CSE 344

AUGUST 6TH

LOSS AND VIEWS

ADMINISTRIVIA

- WQ6 due tonight
- HW7 due Wednesday

DATABASE DESIGN PROCESS

Conceptual Model:

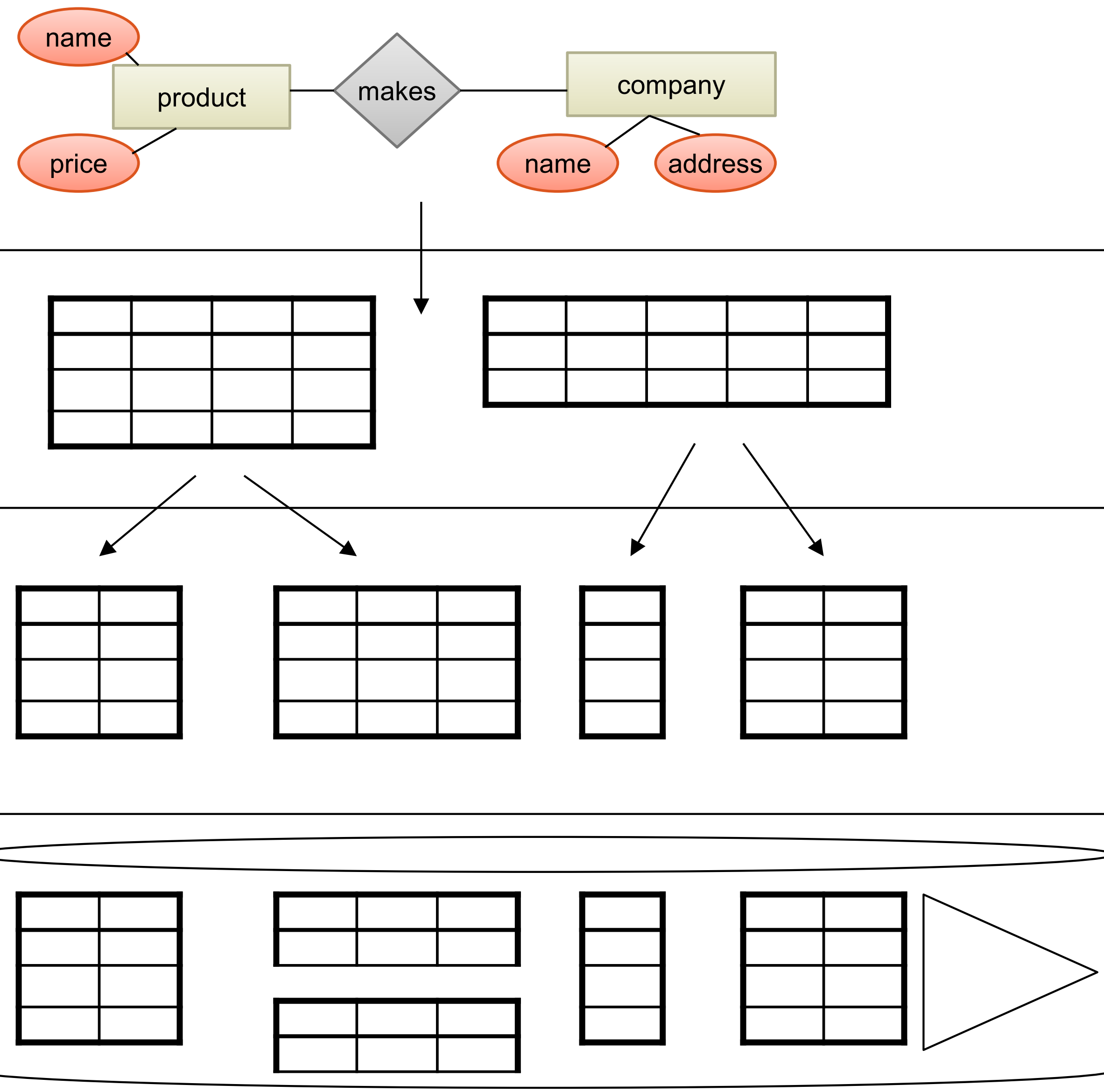
Relational Model:
Tables + constraints
And also functional dep.

Normalization:
Eliminates anomalies

Conceptual Schema

Physical storage details

Physical Schema



ELIMINATING ANOMALIES

Main idea:

X → A is OK if X is a (super)key

X → A is bad 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)

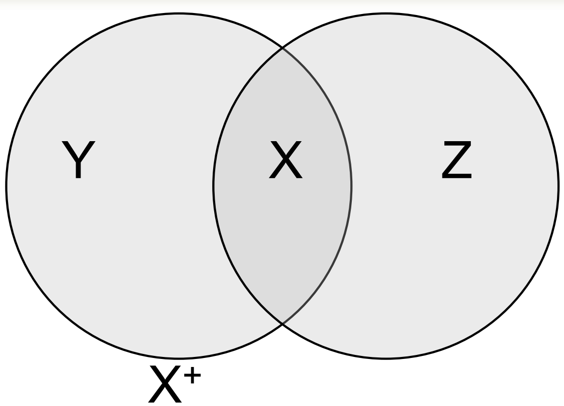
find X s.t.: $X \neq X^+$ and $X^+ \neq$ [all attributes]

if (not found) **then** "R is in BCNF"

let $Y = X^+ - X$; $Z =$ [all attributes] $- X^+$

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

Normalize(R1); Normalize(R2);



R(A,B,C,D)

EXAMPLE: BCNF

A → B
B → C

R(A,B,C,D)

R(A,B,C,D)

EXAMPLE: BCNF

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

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

R(A,B,C,D)

R(A,B,C,D)

