CS2102

Tutorial 01

Question 12 A

- **Superkeys:** subset of attributes in a relation that *uniquely identifies* its tuples
  - No two distinct tuples of relation have the same values in all attributes of superkey
  - Finding superkey?
    - By enumeration, then
    - By process of elimination

Question 12 A

- **Superkeys:** subset of attributes in a relation that *uniquely identifies* its tuples
  - No two distinct tuples of relation have the same values in all attributes of superkey
  - Finding superkey?
    - By enumeration, then
    - By process of elimination
- **Example:**
  - {A}

R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 A

- **Superkeys:** subset of attributes in a relation that ***uniquely identifies*** its tuples
  - No two distinct tuples of relation have the same values in all attributes of superkey
  - Finding superkey?
    - By enumeration, then
    - By process of elimination
- **Example:**
  - {A}
    - do not violate any superkey property

R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 10 A

- **Superkeys:** subset of attributes in a relation that *uniquely identifies* its tuples
  - No two distinct tuples of relation have the same values in all attributes of superkey
  - Finding superkey?
    - By enumeration, then
    - By process of elimination
- **Example:**
  - {B}

R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 A

- **Superkeys:** subset of attributes in a relation that ***uniquely identifies*** its tuples
  - No two distinct tuples of relation have the same values in all attributes of superkey
  - Finding superkey?
    - By enumeration, then
    - By process of elimination
- **Example:**
  - {B}
    - (**2**,*1*,2,0) and (**1**,*1*,2,0) violates superkey property

R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 A

- **Solution:**
  - Continuing the process, we get the following combinations of attributes
    - {A} {B} {C} {D}
    - {A,B} {A,C} {A,D} {B,C} {B,D} {C,D}
    - {A,B,C} {A,B,D} {A,C,D} {B,C,D}
    - {A,B,C,D}

R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 10 A

- **Solution:**
  - Continuing the process, we get the following combinations of attributes
    - {A} {B} {C} {D}
    - {A,B} {A,C} {A,D} {B,C} {B,D} {C,D}
    - {A,B,C} {A,B,D} {A,C,D} {B,C,D}
    - {A,B,C,D}
  - Only the combinations above are *possible* superkeys

R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 A

- **Solution:**
  - Continuing the process, we get the following combinations of attributes
    - {A} {B} {C} {D}
    - {A,B} {A,C} {A,D} {B,C} {B,D} {C,D}
    - {A,B,C} {A,B,D} {A,C,D} {B,C,D}
    - {A,B,C,D}
  - How many combinations in total?
    - $2^n - 1$ where n is the number of attributes
    - *Can we not check them all?*

R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 A

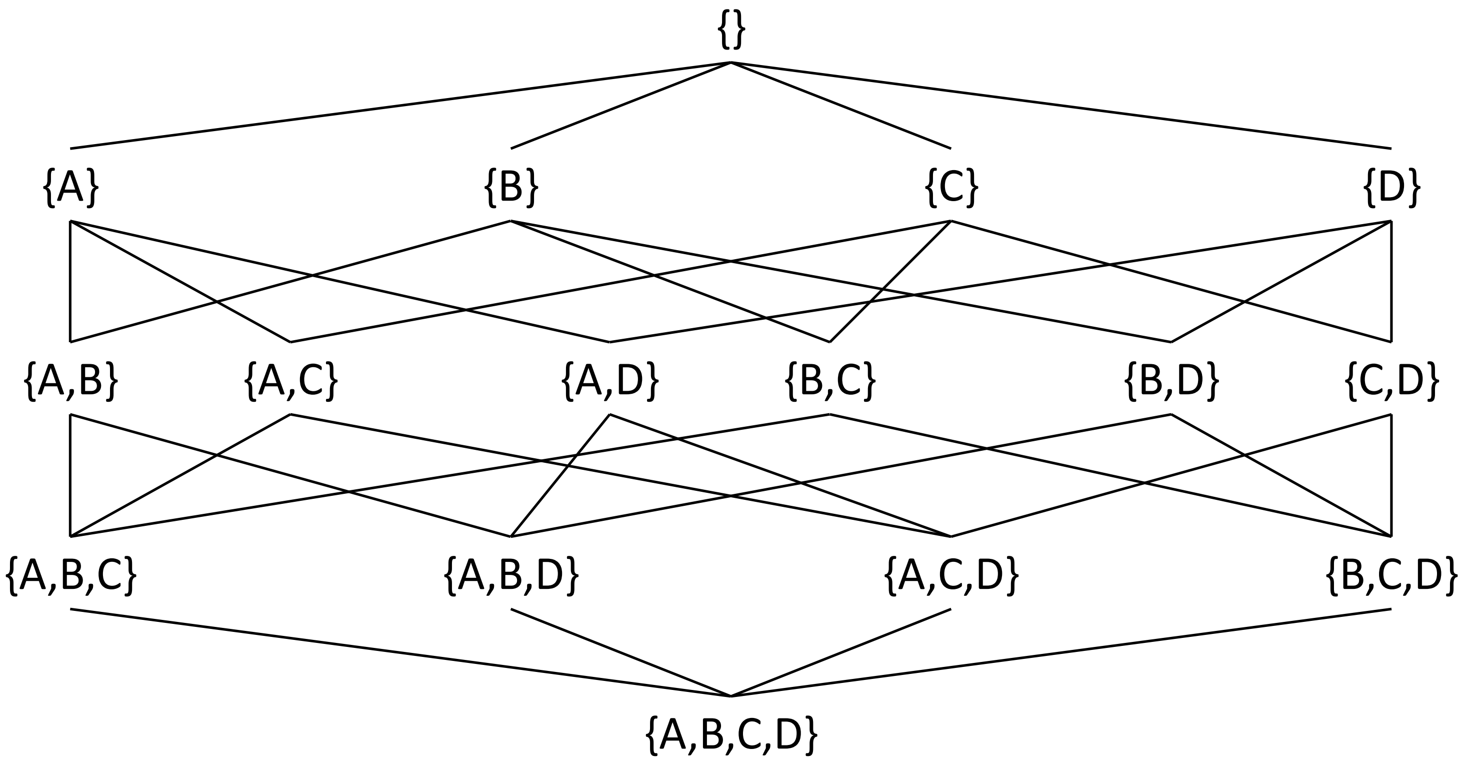
- **Solution:**
  - Continuing the process, we get the following combinations of attributes
    - {A} {B} {C} {D}
    - {A,B} {A,C} {A,D} {B,C} {B,D} {C,D}
    - {A,B,C} {A,B,D} {A,C,D} {B,C,D}
    - {A,B,C,D}
  - How many combinations in total?
    - $2^n - 1$ where n is the number of attributes
    - *Can we not check them all?*

