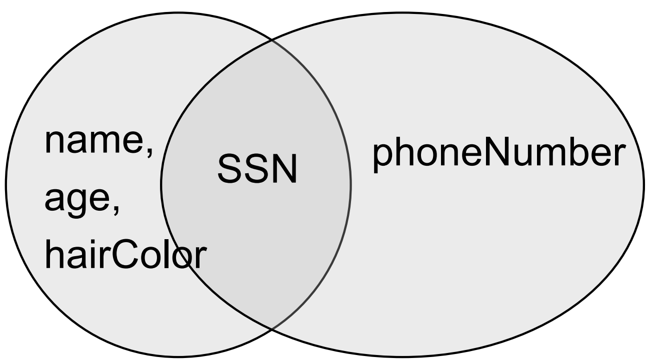
Person(name, SSN, age, hairColor, phoneNumber)

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 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 s.t.: X $\neq X^+$ and $X^+ \neq$ [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)

EXAMPLE: BCNF

A → B
B → C

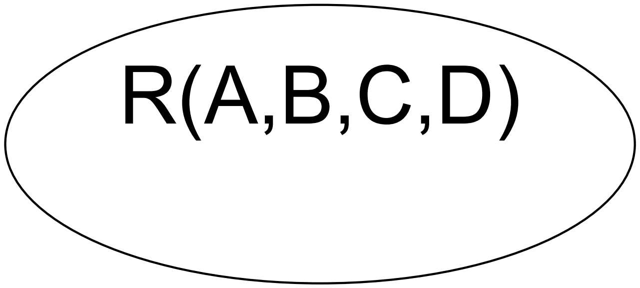
R(A,B,C,D)

R(A,B,C,D)

EXAMPLE: BCNF

A → B
B → C

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



R(A,B,C,D)