EXAMPLE: BCNF

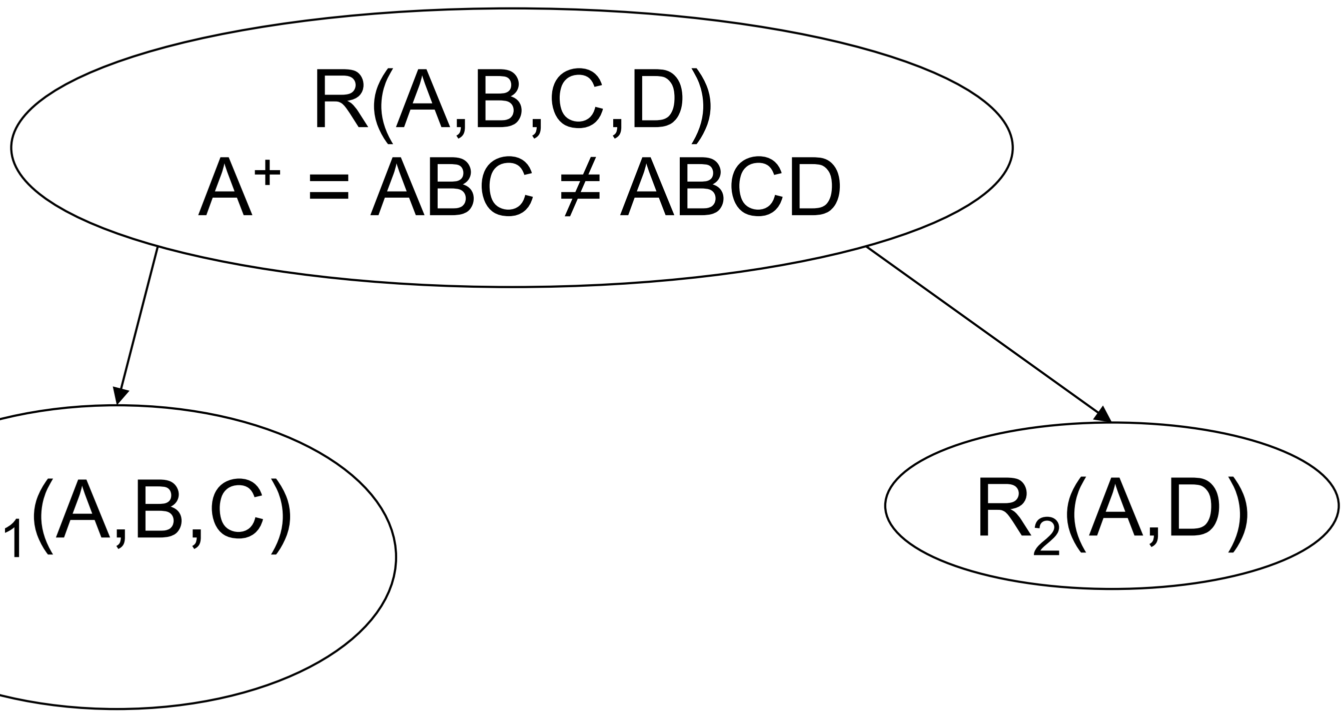
A → B
B → C

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

R(A,B,C,D)

EXAMPLE: BCNF

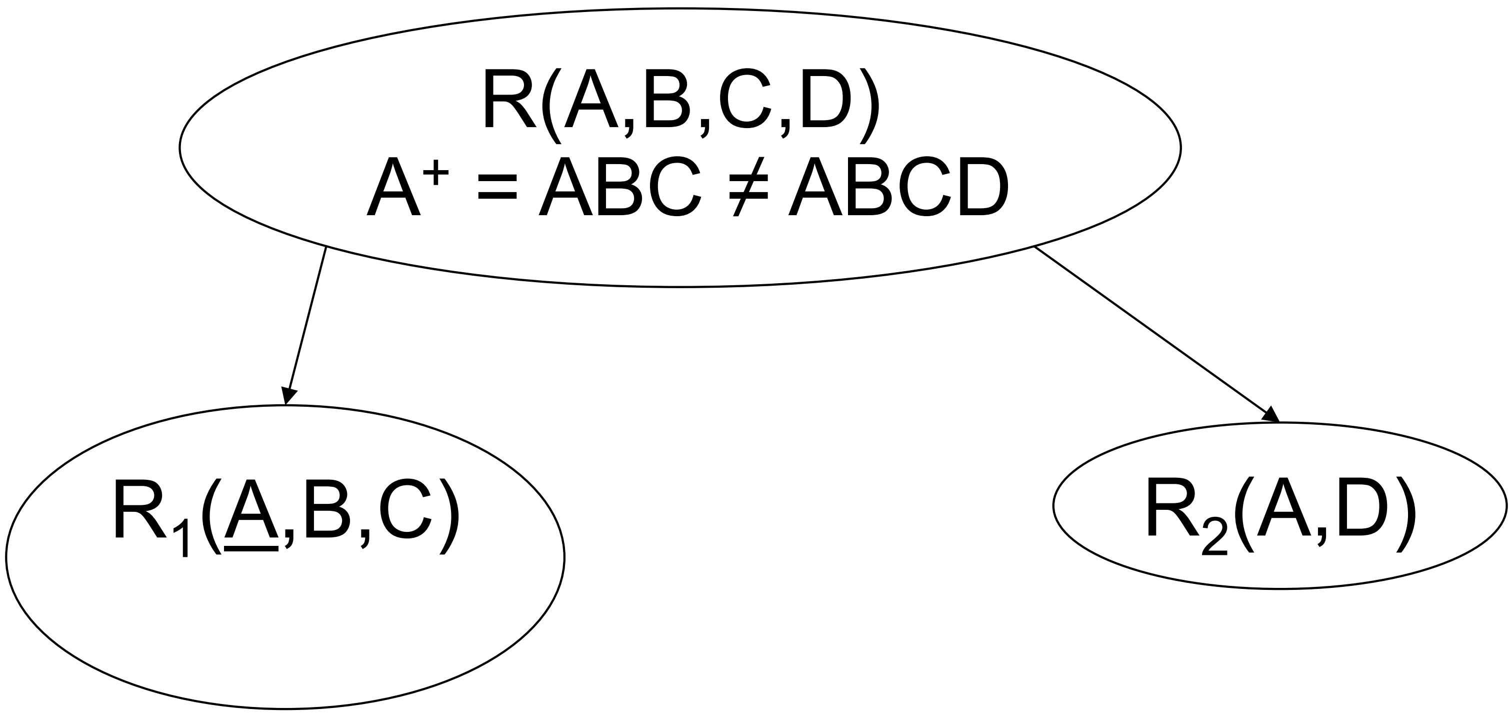
$A \rightarrow B$
$B \rightarrow C$



R(A,B,C,D)