R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 A

- **Improvement:**
  - Arrange the attributes as a *subset* of other attributes

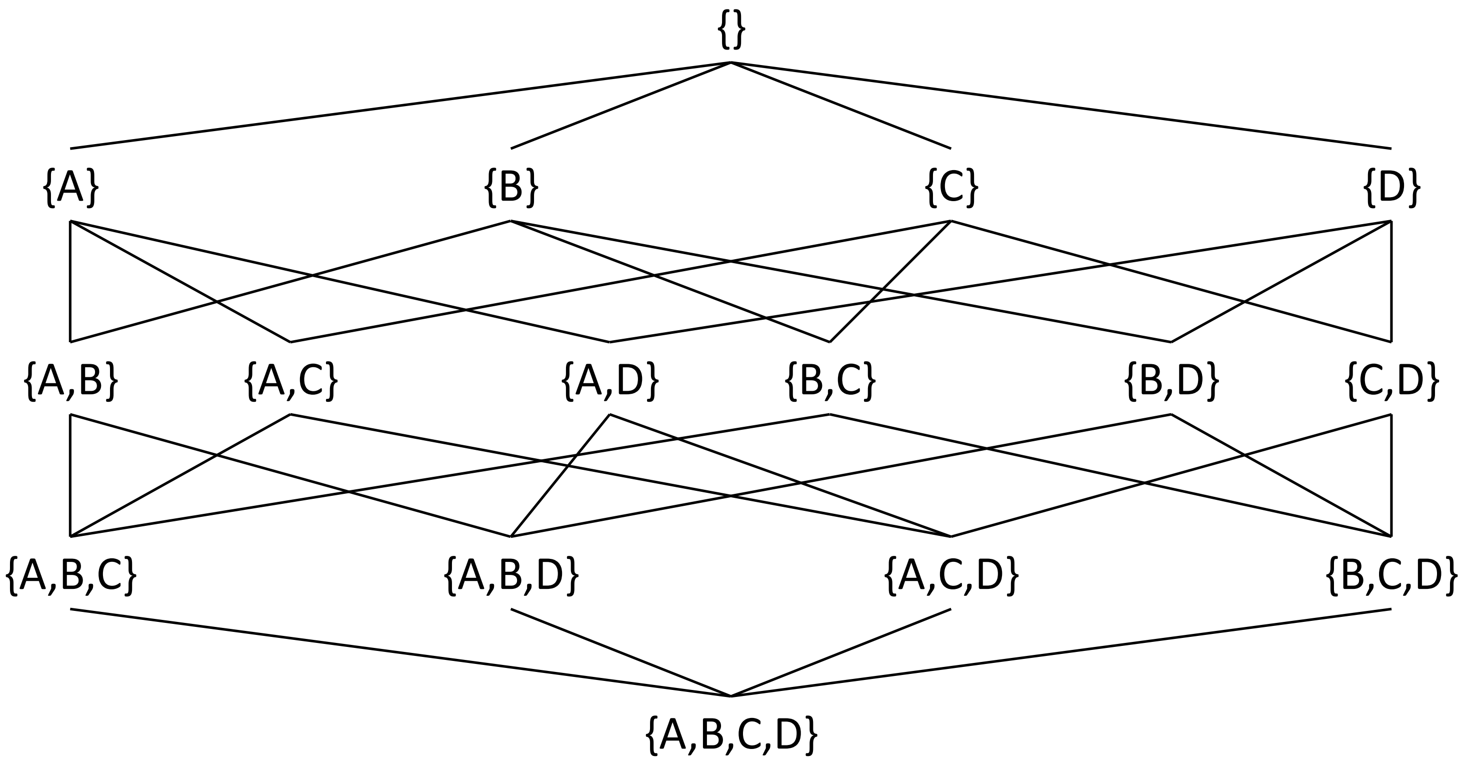


R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 A

- **Improvement:**
  - Check from top to bottom

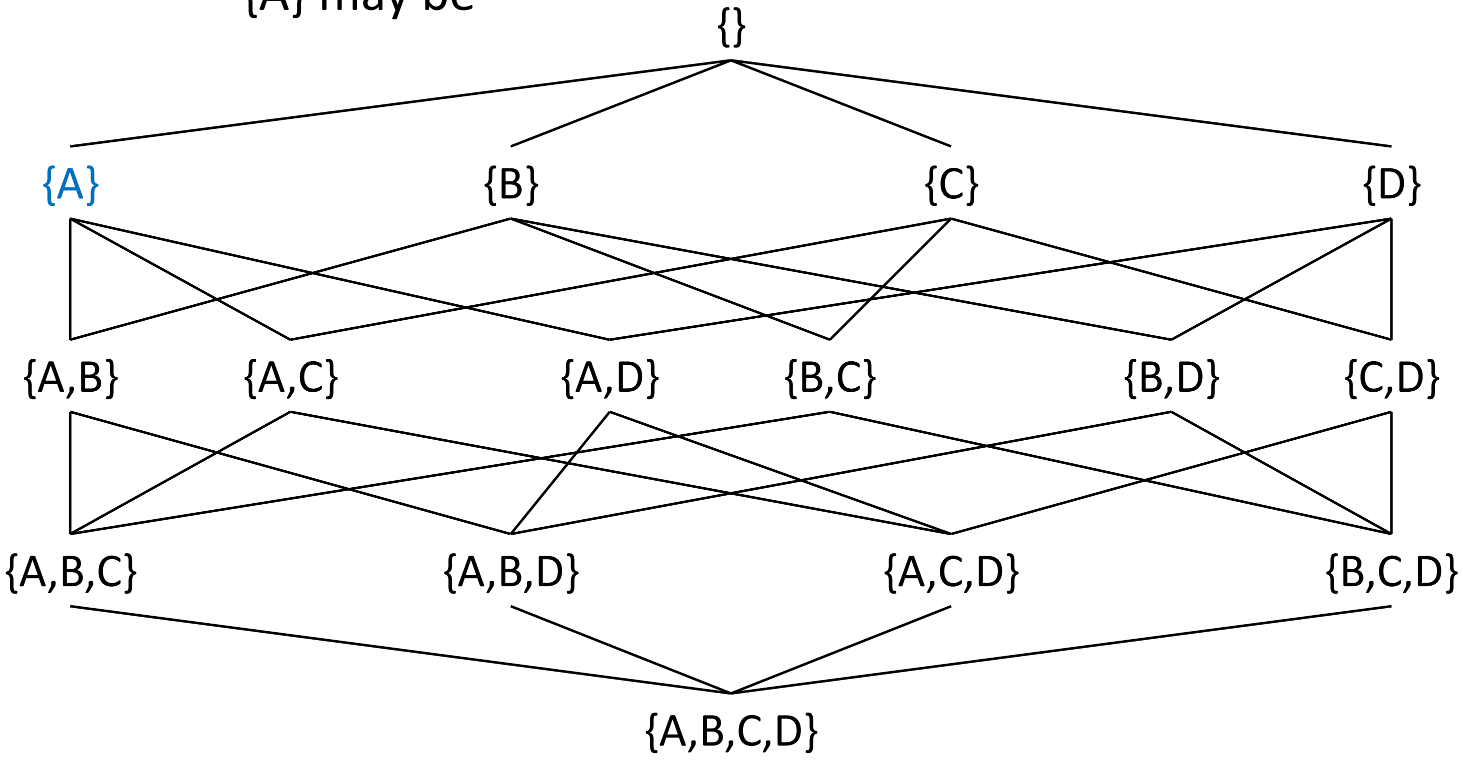


R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 A

- **Improvement:**
  - Check from top to bottom
    - {A} may be

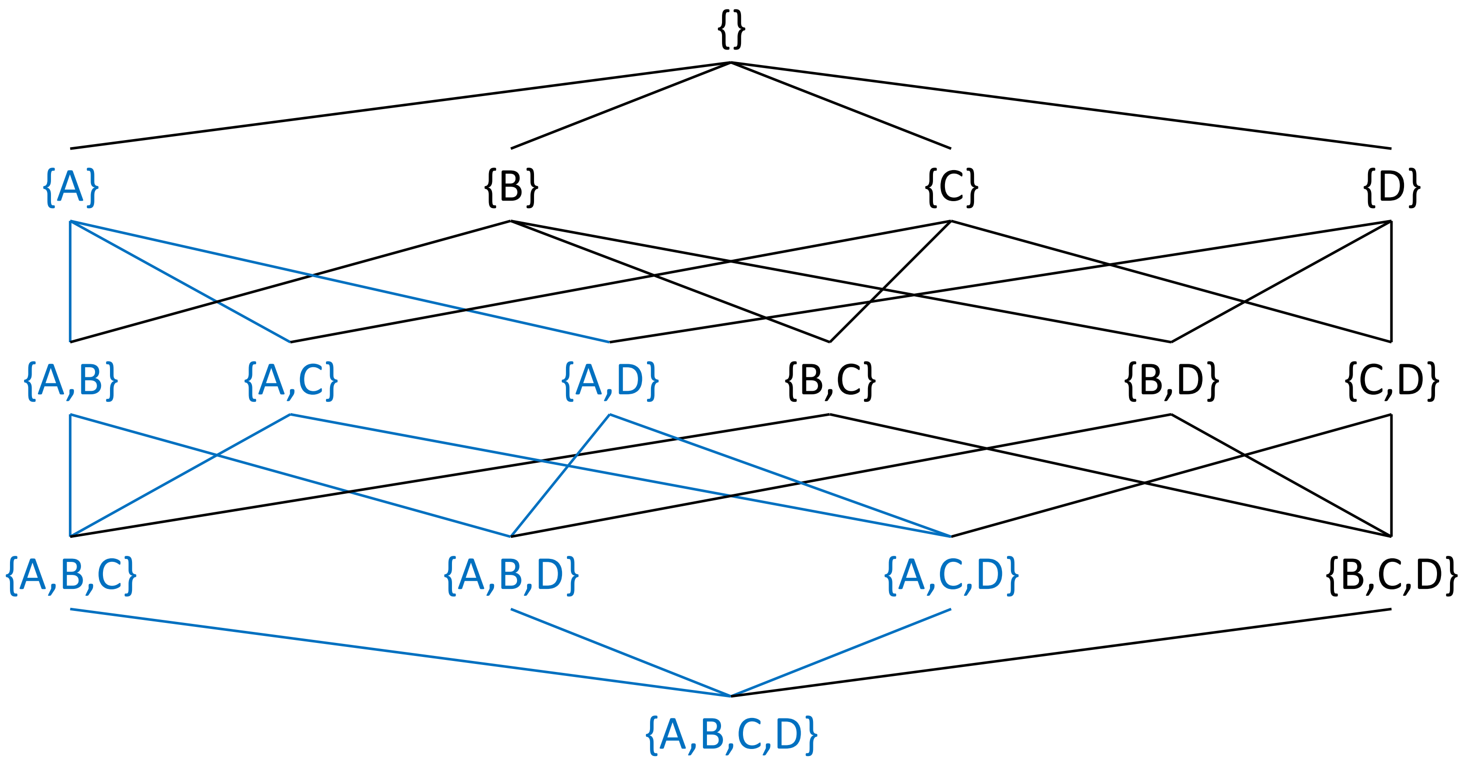


R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 A

- **Improvement:**
  - Anything set that includes {A} may be a superkey!

