Introduction to Management CSE 344

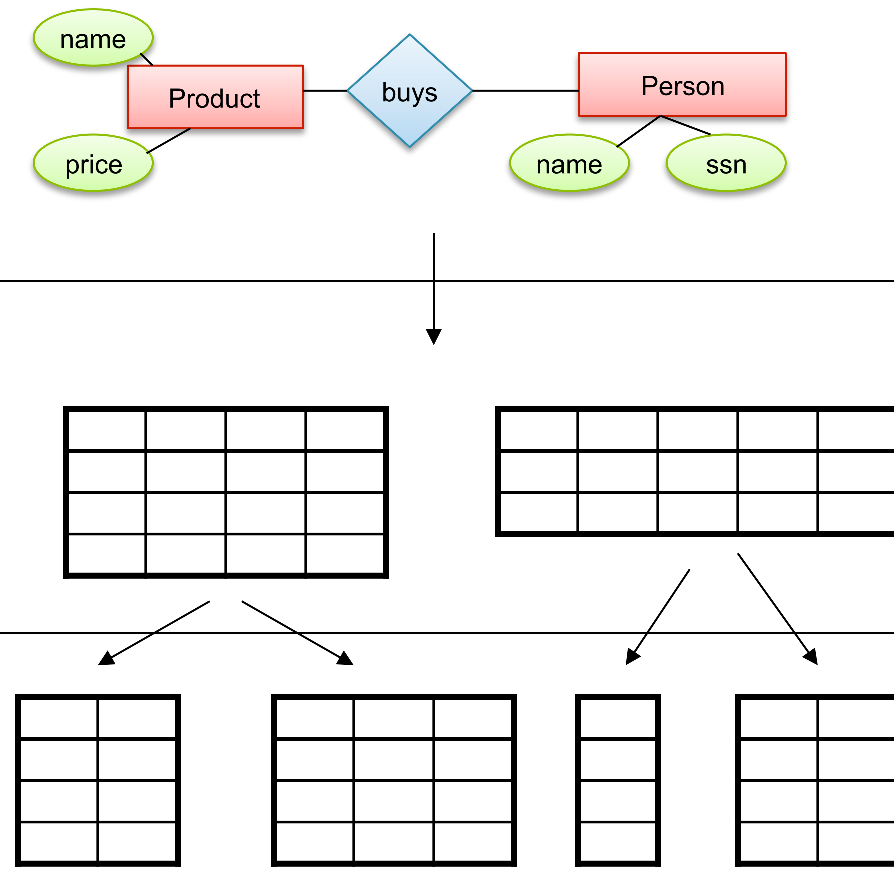
Lectures 16: Database Design

Relational Schema Design

Conceptual Model:

Relational Model:
plus FD's

Normalization:
Eliminates ***anomalies***



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?

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	<u>SSN</u>	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Westfield

<u>SSN</u>	<u>PhoneNumber</u>
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

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

then they must also agree on the attributes

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

Formally:

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

Functional Dependencies (FDs)

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

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

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

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 $\rightarrow$ Name, Phone, Position

Position $\rightarrow$ Phone

but not Phone $\rightarrow$ 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	Black	Toys	99

Does this instance satisfy all the FDs ?

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 ?

An Interesting Observation

If all these FDs are true:

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

Then this FD also holds:

name, category $\rightarrow$ price

Why ??

Goal: Find ALL Functional Dependencies

- Anomalies occur when certain “bad” FDs hold
- We know some of the FDs
- Need to find *all* FDs
- Then look for the bad ones

Armstrong's Rules (1/3)

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

Is equivalent to

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

Splitting rule and Combing rule

	A_1	...	A_m		B_1	...	B_m	

Armstrong's Rules (2/3)

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

Trivial Rule

where i = 1, 2, ..., n

	A_1	…	A_m	

Why ?

Armstrong's Rules (3/3)

Transitive Rule

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

and $\boxed{B_1, B_2, \ldots, B_m \rightarrow C_1, C_2, \ldots, C_p}$

then $\boxed{A_1, A_2, \ldots, A_n \rightarrow C_1, C_2, \ldots, C_p}$

Why ?