EXAMPLE: BCNF

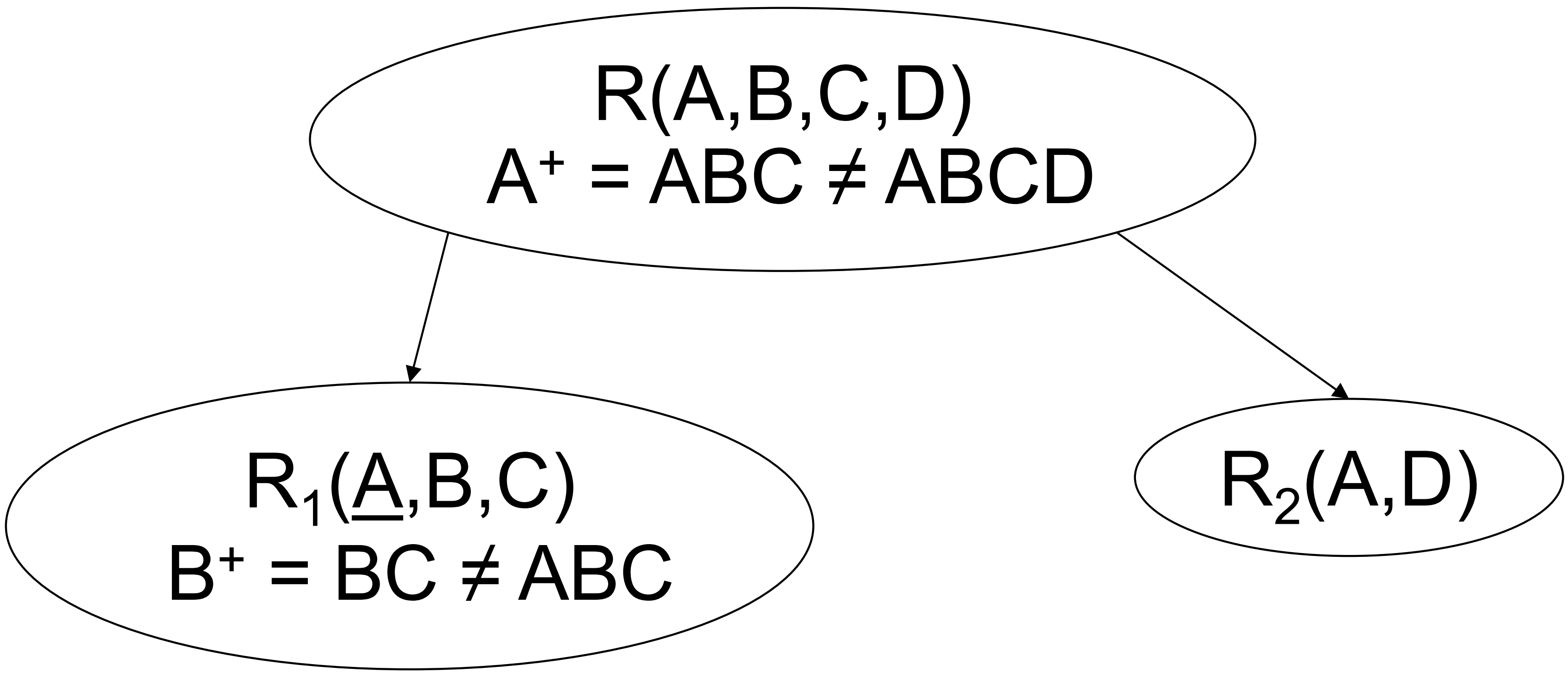
$A \to B$
$B \to C$



R(A,B,C,D)