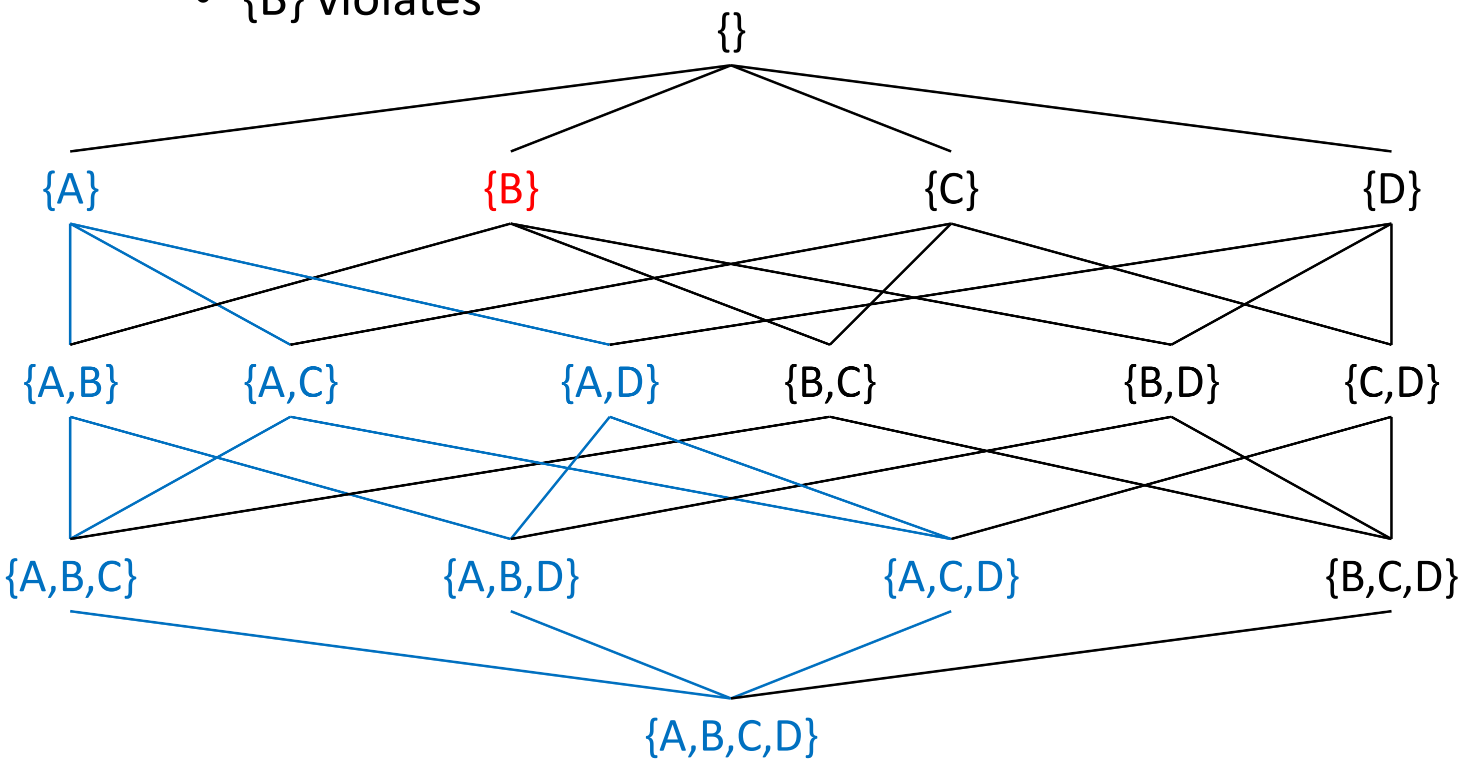


R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 A

- **Improvement:**
  - Check from top to bottom
    - {B} violates

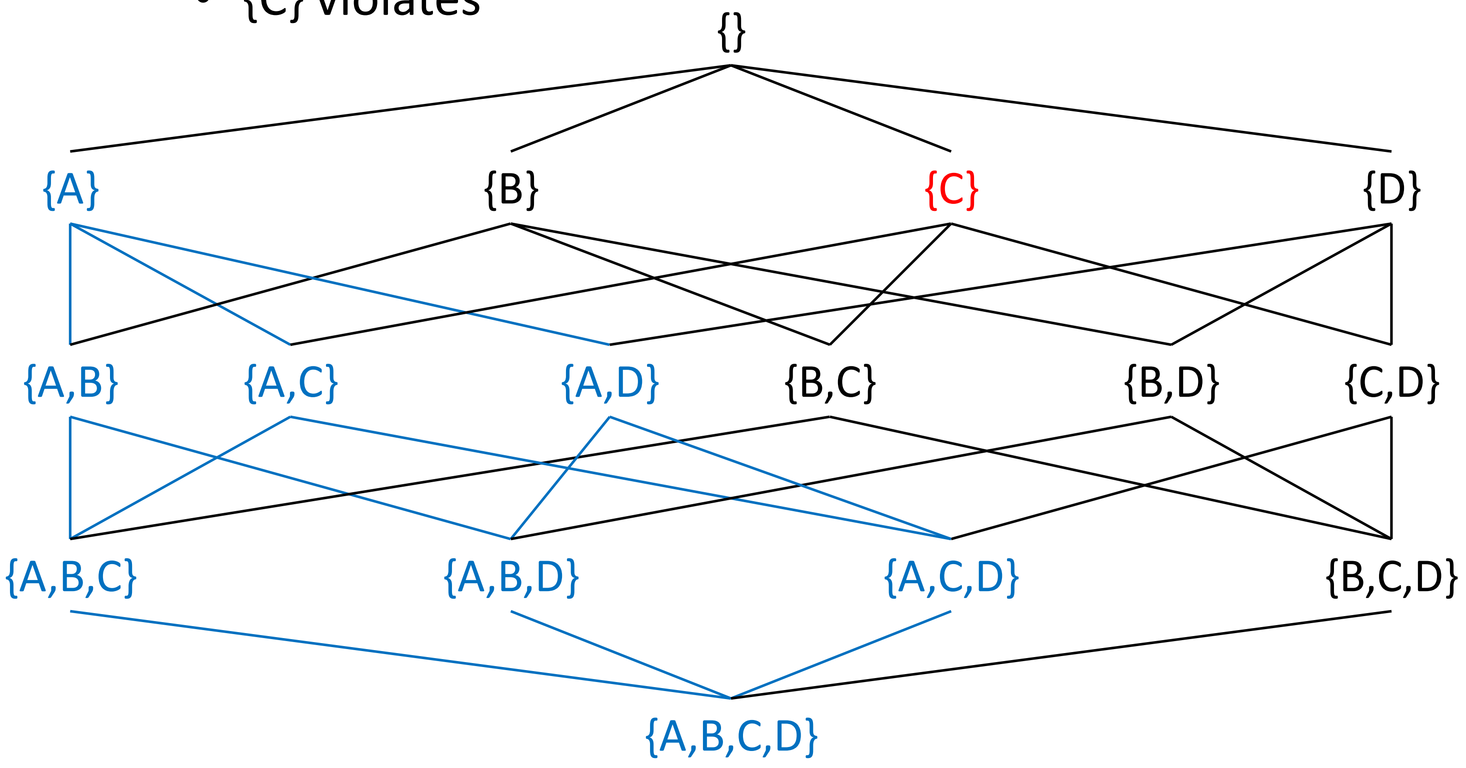


R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 A

- **Improvement:**
  - Check from top to bottom
    - {C} violates

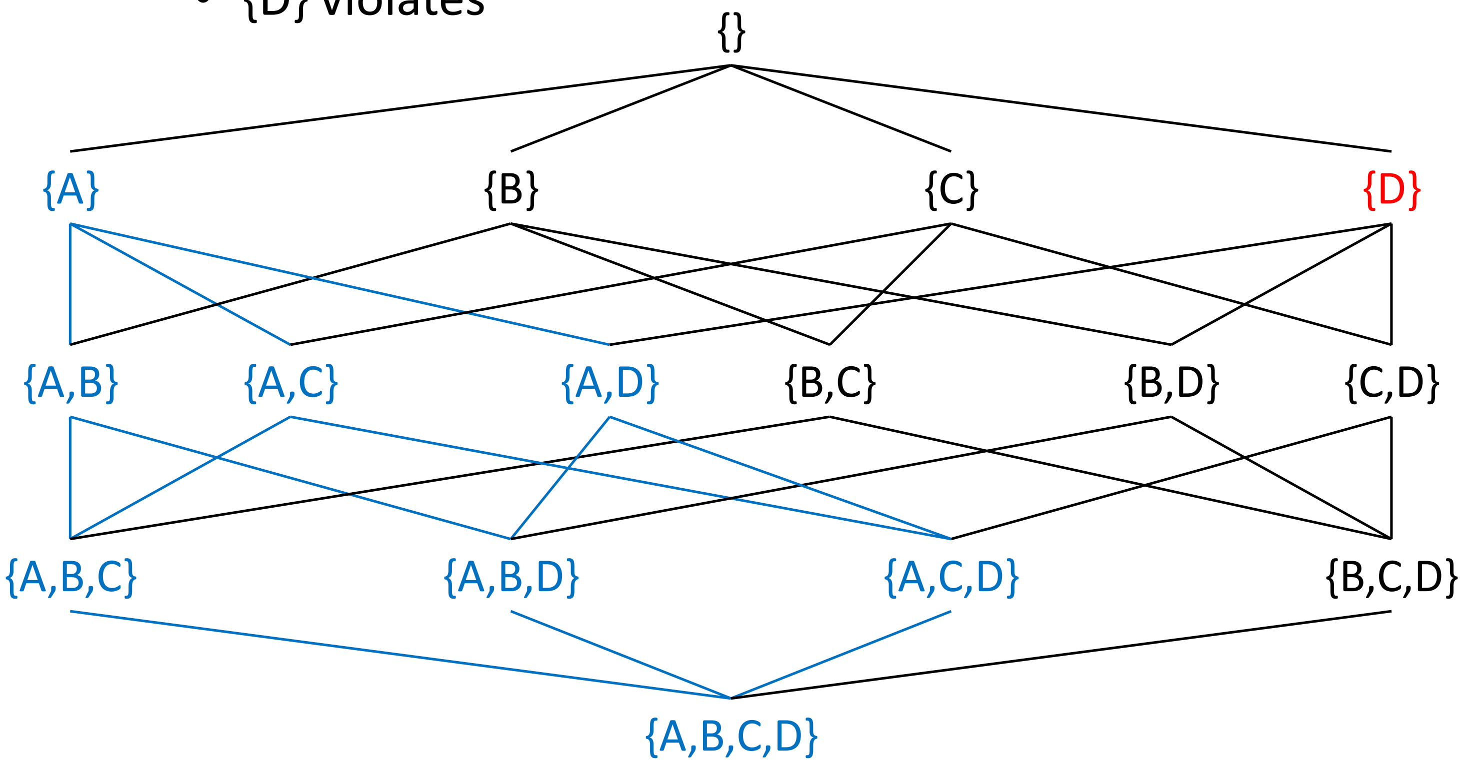


R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 A

- **Improvement:**
