Introduction to Management
CSE 344

Lectures 17: Design Theory

Announcements

- No class/office hour on Monday
- Midterm on Wednesday (Feb 19) in class
- HW5 due next Thursday (Feb 20)
- No WQ next week (WQ6 due on Feb 25)
  - But try to finish early! (Lecture 15-17)

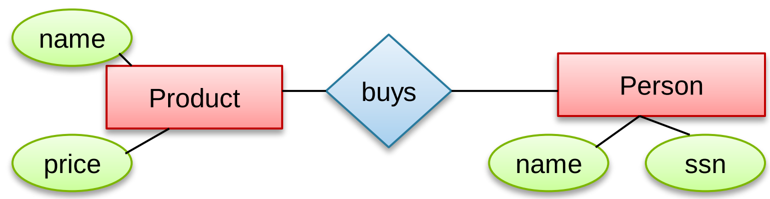
Midterm

- **All material up to and including Lecture 15**
  – SQL, basic evaluation + indexes, RA, datalog-with-negation, RC, XML/XPath/XQuery, E/R diagram

- **Open books, open notes, no electronic devices**
  – Don’t waste paper printing stuff. Normally, you shouldn’t need any notes during the exam. My suggestion is to print, say, 5-6 selected slides from the lecture notes that you had trouble with, and to print your own homework, just in case you forget some cool solution you used there.
  – Make sure you understand all the concepts!

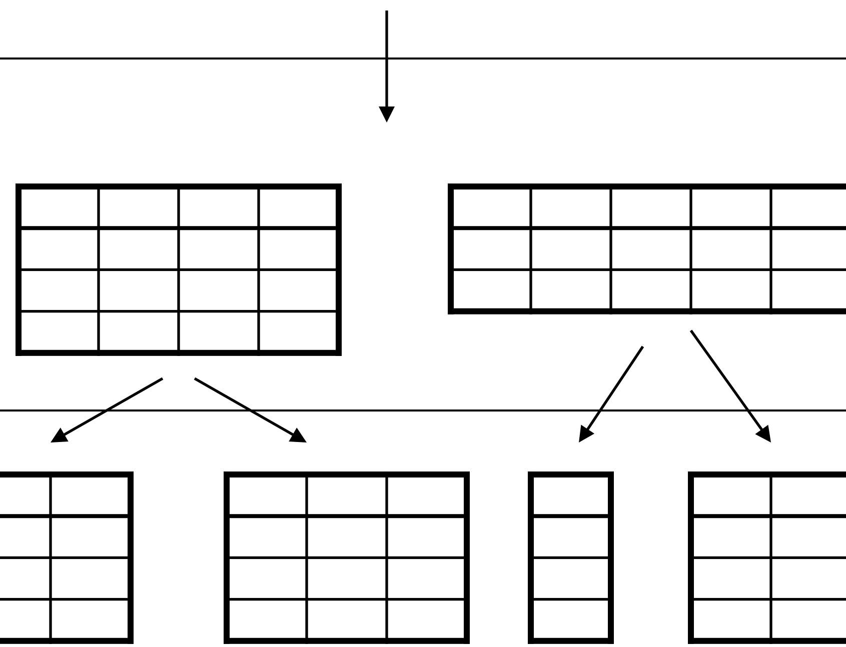
Relational Schema Design

Conceptual Model:



DONE

Relational Model:
plus FD's



DONE

Normalization:
Eliminates ***anomalies***

Today

Relational Schema Design

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

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

Primary key is thus (SSN, PhoneNumber)

What is the problem with this schema?

Relational Schema Design

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

Anomalies:

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

Can you suggest a solution?

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 ?)

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

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

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

R		A_1	...	A_m		B_1	...	B_n		
t										
t'										

if t, t' agree here then t, t' agree here

Example

An FD holds, or does not hold 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	99

Do all the FDs hold on this instance?

No. color,category -> price does not hold

Example

name → color
category → department
color, category → price

name	category	color	department	price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Black	Toys	99
Gizmo	Stationary	Green	Office-supp.	59

What about this one ?

Example

name → color
category → department
color, category → price

name	category	color	department	price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Black	Toys	99
Gizmo	Stationary	Green	Office-supp.	59

What about this one ?