EXAMPLE: BCNF

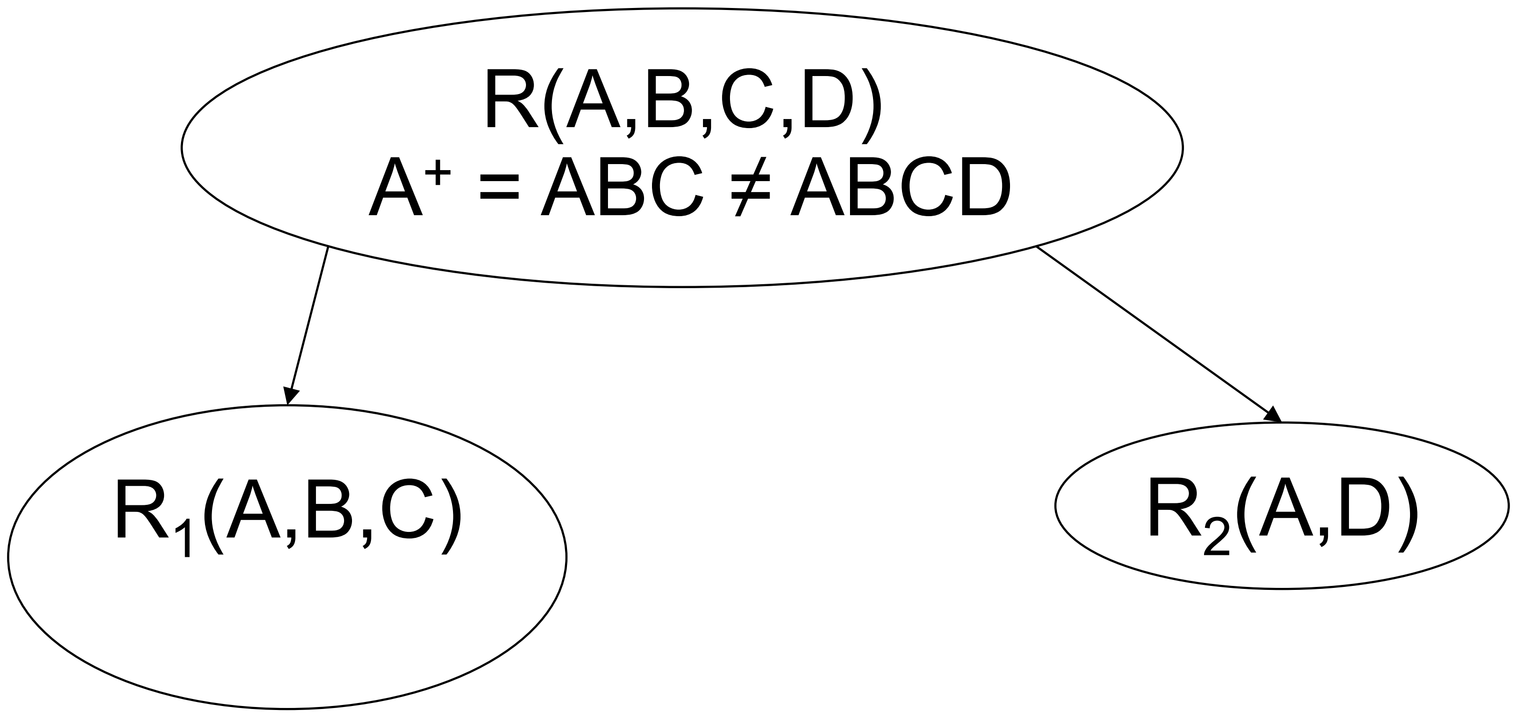
A → B
B → C

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

R(A,B,C,D)

EXAMPLE: BCNF

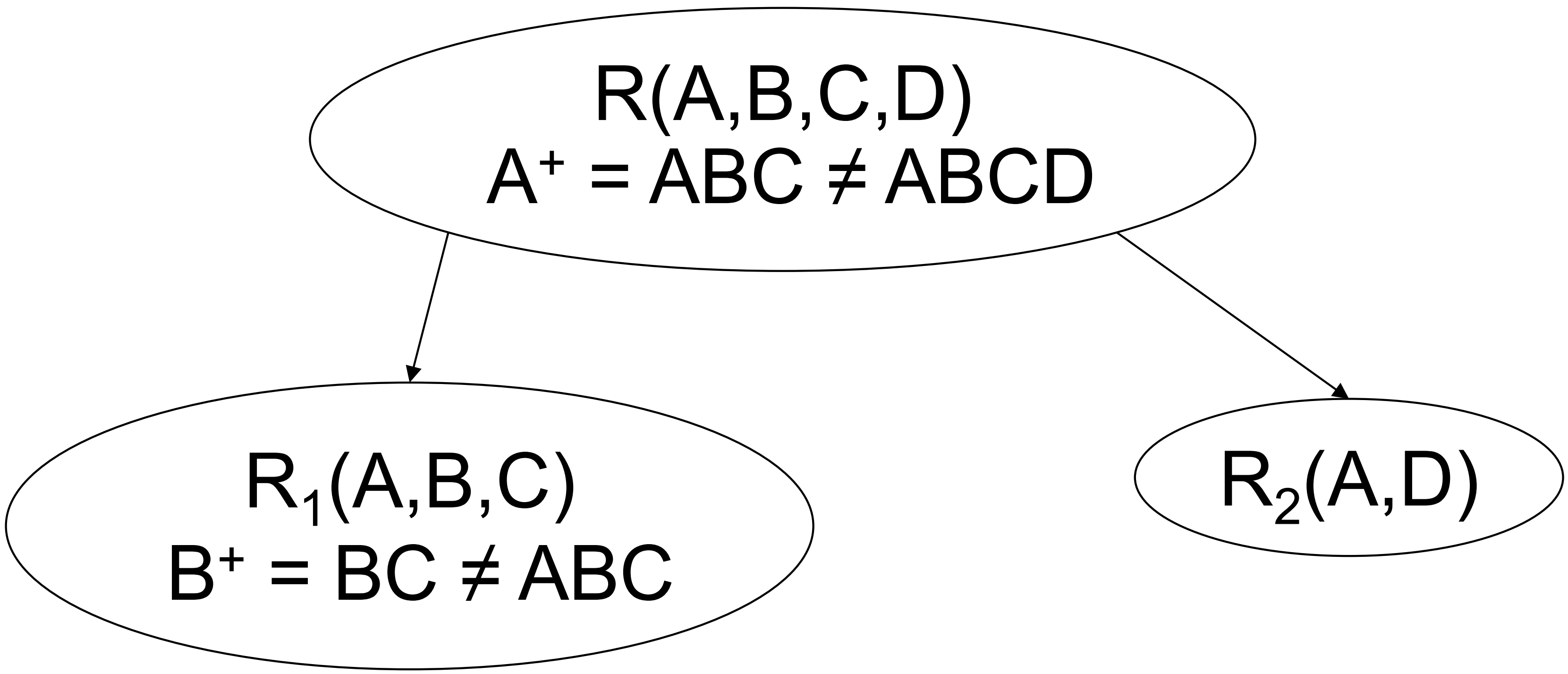
$A \rightarrow B$
$B \rightarrow C$



R(A,B,C,D)