  - Check from top to bottom
    - {D} violates

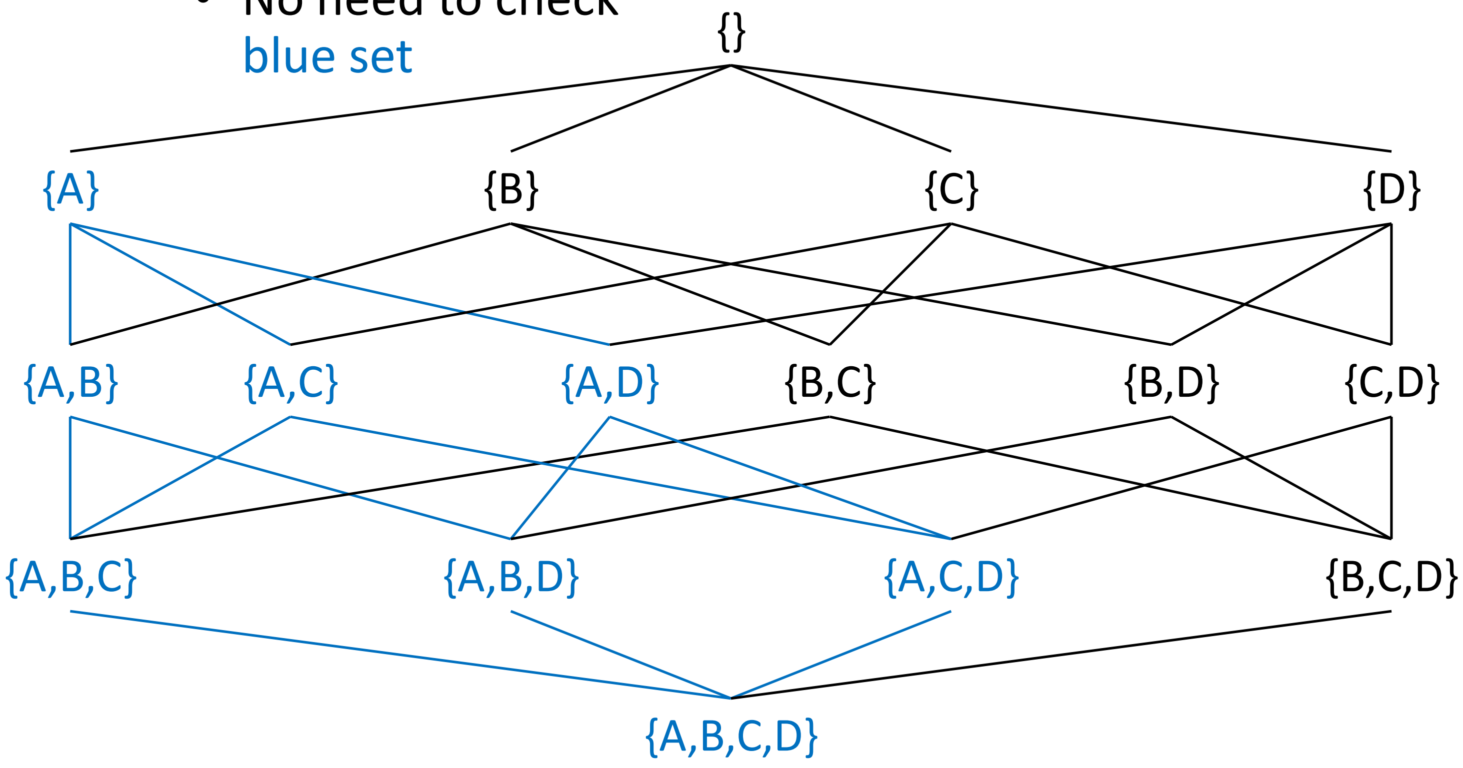


R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 A

- **Improvement:**
  - Check from top to bottom (*at second level*)
    - No need to check blue set

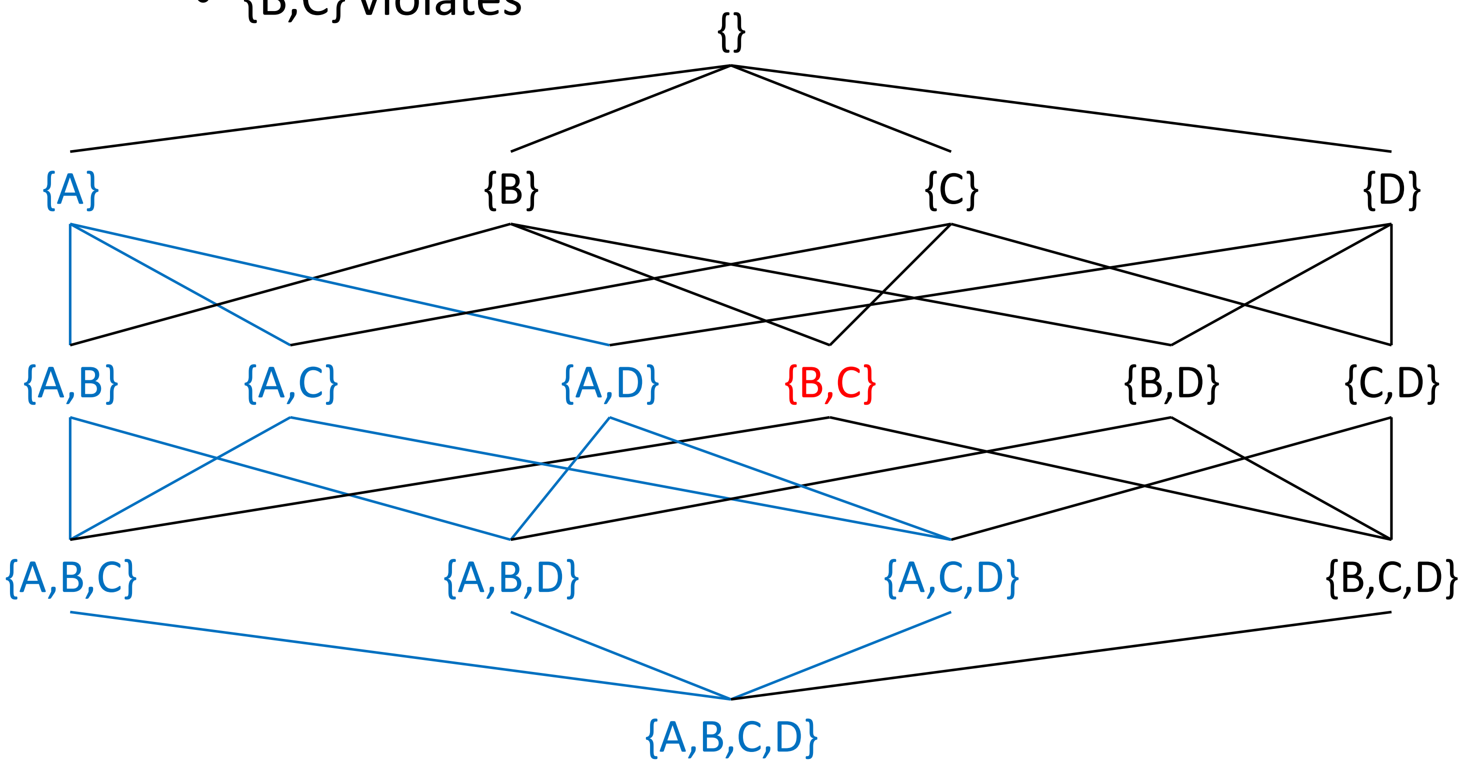


R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 A

- **Improvement:**
  - Check from top to bottom (*at second level*)
    - {B,C} violates