EXAMPLE: BCNF

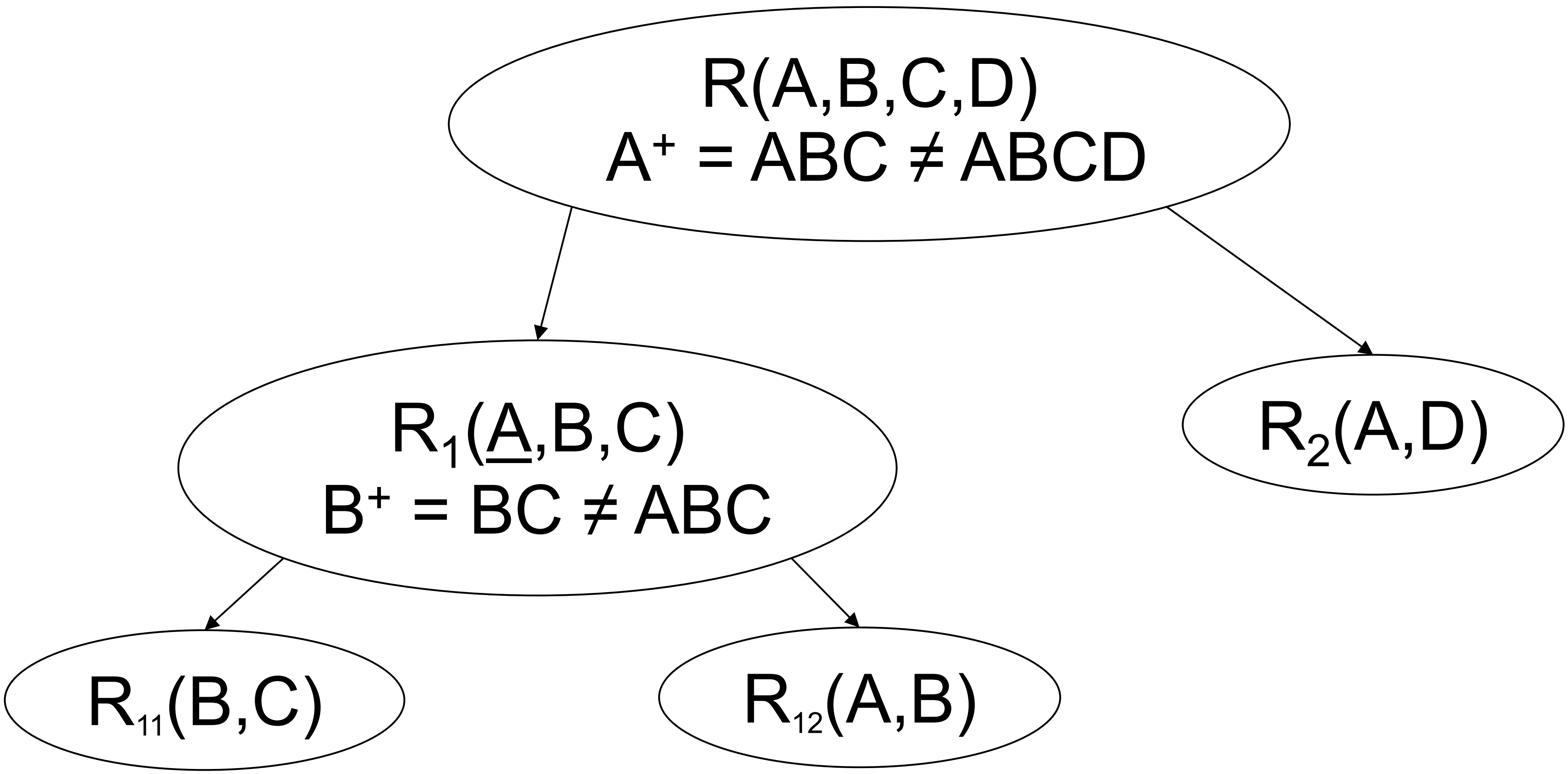
A → B
B → C



R(A,B,C,D)

EXAMPLE: BCNF

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

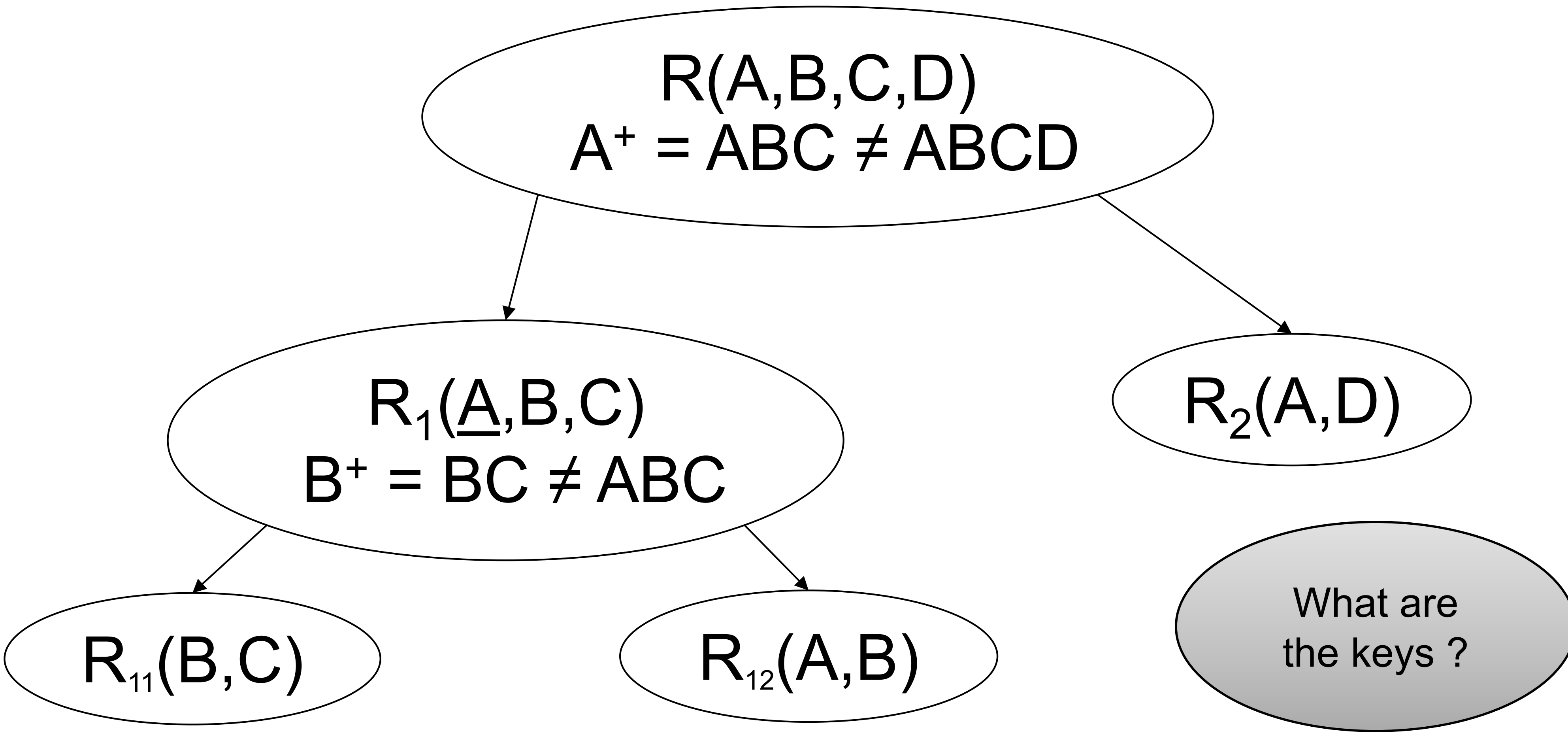


What happens if in R we first pick B^+ ? Or AB^+ ?]