EXAMPLE: BCNF

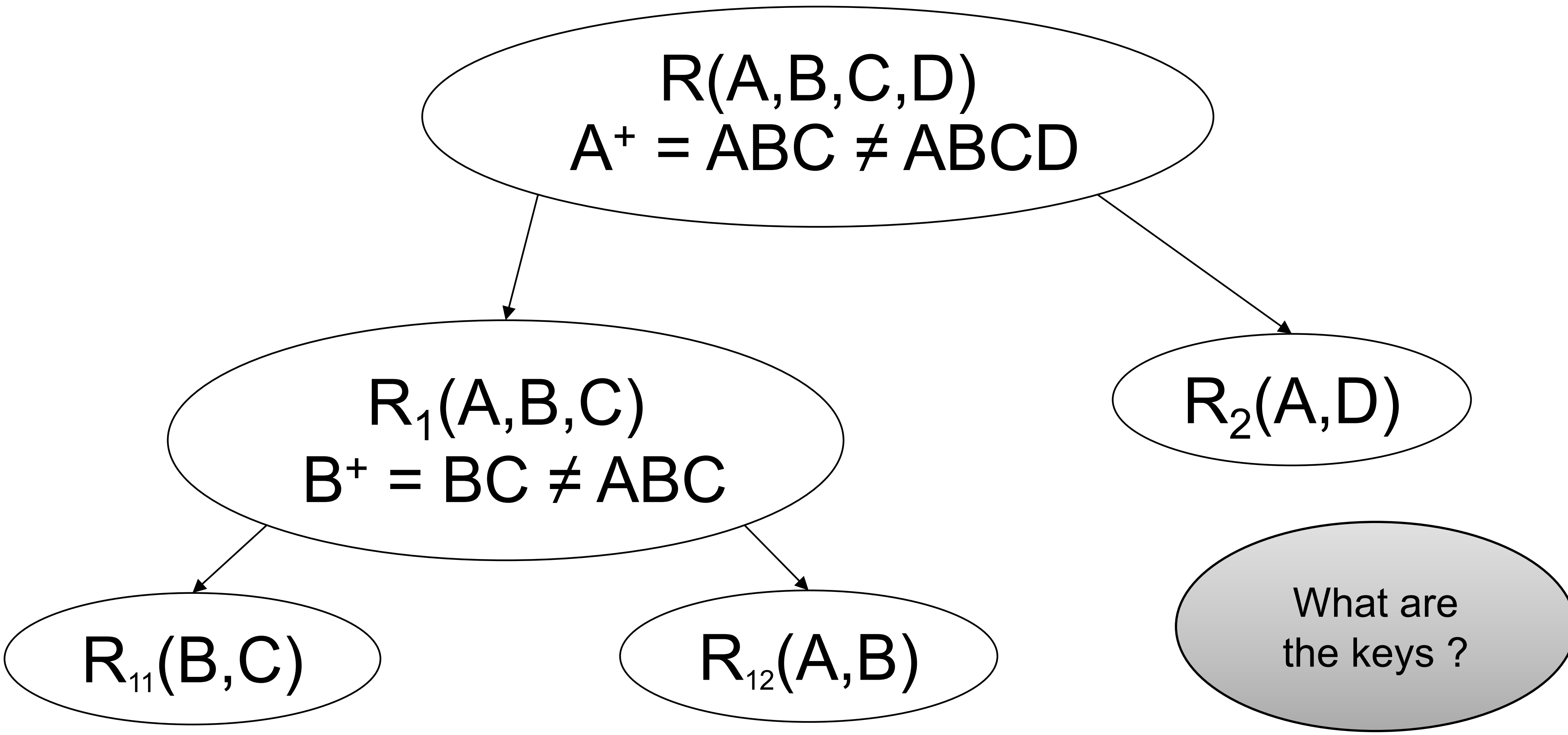
$A \rightarrow B$
$B \rightarrow C$



R(A,B,C,D)

EXAMPLE: BCNF

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



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