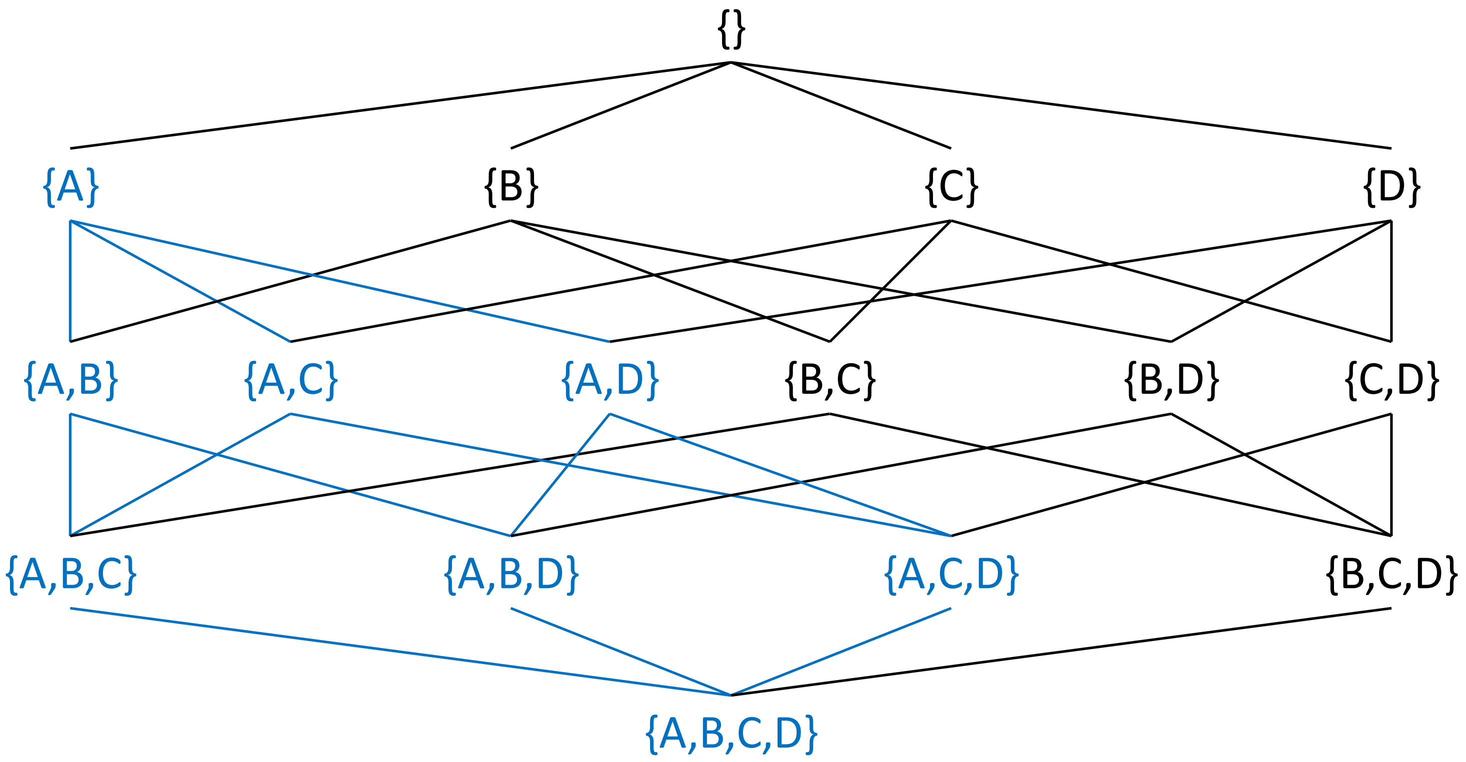


R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 B

- **Candidate keys:** superkey that is *non-null* & *no proper subset* of a key is a superkey

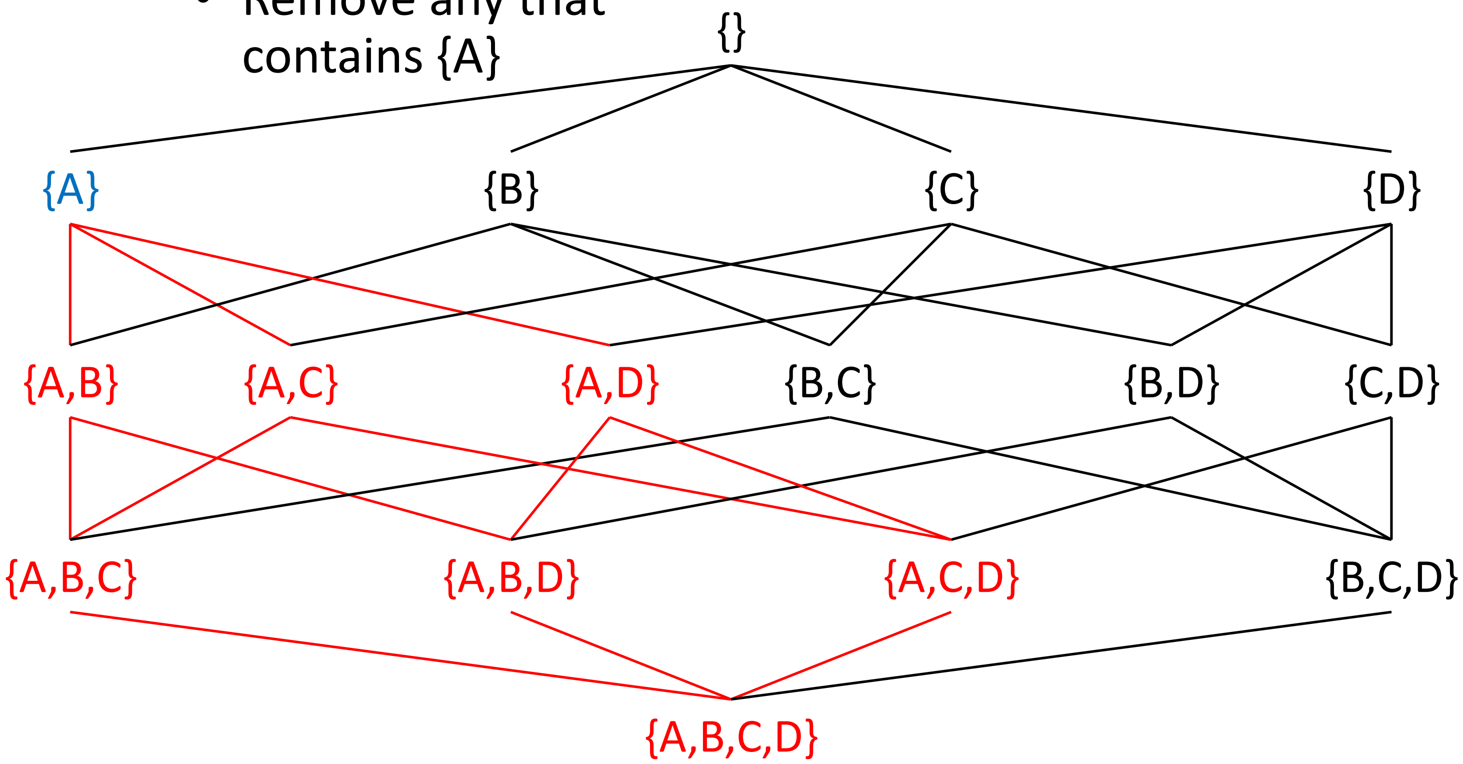


R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 B

- **Candidate keys**
  - We can do top-down approach again, but now remove any superset
    - Remove any that contains {A}

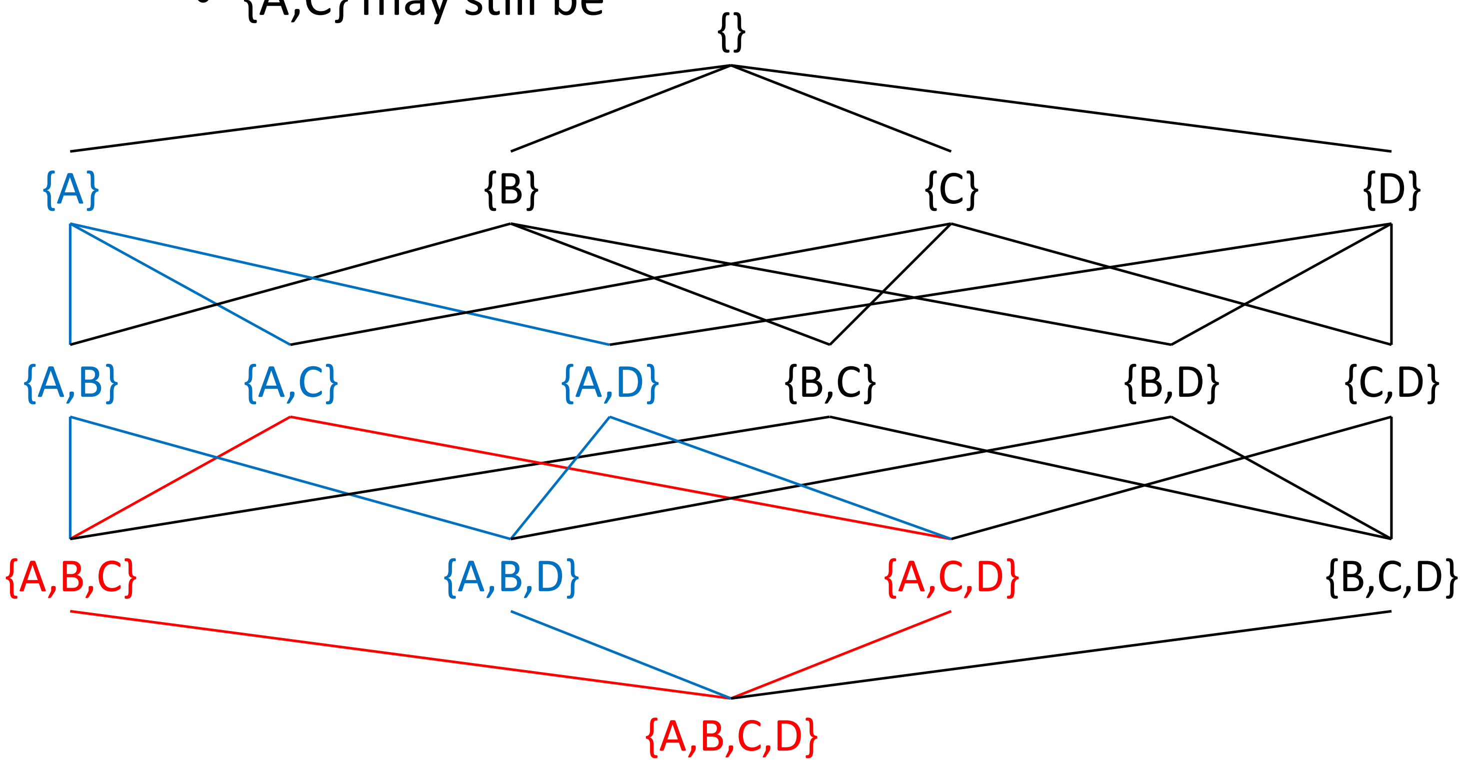


R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 B

- **Candidate keys**
  - But, {A} may NOT be an actual superkey, it is only a possible superkey
    - {A,C} may still be