R(A,B,C,D)

EXAMPLE: BCNF

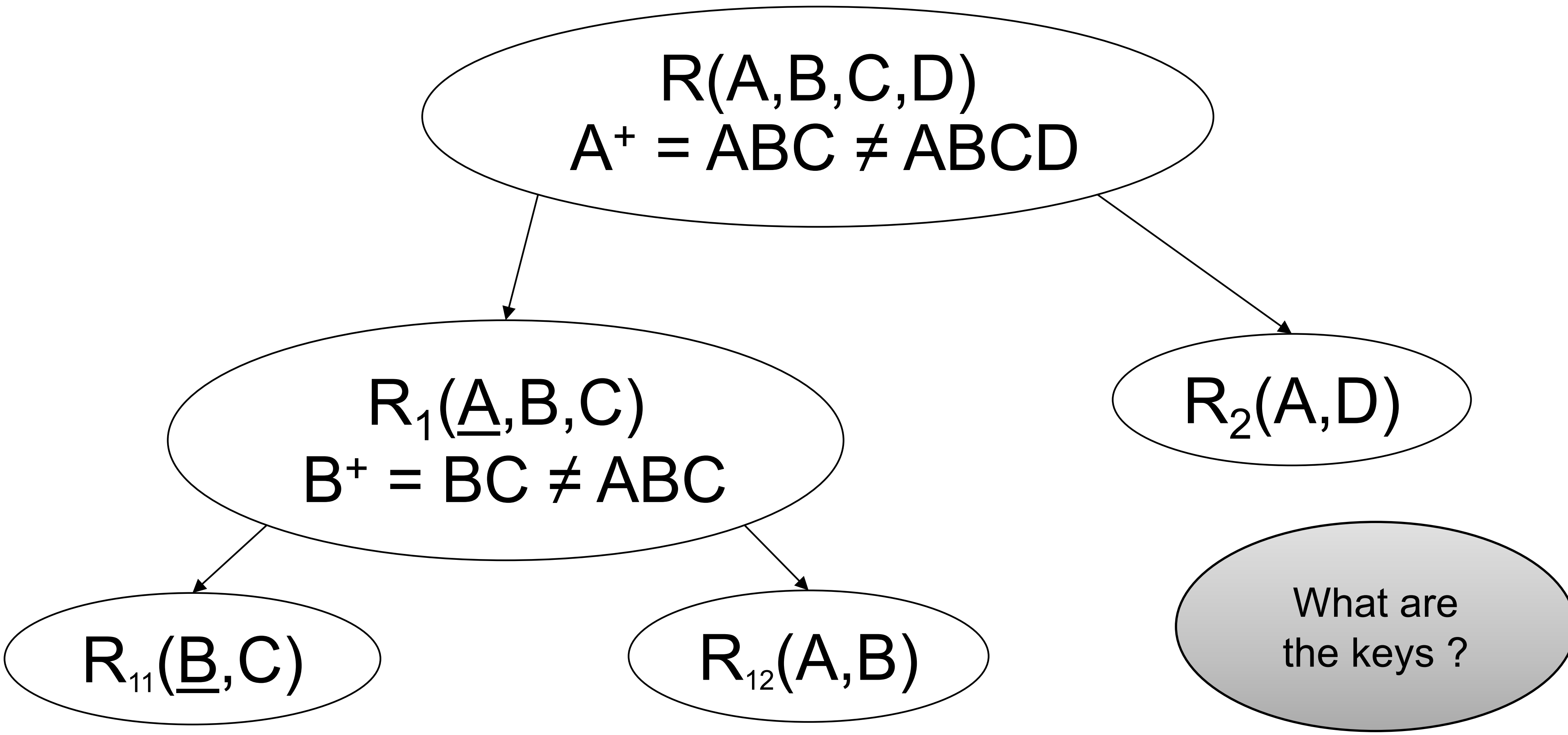
A → B
B → C



R(A,B,C,D)