Armstrong's Rules (3/3)

Illustration

	A_1	…	A_m		B_1	…	B_m		C_1	…	C_p	

Example (continued)

Start from the following FDs:

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

Infer the following FDs:

Inferred FD	Which Rule did we apply ?
4. name, category → name	
5. name, category → color	
6. name, category → category	
7. name, category → color, category	
8. name, category → price	

Example (continued)

Answers:

1. name $\rightarrow$ color
2. category $\rightarrow$ department
3. color, category $\rightarrow$ price

Inferred FD	Which Rule did we apply ?
4. name, category $\rightarrow$ name	Trivial rule
5. name, category $\rightarrow$ color	Transitivity on 4, 1
6. name, category $\rightarrow$ category	Trivial rule
7. name, category $\rightarrow$ color, category	Split/combine on 5, 6
8. name, category $\rightarrow$ price	Transitivity on 3, 7

THIS IS TOO HARD ! Let's see an easier way.

Closure of a set of Attributes

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

The **closure**, $\{A_1, \ldots, A_n\}^+$ = the set of attributes B
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 $\{A,B\}^+$ $\quad$ X = {A, B, C, D, E }

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

Why Do We Need Closure

- With closure we can find all FD's easily

- To check if $X \rightarrow A$
  - Compute X^+
  - Check if $A \in X^+$

Practice at Home

Find all FD's implied by:

$$\begin{aligned} A, B &\rightarrow C \\ A, D &\rightarrow B \\ B &\rightarrow D \end{aligned}$$

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
  – I.e. set of attributes which is a superkey and for which no subset is a superkey

Computing (Super)Keys

- Compute X^+ for all sets X
- If X^+ = all attributes, then X is a superkey
- List only the minimal X's to get the keys

Example

Product(name, price, category, color)

name, category → price
category → color

What is the key ?

Example

Product(name, price, category, color)

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

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

$\boxed{\text{SSN} \rightarrow \text{Name, City}}$

What is the key?

{SSN, PhoneNumber}

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

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.

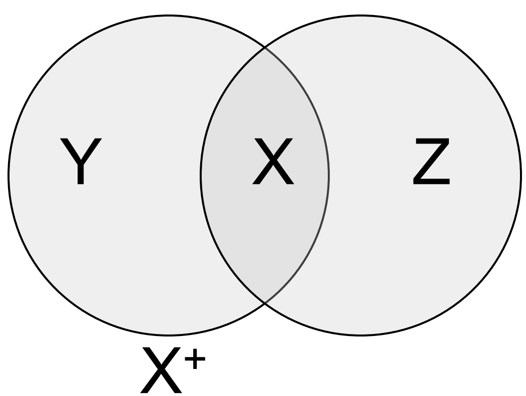
Equivalently:

Definition. A relation R is in BCNF if:

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

BCNF Decomposition Algorithm

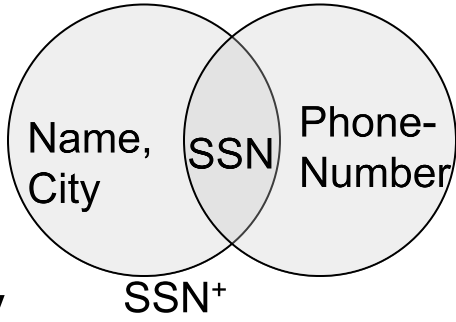
Normalize(R)
  find X s.t.: X $\neq$ X^+ $\neq$ [all attributes]
  if (not found) **then** "R is in BCNF"
  let $Y = X^+ - X$; $Z = \text{[all attributes]} - X^+$
  decompose R into R1($X \cup Y$) and R2($X \cup Z$)
  Normalize(R1); Normalize(R2);



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



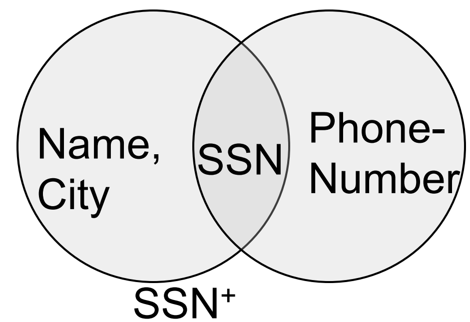
The only key is: {SSN, PhoneNumber}
Hence SSN → Name, City is a “bad” dependency

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 ?
- Delete ?