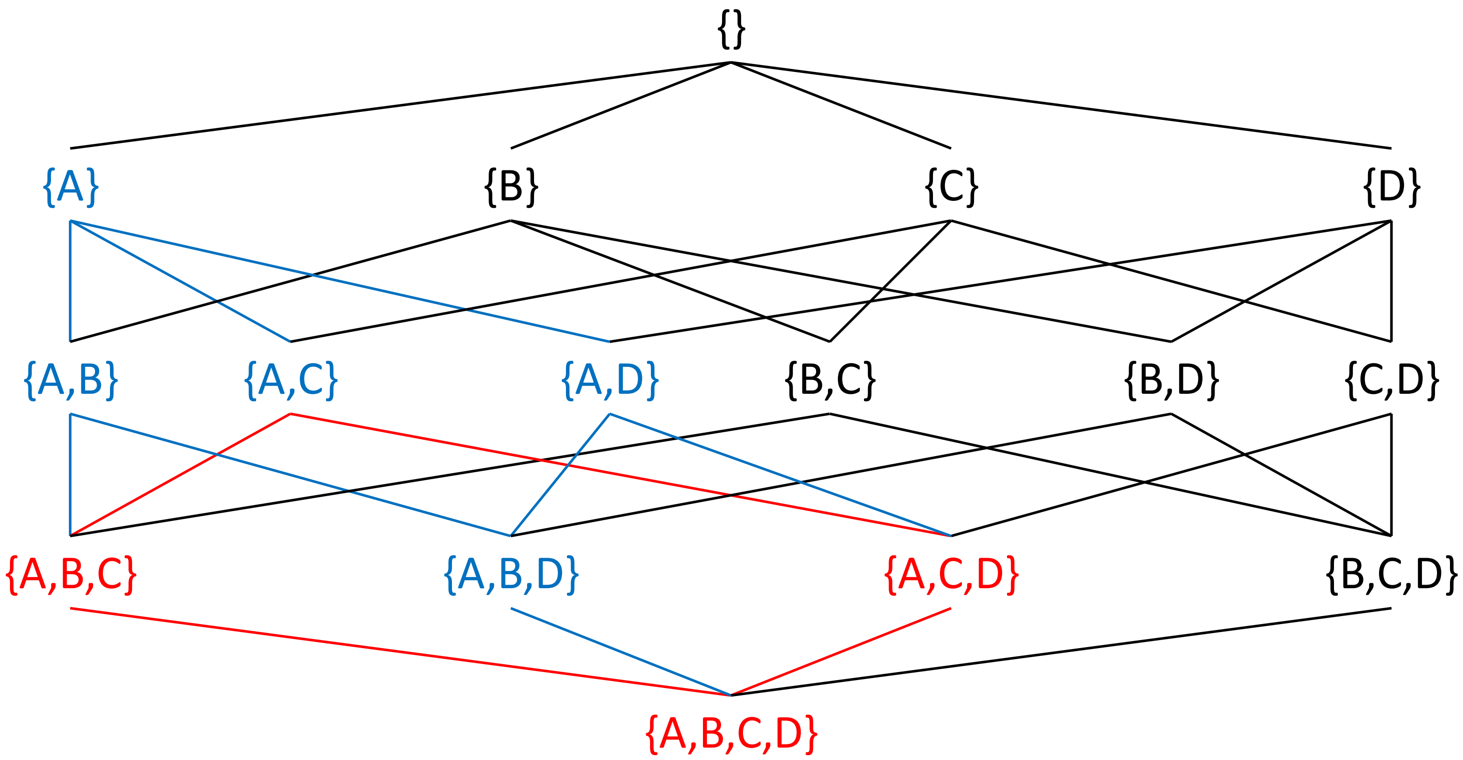


R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 12 B

- **Candidate keys**
  - So the *possible* candidate keys are those in blue



R

A	B	C	D
3	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

Question 13

- **Foreign key:** either *null* or appear as *primary key* in a referenced relation
  - Finding foreign key?
    - Check every attribute and eliminate violations

r

A	B
3	0
2	1
1	1
0	0

s

W	X	Y	Z
0	4	0	null
1	Null	2	null
2	1	2	null
3	0	1	null

Question 13

- **Foreign key:** either *null* or appear as *primary key* in a referenced relation
  - Finding foreign key?
    - Check every attribute and eliminate violations
- **Example:**
  - W No violation
  - X
  - Y
  - Z

r

A	B
3	0
2	1
1	1
0	0

s

W	X	Y	Z
0	4	0	null
1	Null	2	null
2	1	2	null
3	0	1	null

Question 13

- **Foreign key:** either *null* or appear as *primary key* in a referenced relation
  - Finding foreign key?
    - Check every attribute and eliminate violations
- **Example:**
  - W No violation
  - X 4 does not appear in r.A
  - Y
  - Z

r

A	B
3	0
2	1
1	1
0	0

s

W	X	Y	Z
0	4	0	null
1	Null	2	null
2	1	2	null
3	0	1	null

Question 13

- **Foreign key:** either *null* or appear as *primary key* in a referenced relation
  - Finding foreign key?
    - Check every attribute and eliminate violations
- **Example:**
  - W   No violation
  - X   4 does not appear in r.A
  - Y   No violation
  - Z

r

A	B
3	0
2	1
1	1
0	0

s

W	X	Y	Z
0	4	0	null
1	Null	2	null
2	1	2	null
3	0	1	null

Question 13

- **Foreign key:** either *null* or appear as *primary key* in a referenced relation
  - Finding foreign key?
    - Check every attribute and eliminate violations
- **Example:**
  - W   No violation
  - X   4 does not appear in r.A
  - Y   No violation
  - Z   No violation (all null)

r

A	B
3	0
2	1
1	1
0	0

s

W	X	Y	Z
0	4	0	null
1	Null	2	null
2	1	2	null
3	0	1	null

Question 14

- **Equivalent queries:** for every legal instance *d* of *D*, both Q_1 and Q_2 compute the same results on *d*
  - In general, no specific way to check
  - You can use certain property to help find violations/guide your thinking
- **Examples:**
  - Set difference is non-commutative $R - S \neq S - R$
  - Projection removes duplicates
    - Subsequent operations may be affected by this removal of duplicates
  - Operation must be valid
    - Union-compatible
    - The attribute referred must exists

Question 14

a) Equivalent
   - σ removes rows; π removes columns; in the end only the valid cells remain

b) Not equivalent
   - Q_2 refers to non-existent attributes; hence invalid

c) Equivalent
   - For both Q_1 and Q_2, we have D as first column and Y as second column

d) Equivalent
   - π will re-order the columns to be the same

Question 14

e) Equivalent

- This cross-product is associative due to the tuple being “merged”. Normally, cross-product are non-associative in other algebra (like set algebra.

f) Equivalent

- Union before or after projection does not change the order. Duplicates will be removed either way.

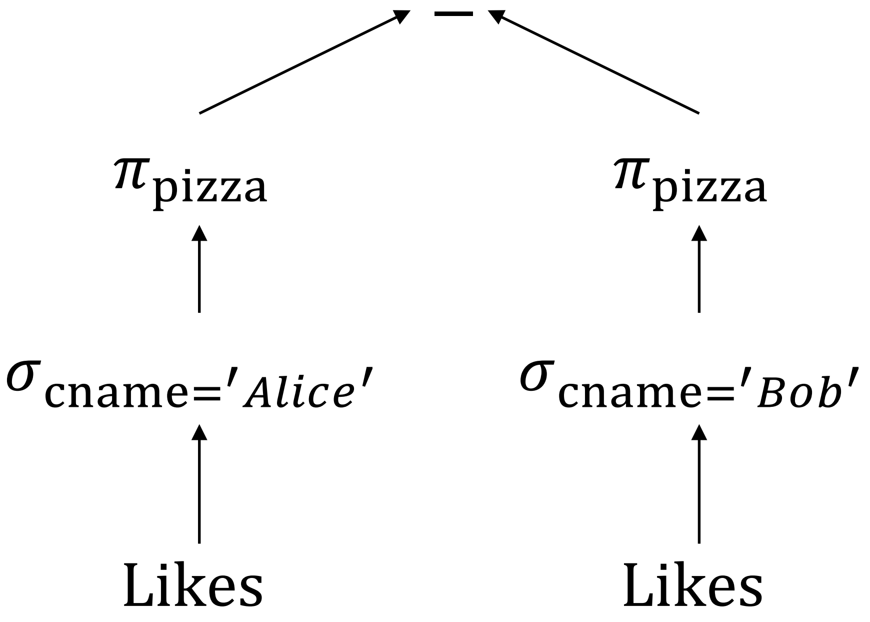
g) Not equivalent

- Let $r = \{(10,10)\}$ and $s = \{(10,20)\}$
- $Q_1 = \pi_A(\{(10,10)\} - \{(10,20)\}) = \pi_A(\{(10,10)\}) = \{(10)\}$
- $Q_2 = \pi_A(\{(10,10)\}) - \pi_A(\{(10,20)\}) = \{(10)\} - \{(10)\} = \emptyset$

Question 15

a) Find pizza that Alice likes but Bob does not like

- Let P_1 be the pizza Alice likes
- Let P_2 be the pizza Bob likes
- Then result is $P_1 - P_2$
- $P_1 = \pi_{\text{pizza}}\left(\sigma_{\text{cname}='\text{Alice}'}(\text{Likes})\right)$
- $P_2 = \pi_{\text{pizza}}\left(\sigma_{\text{cname}='\text{Bob}'}(\text{Likes})\right)$



Question 15

b) Find all customer-restaurant pairs (C,R) where `C` and `R` are both located in the same area, and C likes some pizza that is sold by R.

- To find out location of C
  - `Customers` table
- To find out location of R
  - `Restaurants` table
- To find out what R sells
  - `Sells` table
- To find out what C likes
  - `Likes` table

$\text{Customers} \times \text{Restaurants} \times \text{Sells} \times \text{Likes}$
But need renaming to make sure the columns all have unique names

Question 15

b) Find all customer-restaurant pairs (C,R) where C and R are both located in the same area, and C likes some pizza that is sold by R.