EXAMPLE: BCNF

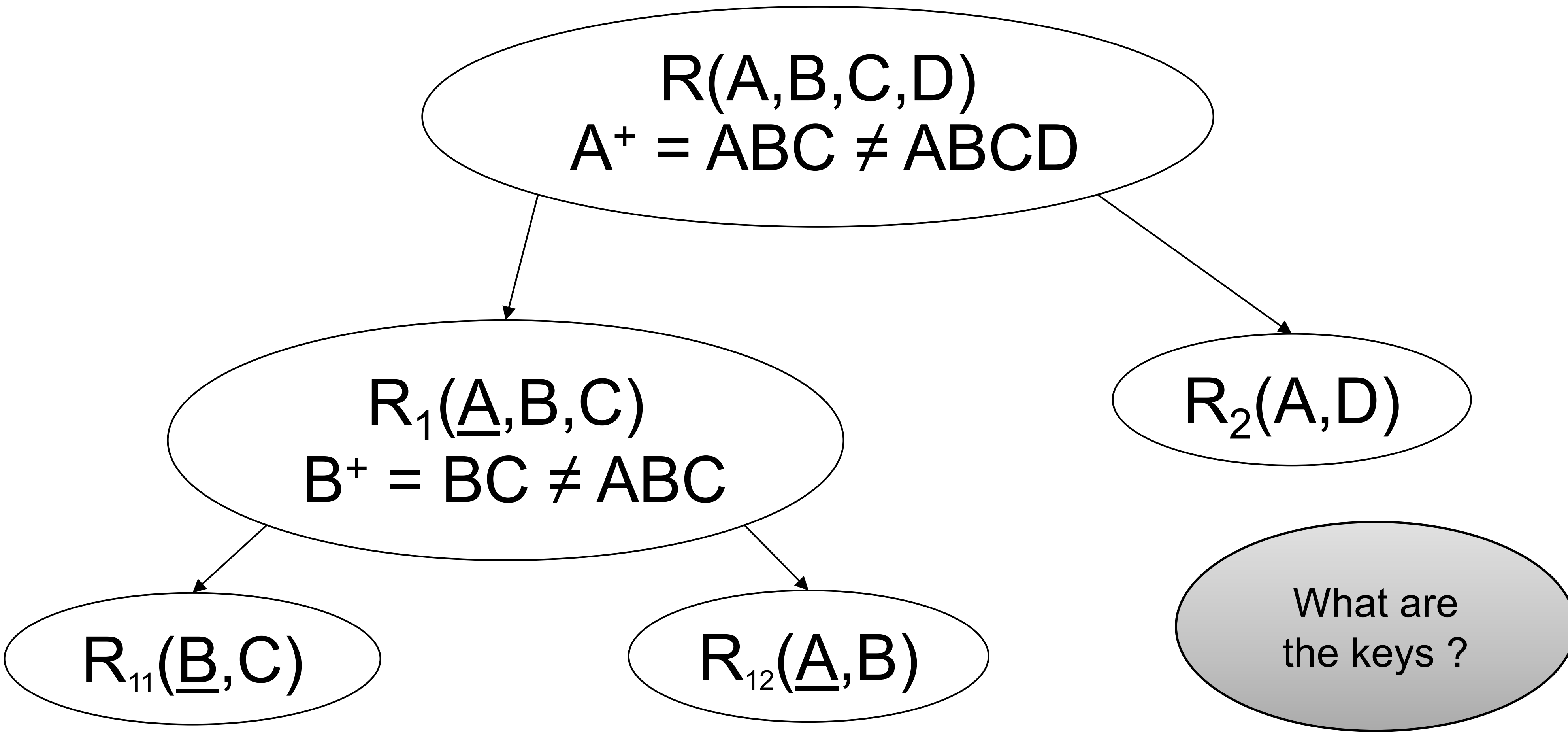
A → B
B → C



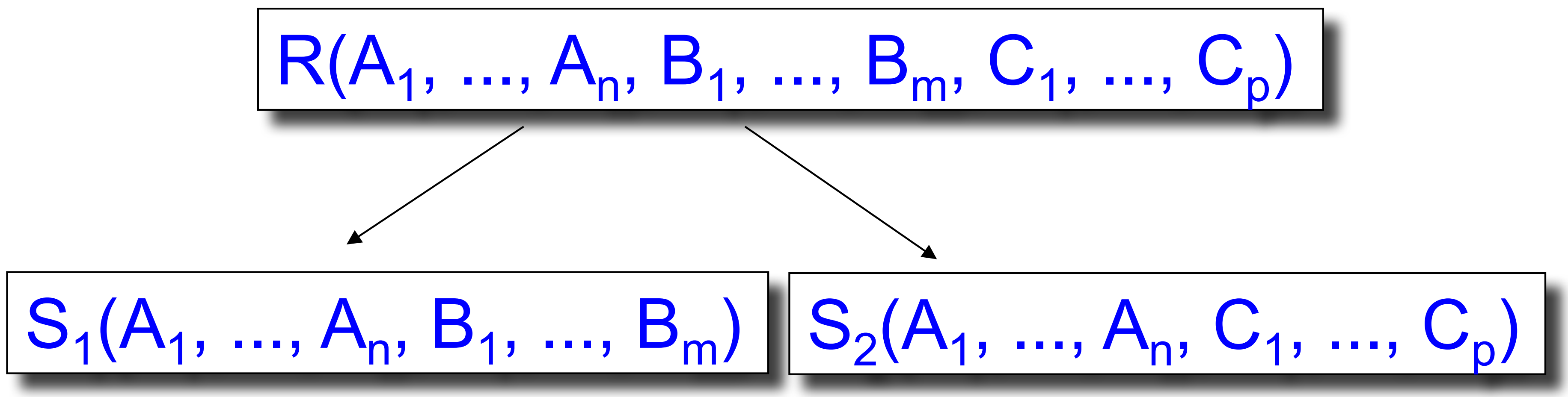
R(A,B,C,D)

EXAMPLE: BCNF

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



DECOMPOSITIONS IN GENERAL



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

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

and R is a subset of $S_1 \times S_2$

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

IS THIS LOSSLESS?

If we decompose R into $\Pi_{S1}(R)$, $\Pi_{S2}(R)$, $\Pi_{S3}(R)$, …
Is it true that $S1 \bowtie S2 \bowtie S3 \bowtie \ldots = R$?

That is true if we can show that:

$R \subseteq S1 \bowtie S2 \bowtie S3 \bowtie \ldots$ always holds (why?)

$R \supseteq S1 \bowtie S2 \bowtie S3 \bowtie \ldots$ neet to check

Example from textbook Ch. 3.4.2

THE CHASE TEST FOR LOSSLESS JOIN

$R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D)$
R satisfies: $A \rightarrow B$, $B \rightarrow C$, $CD \rightarrow A$

$S1 = \Pi_{AD}(R)$, $S2 = \Pi_{AC}(R)$, $S3 = \Pi_{BCD}(R)$,
hence $R \subseteq S1 \bowtie S2 \bowtie S3$
Need to check: $R \supseteq S1 \bowtie S2 \bowtie S3$

Example from textbook Ch. 3.4.2

THE CHASE TEST FOR LOSSLESS JOIN

R(A,B,C,D) = S1(A,D) ⋈ S2(A,C) ⋈ S3(B,C,D)
R satisfies: A→B, B→C, CD→A

S1 = Π_{AD}(R), S2 = Π_{AC}(R), S3 = Π_{BCD}(R),
hence R ⊆ S1 ⋈ S2 ⋈ S3
Need to check: R ⊇ S1 ⋈ S2 ⋈ S3
Suppose (a,b,c,d) ∈ S1 ⋈ S2 ⋈ S3 Is it also in R?