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

Example BCNF Decomposition

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

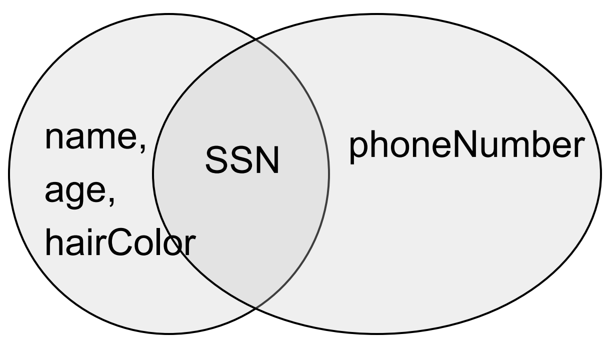
SSN → name, age

age → hairColor

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

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

Phone(SSN, phoneNumber)



Find X s.t.: $X \neq X^{+} \neq$ [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(SSN, phoneNumber)

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)

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

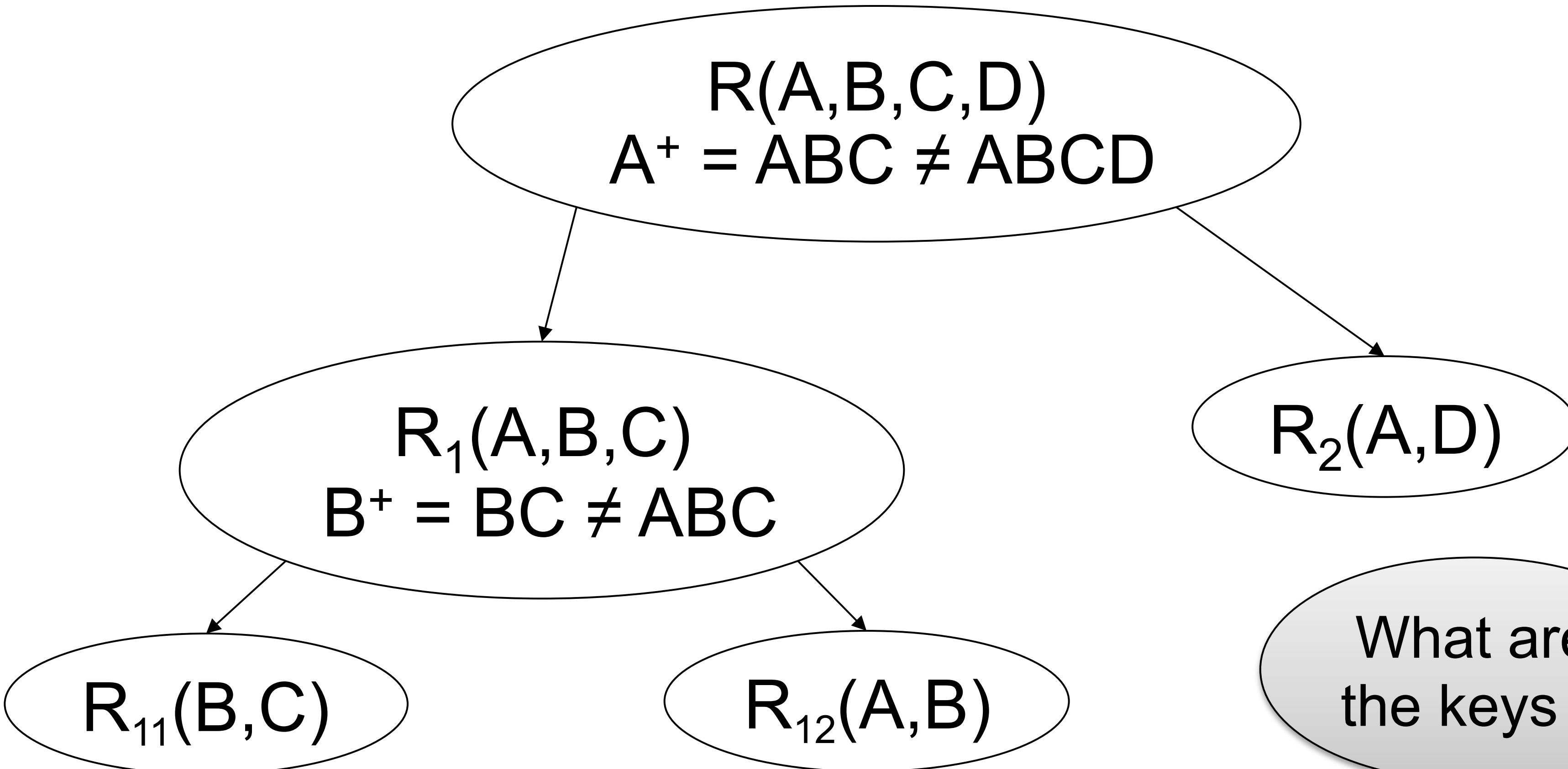
Practice at Home

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

R(A,B,C,D)

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

Practice at Home



What are the keys ?

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

Schema Refinements = Normal Forms

- 1st Normal Form = all tables are flat
- 2nd Normal Form = obsolete
- Boyce Codd Normal Form = today
- 3rd Normal Form = see book