- To find out location of C
  - `Customers` table
- To find out location of R
  - `Restaurants` table
- To find out what R sells
  - `Sells` table
- To find out what C likes
  - `Likes` table

$$\rho_{(\mathrm{n}_1,\mathrm{a}_1)}(\mathrm{Customers}) \\ \times \\ \rho_{(\mathrm{n}_2,\mathrm{a}_2)}(\mathrm{Restaurants}) \\ \times \\ \rho_{(\mathrm{n}_3,\mathrm{p}_3,\mathrm{price})}(\mathrm{Sells}) \\ \times \\ \rho_{(\mathrm{n}_4,\mathrm{p}_4)}(\mathrm{Likes})$$

Question 15

b) Find all customer-restaurant pairs (C,R) where C and R are both located in the same area, and C likes some pizza that is sold by R.

- (C,R) located in the same area
  - $\mathrm{a}_1 = \mathrm{a}_2$
- C likes some pizza sold by R
  - $\mathrm{p}_3 = \mathrm{p}_4$
- Connect `Restaurants` and `Sells`
  - $\mathrm{n}_2 = \mathrm{n}_3$
- Connect `Customers` and `Likes`
  - $\mathrm{n}_1 = \mathrm{n}_4$

$$\sigma_{(a_1=a_2)\wedge(p_3=p_4)\wedge(n_2=n_3)\wedge(n_1=n_4)}\left(\rho_{(\mathrm{n}_1,\mathrm{a}_1)}(\text{Customers}) \times \rho_{(\mathrm{n}_2,\mathrm{a}_2)}(\text{Restaurants}) \times \rho_{(\mathrm{n}_3,\mathrm{p}_3,\text{price})}(\text{Sells}) \times \rho_{(\mathrm{n}_4,\mathrm{p}_4)}(\text{Likes})\right)$$

Question 15

b) Find all customer-restaurant pairs (C,R) where C and R are both located in the same area, and C likes some pizza that is sold by R.

- Projection on only (C,R)
  - $\mathrm{n}_1, \mathrm{n}_2$

$$\boxed{\pi_{\mathrm{n}_1,\mathrm{n}_2}}\ \boxed{\sigma_{\substack{(\mathrm{a}_1=\mathrm{a}_2)\wedge(\mathrm{p}_3=\mathrm{p}_4)\wedge \\ (\mathrm{n}_2=\mathrm{n}_3)\wedge(\mathrm{n}_1=\mathrm{n}_4)}}}\ \boxed{\begin{array}{l} \rho_{(\mathrm{n}_1,\mathrm{a}_1)}(\text{Customers}) \\ \times \\ \rho_{(\mathrm{n}_2,\mathrm{a}_2)}(\text{Restaurants}) \\ \times \\ \rho_{(\mathrm{n}_3,\mathrm{p}_3,\mathrm{price})}(\text{Sells}) \\ \times \\ \rho_{(\mathrm{n}_4,\mathrm{p}_4)}(\text{Likes}) \end{array}}$$

Question 15

c) Suppose that the database contains an additional relation `Dislikes(cname,pizza)` which indicates the pizzas that customers do not like. The database also satisfies the following constraint: for every customer $c \in \pi_{\text{cname}}(\text{Customers})$ and for every pizza $p \in \pi_{\text{pizza}}(\text{Contains})$, either $(c, p) \in \text{Likes}$ or $(c, p) \in \text{Dislikes}$ (*in other words, you know the likes and dislikes of every customers with respect to all pizzas*). Given this database, find all customer pairs (C_1, C_2) such that C_1 likes some pizza that C_2 does not like.

- What C_1 likes
  - `Like` table
- What C_2 dislikes
  - `Dislike` table

$$\rho_{(\text{n}_1,\text{p}_1)}(\text{Likes}) \\ \times \\ \rho_{(\text{n}_2,\text{p}_2)}(\text{Dislikes})$$

Question 15

c) Suppose that the database contains an additional relation `Dislikes(cname,pizza)` which indicates the pizzas that customers do not like. The database also satisfies the following constraint: for every customer $c \in \pi_{\text{cname}}(\text{Customers})$ and for every pizza $p \in \pi_{\text{pizza}}(\text{Contains})$, either $(c, p) \in \text{Likes}$ or $(c, p) \in \text{Dislikes}$ (*in other words, you know the likes and dislikes of every customers with respect to all pizzas*). Given this database, find all customer pairs (C_1, C_2) such that C_1 likes some pizza that C_2 does not like.

- The pizza must be the same
  - $\mathrm{p}_1 = \mathrm{p}_2$
- Do we need to check that the two customers are different?
  - i.e., $\mathrm{n}_1 \neq \mathrm{n}_2$
  - No, because it is guaranteed to be distinct

$$\sigma_{\mathrm{p}_1=\mathrm{p}_2}\left(\rho_{(\mathrm{n}_1,\mathrm{p}_1)}(\text{Likes}) \times \rho_{(\mathrm{n}_2,\mathrm{p}_2)}(\text{Dislikes})\right)$$

Question 15

c) Suppose that the database contains an additional relation `Dislikes(cname,pizza)` which indicates the pizzas that customers do not like. The database also satisfies the following constraint: for every customer $c \in \pi_{\text{cname}}(\text{Customers})$ and for every pizza $p \in \pi_{\text{pizza}}(\text{Contains})$, either $(c, p) \in \text{Likes}$ or $(c, p) \in \text{Dislikes}$ (*in other words, you know the likes and dislikes of every customers with respect to all pizzas*). Given this database, find all customer pairs (C_1, C_2) such that C_1 likes some pizza that C_2 does not like.

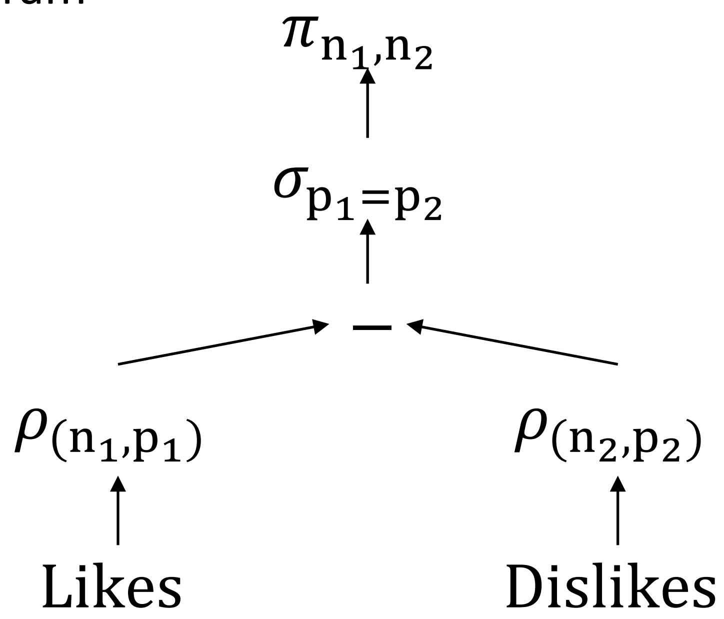
- Project only on (C_1, C_2)
  - $\mathrm{n}_1, \mathrm{n}_2$

$$\boxed{\pi_{\mathrm{n}_1,\mathrm{n}_2}}\ \boxed{\sigma_{\mathrm{p}_1=\mathrm{p}_2}}\ \boxed{\begin{array}{l}\rho_{(\mathrm{n}_1,\mathrm{p}_1)}(\text{Likes}) \\ \times \\ \rho_{(\mathrm{n}_2,\mathrm{p}_2)}(\text{Dislikes})\end{array}}$$

Question 15

c) Suppose that the database contains an additional relation `Dislikes(cname,pizza)` which indicates the pizzas that customers do not like. The database also satisfies the following constraint: for every customer $c \in \pi_{\text{cname}}(\text{Customers})$ and for every pizza $p \in \pi_{\text{pizza}}(\text{Contains})$, either $(c, p) \in \text{Likes}$ or $(c, p) \in \text{Dislikes}$ (*in other words, you know the likes and dislikes of every customers with respect to all pizzas*). Given this database, find all customer pairs (C_1, C_2) such that C_1 likes some pizza that C_2 does not like.

- Diagram



$\pi_{\mathrm{n_1,n_2}}$	$\sigma_{\mathrm{p_1=p_2}}$	$\rho_{(\mathrm{n_1,p_1})}(\text{Likes})$ $\times$ $\rho_{(\mathrm{n_2,p_2})}(\text{Dislikes})$

Question 15

d) Consider the original database schema without the `Dislikes` relation. Write a query to compute the `Dislikes` relation.

- Since we know all the `Likes`
- We need to get all combination of customer name and pizza

- Then remove the `Likes`

Question 15

d) Consider the original database schema without the `Dislikes` relation. Write a query to compute the `Dislikes` relation.
- Since we know all the `Likes`
- We need to get all combination of customer name and pizza
  - Customer name: $\pi_{\text{cname}}(\text{Customers})$
- Then remove the `Likes`

Question 15

d) Consider the original database schema without the `Dislikes` relation. Write a query to compute the `Dislikes` relation.