Example from textbook Ch. 3.4.2

THE CHASE TEST FOR LOSSLESS JOIN

R(A,B,C,D) = S1(A,D) ⋈ S2(A,C) ⋈ S3(B,C,D)
R satisfies: A→B, B→C, CD→A

$S1 = \Pi_{AD}(R)$, $S2 = \Pi_{AC}(R)$, $S3 = \Pi_{BCD}(R)$,
hence $R \subseteq S1 \bowtie S2 \bowtie S3$
Need to check: $R \supseteq S1 \bowtie S2 \bowtie S3$
Suppose $(a,b,c,d) \in S1 \bowtie S2 \bowtie S3$ Is it also in R?
R must contain the following tuples:

A	B	C	D	
a	b1	c1	d	$(a,d) \in S1 = \Pi_{AD}(R)$

Why ?

Example from textbook Ch. 3.4.2

THE CHASE TEST FOR LOSSLESS JOIN

R(A,B,C,D) = S1(A,D) ⋈ S2(A,C) ⋈ S3(B,C,D)
R satisfies: A→B, B→C, CD→A

$S1 = \Pi_{AD}(R)$, $S2 = \Pi_{AC}(R)$, $S3 = \Pi_{BCD}(R)$,
hence $R \subseteq S1 \bowtie S2 \bowtie S3$
Need to check: $R \supseteq S1 \bowtie S2 \bowtie S3$
Suppose $(a,b,c,d) \in S1 \bowtie S2 \bowtie S3$ Is it also in R?
R must contain the following tuples:

A	B	C	D	Why ?
a	b1	c1	d	$(a,d) \in S1 = \Pi_{AD}(R)$
a	b2	c	d2	$(a,c) \in S2 = \Pi_{BD}(R)$

Example from textbook Ch. 3.4.2

THE CHASE TEST FOR LOSSLESS JOIN

R(A,B,C,D) = S1(A,D) ⋈ S2(A,C) ⋈ S3(B,C,D)
R satisfies: A→B, B→C, CD→A

$S1 = \Pi_{AD}(R)$, $S2 = \Pi_{AC}(R)$, $S3 = \Pi_{BCD}(R)$,
hence $R \subseteq S1 \bowtie S2 \bowtie S3$
Need to check: $R \supseteq S1 \bowtie S2 \bowtie S3$
Suppose $(a,b,c,d) \in S1 \bowtie S2 \bowtie S3$ Is it also in R?
R must contain the following tuples:

A	B	C	D	Why ?
a	b1	c1	d	$(a,d) \in S1 = \Pi_{AD}(R)$
a	b2	c	d2	$(a,c) \in S2 = \Pi_{BD}(R)$
a3	b	c	d	$(b,c,d) \in S3 = \Pi_{BCD}(R)$

Example from textbook Ch. 3.4.2

THE CHASE TEST FOR LOSSLESS JOIN

$R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D)$
R satisfies: $A \rightarrow B$, $B \rightarrow C$, $CD \rightarrow A$

$S1 = \Pi_{AD}(R)$, $S2 = \Pi_{AC}(R)$, $S3 = \Pi_{BCD}(R)$,
hence $R \subseteq S1 \bowtie S2 \bowtie S3$
Need to check: $R \supseteq S1 \bowtie S2 \bowtie S3$
Suppose $(a,b,c,d) \in S1 \bowtie S2 \bowtie S3$ Is it also in R?
R must contain the following tuples:

A	B	C	D	Why ?
a	b1	c1	d	$(a,d) \in S1 = \Pi_{AD}(R)$
a	b2	c	d2	$(a,c) \in S2 = \Pi_{BD}(R)$
a3	b	c	d	$(b,c,d) \in S3 = \Pi_{BCD}(R)$

“Chase” them (apply FDs):

$A \rightarrow B$

A	B	C	D
a	b1	c1	d
a	b1	c	d2
a3	b	c	d

Example from textbook Ch. 3.4.2

THE CHASE TEST FOR LOSSLESS JOIN

R(A,B,C,D) = S1(A,D) ⋈ S2(A,C) ⋈ S3(B,C,D)
R satisfies: A→B, B→C, CD→A

S1 = Π_{AD}(R), S2 = Π_{AC}(R), S3 = Π_{BCD}(R),
hence R ⊆ S1 ⋈ S2 ⋈ S3
Need to check: R ⊇ S1 ⋈ S2 ⋈ S3
Suppose (a,b,c,d) ∈ S1 ⋈ S2 ⋈ S3 Is it also in R?
R must contain the following tuples:

A	B	C	D	Why ?
a	b1	c1	d	(a,d) ∈ S1 = Π_{AD}(R)
a	b2	c	d2	(a,c) ∈ S2 = Π_{BD}(R)
a3	b	c	d	(b,c,d) ∈ S3 = Π_{BCD}(R)

"Chase" them (apply FDs):

A→B

A	B	C	D
a	b1	c1	d
a	b1	c	d2
a3	b	c	d

B→C

A	B	C	D
a	b1	c	d
a	b1	c	d2
a3	b	c	d

Example from textbook Ch. 3.4.2

THE CHASE TEST FOR LOSSLESS JOIN

R(A,B,C,D) = S1(A,D) ⋈ S2(A,C) ⋈ S3(B,C,D)
R satisfies: A→B, B→C, CD→A

S1 = Π_{AD}(R), S2 = Π_{AC}(R), S3 = Π_{BCD}(R),
hence R ⊆ S1 ⋈ S2 ⋈ S3
Need to check: R ⊇ S1 ⋈ S2 ⋈ S3
Suppose (a,b,c,d) ∈ S1 ⋈ S2 ⋈ S3 Is it also in R?
R must contain the following tuples:

A	B	C	D	Why ?
a	b1	c1	d	(a,d) ∈ S1 = Π_{AD}(R)
a	b2	c	d2	(a,c) ∈ S2 = Π_{BD}(R)
a3	b	c	d	(b,c,d) ∈ S3 = Π_{BCD}(R)

"Chase" them (apply FDs):

A→B

A	B	C	D
a	b1	c1	d
a	b1	c	d2
a3	b	c	d

B→C

A	B	C	D
a	b1	c	d
a	b1	c	d2
a3	b	c	d

CD→A

A	B	C	D
a	b1	c	d
a	b1	c	d2
a	b	c	d

Hence R contains (a,b,c,d)

SCHEMA REFINEMENTS = NORMAL FORMS

- **1st Normal Form = all tables are flat**
- **2nd Normal Form = no FD with “non-prime” attributes**
  - *Obsolete*
  - Prime attributes: attributes part of a key
- **Boyce Codd Normal Form = no “bad” FDs**
  - Are there problems with BCNF?

DEPENDENCY PRESERVATION

- **Bookings(title,theatre,city)**
  - FD:
    - theatre -> city
    - title,city -> theatre
- What are the keys?

DEPENDENCY PRESERVATION

- **Bookings(title,theatre,city)**
  - FD:
    - theatre -> city
    - title,city -> theatre
- What are the keys?
  - None of the single attributes
  - {title,city},{theatre,title}
- **BCNF?**

DEPENDENCY PRESERVATION

- **Bookings(title,theatre,city)**
  - FD:
    - theatre -> city
    - title,city -> theatre
- What are the keys?
  - None of the single attributes
  - {title,city},{theatre,title}
- **BCNF?**
  - No, {theatre} is neither a trivial dependency nor a superkey
  - Decompose?

DEPENDENCY PRESERVATION

- **Bookings(title,theatre,city)**
  - FD:
    - theatre -> city
    - title,city -> theatre
- What are the keys?
  - None of the single attributes
  - {title,city},{theatre,title}
- **BCNF?**
  - No, {theatre} is neither a trivial dependency nor a superkey
  - Decompose? R1(theatre,city) R2(theatre,title)
  - What’s wrong? *(think of FDs)*

DEPENDENCY PRESERVATION

- **Bookings(title,theatre,city)**
  - FD:
    - theatre -> city
    - title,city -> theatre
- What are the keys?
  - None of the single attributes
  - {title,city},{theatre,title}
- **BCNF?**
  - No, {theatre} is neither a trivial dependency nor a superkey
  - Decompose? R1(theatre,city) R2(theatre,title)
  - What's wrong? *(think of FDs)*
  - We can't guarantee title,city -> theatre with **simple** constraints (now need to join)

NORMAL FORMS

- **3rd Normal form**
  - Allows tables with BCNF violations if a decomposition separates an FD
  - Can result in redundancy
- **4th Normal form**
  - Multi-valued dependencies
    - Incorporate info about attributes in neither A nor B
    - All MVDs are also FDs
  - Apply BCNF alg with MVD and FD

NORMAL FORMS

- **5th Normal Form**
  - Join dependency
    - Lossless/exact joining
    - Join independent Tables
- **6th Normal Form**
  - Only allow trivial join dependencies
  - Only need key/tuple constraints to represent all constraints

KEY POINTS

- Produce and verify FDs, superkeys, keys
- Be able to decompose a table into BCNF
- Flaws of 1NF & BCNF
- Identify loss and be able to apply the chase test

IMPLEMENTATION

We learned about how to normalize tables to avoid anomalies

How can we implement normalization in SQL if we can't modify existing tables?

- This might be due to legacy applications that rely on previous schemas to run
- Can recover original tables via join on demand and we want those available to queries

VIEWS

A view in SQL =

- A table computed from other tables, s.t., whenever the base tables are updated, the view is updated too

More generally:

- A view is derived data that keeps track of changes in the original data

Compare:

- A function computes a value from other values, but does not keep track of changes to the inputs

Purchase(customer, product, store)
Product(pname, price)

StorePrice(store, price)

A SIMPLE VIEW

Create a view that returns for each store
the prices of products purchased at that store

```
CREATE VIEW  StorePrice AS
  SELECT DISTINCT x.store, y.price
  FROM  Purchase x, Product y
  WHERE x.product = y.pname
```

This is like a new table
StorePrice(store,price)

Purchase(customer, product, store)
Product(pname, price)

StorePrice(store, price)

WE USE A VIEW LIKE ANY TABLE

A "high end" store is a store that sell some products over 1000.

For each customer, return all the high end stores that they visit.

```
SELECT DISTINCT u.customer, u.store
FROM Purchase u, StorePrice v
WHERE u.store = v.store
    AND v.price > 1000
```

TYPES OF VIEWS

Virtual views

- Computed only on-demand – slow at runtime
- Always up to date

Materialized views

- Pre-computed offline – fast at runtime
- May have stale data (must recompute or update)

The key components of physical tuning of databases are the selection of materialized views and indexes

MATERIALIZED VIEWS

```
CREATE MATERIALIZED VIEW View_name
BUILD [IMMEDIATE/DEFERRED]
REFRESH [FAST/COMPLETE/FORCE]
ON [COMMIT/DEMAND]
AS Sql_query
```

- **Immediate v deferred**
  - Build immediately, or after a query
- **Fast v. Complete v. Force**
  - Level of refresh – log based v. complete rebuild
- **Commit v. Demand**
  - Commit: after data is added
  - Demand: after conditions are set (time is common)

CONCLUSION

Poor schemas can lead to bugs and inefficiency

E/R diagrams are means to structurally visualize and design relational schemas

Normalization is a principled way of converting schemas into a form that avoid such problems

BCNF is one of the most widely used normalized form in practice