Yes

Terminology

- 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 F, we are **stating a constraint on R**

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 $A_1, \ldots, A_n$ (under FD set F)

The **closure**, $\{A_1, \ldots, A_n\}^+$ = the set of attributes B that
(any reln that satisfies F) satisfies $A_1, \ldots, A_n \rightarrow B$

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={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,…..}

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	→	C
A, D	→	E
B	→	D
A, F	→	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 }

Can you see a “key” of R?

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:

Step 2: Enumerate all FD's $X \rightarrow Y$, s.t. $Y \subseteq X^+$ and $X \cap Y = \emptyset$:

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

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. $Y \subseteq X^+$ and $X \cap Y = \emptyset$:

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

Keys

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

- 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 only the minimal X's to get the keys
  - No subset of X is a superkey

Example

Product(name, price, category, color)

name, category → price
category → 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 keys

Try in class..

.....

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 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

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

What is the key?

Suggest a rule for decomposing the table to eliminate anomalies

Eliminating Anomalies

Main idea:

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

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:

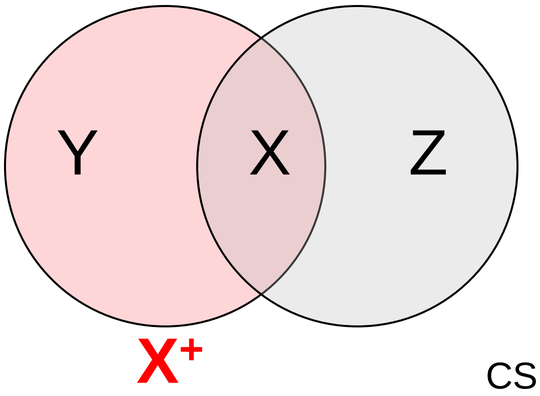
Definition. A relation R is in BCNF if:

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

Trivial FD

BCNF Decomposition Algorithm

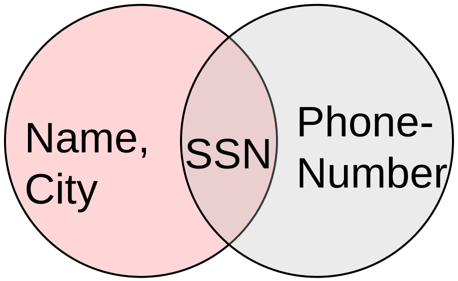
Normalize(R)
  find X s.t.: $X \neq X^+ \neq$ [all attributes]
  <u>if</u> (not found) **<u>then</u>** "R is in BCNF"
  <u>let</u> $Y = X^+ - X$; $Z =$ [all attributes] $- X^+$
  decompose R into $R1(X \cup Y)$ and $R2(X \cup Z)$
  Normalize(R1); Normalize(R2);



Example

Name	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



SSN+

The only key is: {SSN, PhoneNumber}

Hence SSN → Name, City is a “bad” dependency

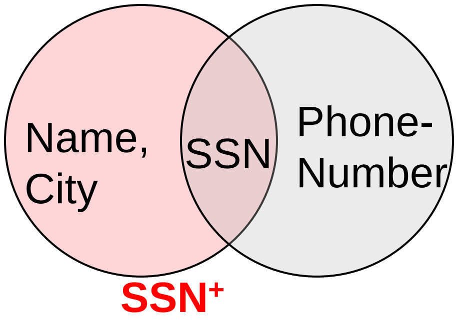
In other words:

SSN+ = Name, City and is neither SSN nor All Attributes

Example BCNF Decomposition

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

SSN → Name, City



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 Fred’s city?
- Delete Joe’s all ph?