- Since we know all the `Likes`
- We need to get all combination of customer name and pizza
  - Customer name: $\pi_{\text{cname}}(\text{Customers})$
  - Combinarion: $\pi_{\text{cname}}(\text{Customers}) \times \text{Pizza}$
- Then remove the `Likes`

Question 15

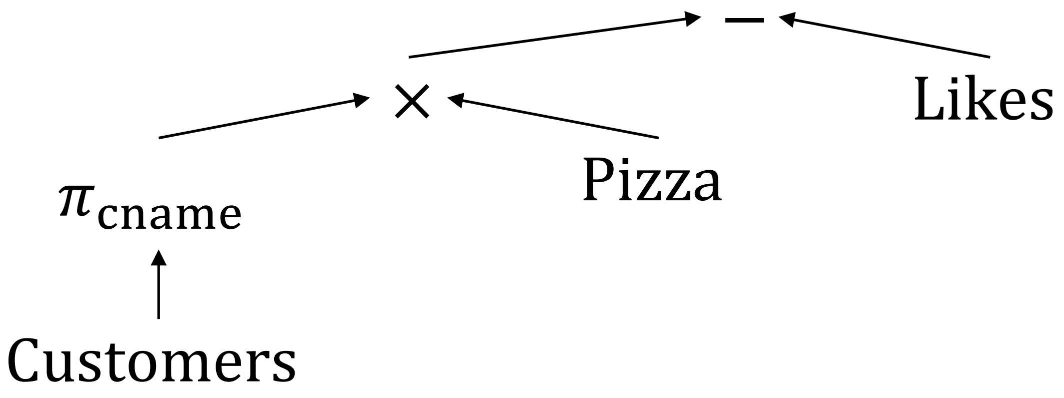
d) Consider the original database schema without the `Dislikes` relation. Write a query to compute the `Dislikes` relation.

- Since we know all the `Likes`
- We need to get all combination of customer name and pizza
  - Customer name: $\pi_{\text{cname}}(\text{Customers})$
  - Combinarion: $\pi_{\text{cname}}(\text{Customers}) \times \text{Pizza}$
- Then remove the `Likes`
  - $(\pi_{\text{cname}}(\text{Customers}) \times \text{Pizza}) - \text{Likes}$

Question 15

d) Consider the original database schema without the `Dislikes` relation. Write a query to compute the `Dislikes` relation.

- Since we know all the `Likes`
- We need to get all combination of customer name and pizza
  - Customer name: $\pi_{\text{cname}}(\text{Customers})$
  - Combinarion: $\pi_{\text{cname}}(\text{Customers}) \times \text{Pizza}$
- Then remove the `Likes`
  - $(\pi_{\text{cname}}(\text{Customers}) \times \text{Pizza}) - \text{Likes}$



Question 15

e) Find all customer pairs (C_1, C_2) such that $C_1 < C_2$ and they like exactly the same pizzas.
  - *Deceptively simple statement!*
  - We need to first get customers that like some pizza another customer dislikes
    - We already have this: from Q15 (c)
    - Let's call it LD

Question 15

e) Find all customer pairs (C_1, C_2) such that $C_1 < C_2$ and they like exactly the same pizzas.

- <u>*Deceptively simple statement!*</u>
- We need to first get customers that like some pizza another customer dislikes
  - We already have this: from Q15 (c)
  - Let's call it LD

But LD does not have ordering!
We need $C_1 < C_2$

Question 15

e) Find all customer pairs (C_1, C_2) such that $C_1 < C_2$ and they like exactly the same pizzas.

- *Deceptively simple statement!*
- We need to first get customers that like some pizza another customer dislikes
  - We already have this: from Q15 (c)
  - Let's call it LD

Problem But LD does not have ordering! We need $C_1 < C_2$
Solution Reorder column from LD and union it with original LD; then we have both ordering.

Question 15

e) Find all customer pairs (C_1, C_2) such that $C_1 < C_2$ and they like exactly the same pizzas.

- *Deceptively simple statement!*
- We need to first get customers that like some pizza another customer dislikes
    - We already have this: from Q15 (c)
    - Let's call it LD

$$\boxed{\mathrm{LD} \cup \left(\pi_{\mathrm{n}_2,\mathrm{n}_1}(\mathrm{LD})\right)}$$

Question 15

e) Find all customer pairs (C_1, C_2) such that $C_1 < C_2$ and they like exactly the same pizzas.

- *Deceptively simple statement!*
- We need to first get customers that like some pizza another customer dislikes
  - We already have this: from Q15 (c)
  - Let's call it LD
- We need to get other pairs besides those in LD
  - Then we can use set difference operator

$$\boxed{\mathrm{LD} \cup \left(\pi_{\mathrm{n_2,n_1}}(\mathrm{LD})\right)}$$

Question 15

e) Find all customer pairs (C_1, C_2) such that $C_1 < C_2$ and they like exactly the same pizzas.

- *<u>Deceptively simple statement!</u>*
- We need to first get customers that like some pizza another customer dislikes
  - We already have this: from Q15 (c)
  - Let's call it LD
- We need to get other pairs besides those in LD
  - Then we can use set difference operator
- Question: *<u>must they like at least one pizza</u>*?
  - YES: Use the Likes table (if not there, customer likes no pizza)
  - NO: Use Customers table (all customers are there)

$$\boxed{\mathrm{LD} \cup \left(\pi_{\mathrm{n}_2,\mathrm{n}_1}(\mathrm{LD})\right)}$$

Question 15

e) Find all customer pairs (C_1, C_2) such that $C_1 < C_2$ and they like exactly the same pizzas.

- *Deceptively simple statement!*
- We need to first get customers that like some pizza another customer dislikes
  - We already have this: from Q15 (c)
  - Let's call it LD
- We need to get other pairs besides those in LD
  - Then we can use set difference operator
- Question: *must they like at least one pizza*?
  - YES: $\mathrm{Likes} \times \mathrm{Likes}$
  - NO: $\mathrm{Customers} \times \mathrm{Customers}$

$$\boxed{\mathrm{LD} \cup \left(\pi_{\mathrm{n_2,n_1}}(\mathrm{LD})\right)}$$

Question 15

e) Find all customer pairs (C_1, C_2) such that $C_1 < C_2$ and they like exactly the same pizzas.

- *Deceptively simple statement!*
- We need to first get customers that like some pizza another customer dislikes
  - We already have this: from Q15 (c)
  - Let's call it LD
- We need to get other pairs besides those in LD
  - Then we can use set difference operator
- Question: *must they like at least one pizza*?
  - YES: $\text{Likes} \times \text{Likes}$
  - NO: $\text{Customers} \times \text{Customers}$

$$\boxed{\text{LD} \cup \left(\pi_{\text{n}_2,\text{n}_1}(\text{LD})\right)}$$

$$\boxed{\pi_{\text{n}_1,\text{n}_2}\left(\sigma_{\text{n}_1<\text{n}_2}\left(\text{Likes} \times \rho_{(\text{n}_2,\text{p}_2)}(\text{Likes})\right)\right)}$$

Question 15

e) Find all customer pairs (C_1, C_2) such that $C_1 < C_2$ and they like exactly the same pizzas.

- *Deceptively simple statement!*
- We need to first get customers that like some pizza another customer dislikes
  - We already have this: from Q15 (c)
  - Let's call it LD
- We need to get other pairs besides those in LD
  - Then we can use set difference operator
- Question: *must they like at least one pizza*?
  - YES: $\mathrm{Likes} \times \mathrm{Likes}$
  - NO: $\mathrm{Customers} \times \mathrm{Customers}$

$$\boxed{\mathrm{LD} \cup \left(\pi_{\mathrm{n_2,n_1}}(\mathrm{LD})\right)}$$

$$\boxed{\pi_{\mathrm{n_1,n_2}}\left(\sigma_{\mathrm{n_1<n_2}}\left(\mathrm{Customers}\right.\right.}$$

Question 15

f) For each restaurant, find the price of the most expensive pizzas sold by that restaurant. Excludes restaurants that do not sell any pizza.

- How to find the most expensive pizza?

Question 15

f) For each restaurant, find the price of the most expensive pizzas sold by that restaurant. Excludes restaurants that do not sell any pizza.

- How to find the most expensive pizza?
  - $\text{price} \geq \text{price}_2$ for any price
  - $\text{rname} = \text{rname}_2$ must be the same restaurant

Question 15

f) For each restaurant, find the price of the most expensive pizzas sold by that restaurant. Excludes restaurants that do not sell any pizza.

- How to find the most expensive pizza?
  - $\text{price} \geq \text{price}_2$ for any price
  - $\text{rname} = \text{rname}_2$ must be the same restaurant
  - BUT WAIT!
    - This is the most expensive pizza from ALL possible restaurants!

Question 15

f) For each restaurant, find the price of the most expensive pizzas sold by that restaurant. Excludes restaurants that do not sell any pizza.

- How to find the most expensive pizza?
    - Think in reverse!
    - What if we find all the NOT most expensive pizza?
        - $(\text{price} < \text{price}_2) \wedge (\text{rname} = \text{rname}_2)$
    - Then we remove this pizza from the solution!

Question 15

f) For each restaurant, find the price of the most expensive pizzas sold by that restaurant. Excludes restaurants that do not sell any pizza.

- How to find the most expensive pizza?
  - Think in reverse!
  - What if we find all the NOT most expensive pizza?
    - $Q_{\text{not\_expensive}} \text{Sells} \times \rho_{(\text{rname}_2, \text{pizza}_2, \text{price}_2)}(\text{Sells})$
  - Then we remove this pizza from the solution!

Question 15

f) For each restaurant, find the price of the most expensive pizzas sold by that restaurant. Excludes restaurants that do not sell any pizza.

- How to find the most expensive pizza?
  - Think in reverse!
  - What if we find all the NOT most expensive pizza?
    - $\mathrm{Q}_{\mathrm{not\_expensive}} = \mathrm{Sells} \times \rho_{(\mathrm{rname}_2, \mathrm{pizza}_2, \mathrm{price}_2)}(\mathrm{Sells})$
  - Then we remove this pizza from the solution!
    - $\pi_{\mathrm{rname,price}}(\mathrm{Sells}) - \pi_{\mathrm{rname,price}}\left(\mathrm{Q}_{\mathrm{not\_expensive}}\right)$