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

Example BCNF Decomposition

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

SSN → name, age

age → hairColor

Find X s.t.: X ≠ X^+ ≠ [all attributes]

Example BCNF Decomposition

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

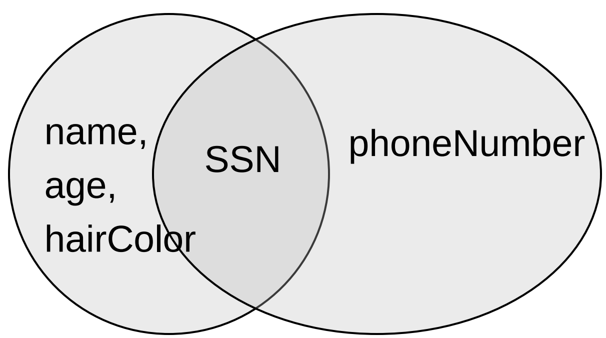
SSN → name, age

age → hairColor

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

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

Phone(SSN, phoneNumber, attr)



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

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

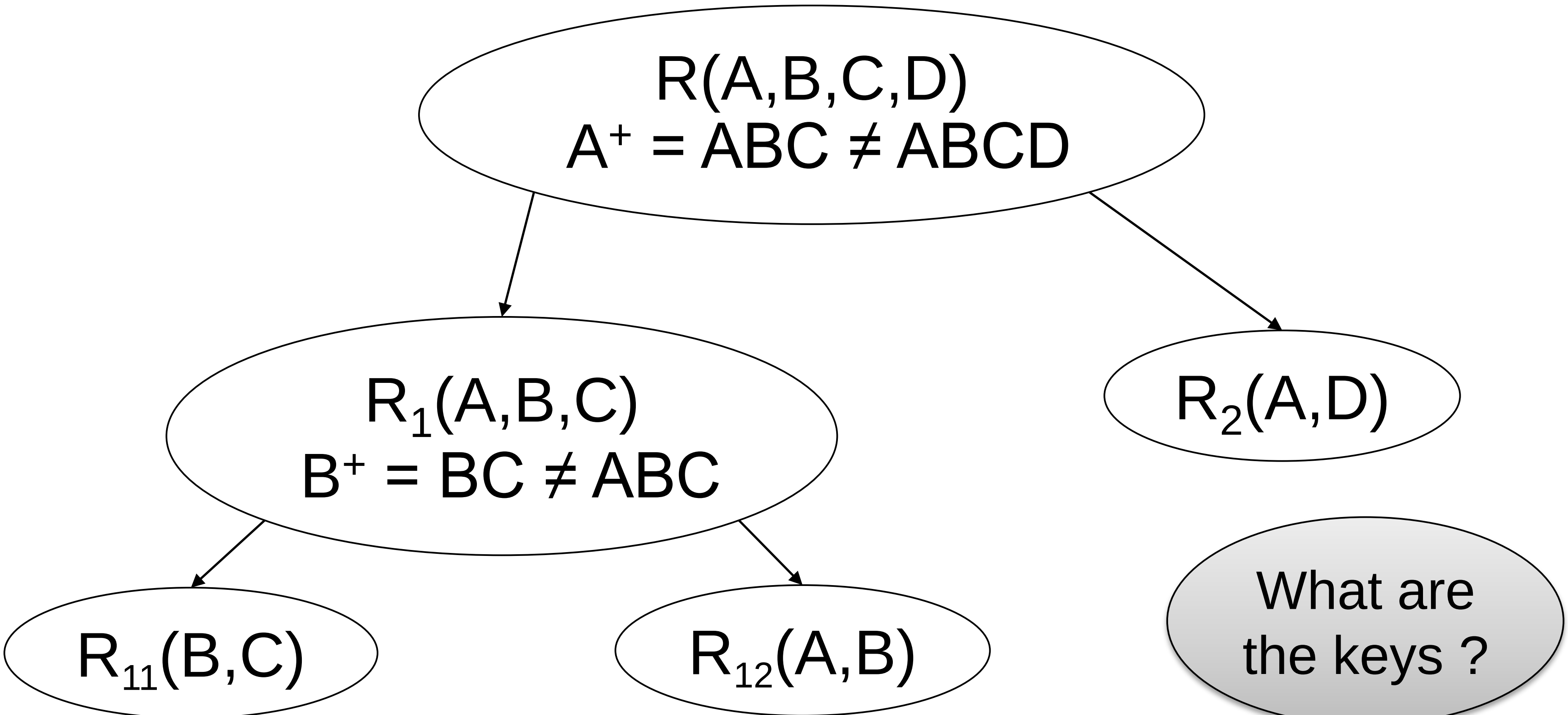
Practice at Home

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

R(A,B,C,D)

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

Practice at Home



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