CS 480: Database Systems

Relational Algebra

Data Manipulation Languages (DML)

- Language for **accessing** and manipulating the data.
- Languages for expressing *queries.*
- Queries – requests for information from the database.
- Ex.
  - Retrieve the supplier names for suppliers located in London.
  - Retrieve the parts that are shipped by suppliers in Paris (involves 2 relations).

Data Manipulation Languages (DML)

- Procedural – Require the user to specify *what* data are needed and *how* to get those data.
- Declarative (non-procedural) – Require the user to specify *what* data are needed without specifying how to get those data.
- Formal Languages
  - **Relational Algebra** (procedural), **Tuple Relational Calculus** (declarative)
- Commercial
  - **SQL** (non-procedural), QBE, Datalog

Relational Algebra

- Relational Algebra is a set of operators.
- Each operator is a function that maps one or two relations into a new relation.
- Fundamental Operators:
  - Projection, Π
  - Selection, σ
  - Union, ∪
  - Difference, –
  - Cartesian Product, ×
- Complex Operators – Can be performed by combining fundamental operators.
  - Natural Join, ⋈
  - Intersection, ∩
  - Division, ÷

Projection (Π)

- Unary Operator that eliminates columns from a relation.
- Syntax: $\Pi_{A_{i_1},A_{i_2},\ldots,A_{i_k}}(r)$
- Example:
  - In what cities are there suppliers?
  - Give me the part-names and their weights in the database.

Projection (Π)

- In what cities are there suppliers?
- $\Pi_{city}(supplier)$

supplier:

Supplier-id	Supplier-name	City
123	ACME	Chicago
234	BBQ Inc.	New York
345	COLE	London
456	Dave's	Chicago

Projection (Π)

- In what cities are there suppliers?
- $\Pi_{city}(supplier)$

supplier:

Supplier-id	Supplier-name	City
123	ACME	Chicago
234	BBQ Inc.	New York
345	COLE	London
456	Dave's	Chicago

$\Pi_{city}(supplier)$

City
Chicago
London
New York

Projection (Π)

- Give me the part names and their weights in the database.
- Syntax: $\Pi_{\text{Part-name,Weight}}(\text{part})$

part:

Part-id	Part-name	Color	Weight
12	Nut	Gray	12
15	Bolt	Orange	17
5	Screw	Blue	17
7	Screw	Gray	17
8	Screw	Gray	12

Projection (Π)

- Give me the part names and their weights in the database.
- Syntax: $\Pi_{\text{Part-name,Weight}}(\text{part})$

part:

Part-id	Part-name	Color	Weight
12	Nut	Gray	12
15	Bolt	Orange	17
5	Screw	Blue	17
7	Screw	Gray	17
8	Screw	Gray	12

How many tuples in the result?

Projection (Π)

- Give me the part names and their weights in the database.
- Syntax: $\Pi_{\text{Part-name,weight}}(\text{Part})$

part:

Part-id	Part-name	Color	Weight
12	Nut	Gray	12
15	Bolt	Orange	17
5	Screw	Blue	17
7	Screw	Gray	17
8	Screw	Gray	12

$\Pi_{\text{Part-name,weight}}(\text{Part})$

Part-name	Weight
Nut	12
Bolt	17
Screw	17
Screw	12

Projection (Π)

- More formal definition:
  - Let $R(A_1,A_2,\ldots,A_n)$ be a relation scheme
  - Let $r(R)$ be a relation.
  - Let $\{B_1,B_2,\ldots,B_k\} \subseteq \{A_1,A_2,\ldots,A_n\}$
  - Then

$$\Pi_{B_1,B_2,\ldots,B_k}(r) = \{t : \exists u \in r \text{ such that } u(B_1, B_2, \ldots, B_k) = t\}$$

$u(B_1,B_2,\ldots,B_k)$ is the tuple u in which the values of each attribute not in $\{B_1,B_2,\ldots,B_k\}$ has been eliminated.

Projection (Π)

- More formal definition:
  - Let $R(A_1,A_2,\ldots,A_n)$ be a relation scheme
  - Let $r(R)$ be a relation.
  - Let $\{B_1,B_2,\ldots,B_k\} \subseteq \{A_1,A_2,\ldots,A_n\}$
  - Then

$$\Pi_{B_1,B_2,\ldots,B_k}(r) = \{t : \exists u \in r \text{ such that } u(B_1, B_2, \ldots, B_k) = t\}$$

$u(B_1,B_2,\ldots,B_k)$ is the tuple u in which the values of each attribute not in $\{B_1,B_2,\ldots,B_k\}$ has been eliminated.

What are the attributes in the relation scheme for the projected relation?

Projection (Π)

- More formal definition:
  - Let $R(A_1,A_2,\ldots,A_n)$ be a relation scheme
  - Let $r(R)$ be a relation.
  - Let $\{B_1,B_2,\ldots,B_k\} \subseteq \{A_1,A_2,\ldots,A_n\}$
  - Then

$$\Pi_{B_1,B_2,\ldots,B_k}(r) = \{t : \exists u \in r \text{ such that } u(B_1, B_2, \ldots, B_k) = t\}$$

$u(B_1,B_2,\ldots,B_k)$ is the tuple u in which the values of each attribute not in $\{B_1,B_2,\ldots,B_k\}$ has been eliminated.

What are the attributes in the relation scheme for the projected relation?

The projection will be defined over attributes $B_1,B_2,\ldots,B_k$.

Selection (σ)

- Unary operator that eliminates tuples from a relation.
- Syntax: $\sigma_F(r)$, where F is a boolean formula
- Ex:
  - Give me the suppliers located in Chicago.
  - Give me the parts that have gray color and weigh less than 15.

Selection (σ)

- Give me the suppliers located in Chicago.
- $\sigma_{\text{City='Chicago'}}(\textit{supplier})$

supplier:

Supplier-id	Supplier-name	City
123	ACME	Chicago
234	BBQ Inc.	New York
345	COLE	London
456	Dave's	Chicago

Selection (σ)

- Give me the suppliers located in Chicago.
- $\sigma_{\text{City='Chicago'}}(supplier)$

supplier:

Supplier-id	Supplier-name	City
123	ACME	Chicago
234	BBQ Inc.	New York
345	COLE	London
456	Dave's	Chicago

$\sigma_{\text{City='Chicago'}}(supplier)$

Supplier-id	Supplier-name	City
123	ACME	Chicago
456	Dave's	Chicago

Selection (σ)

- Give me the parts that have gray color and weigh less than 15.
- $\sigma_{\text{Color='Gray'} \wedge \text{Weight<15}}$ (*part*)

part:

Part-id	Part-name	Color	Weight
12	Nut	Gray	12
15	Bolt	Orange	17
5	Screw	Blue	17
7	Screw	Gray	17
8	Screw	Gray	12

Selection (σ)

- Give me the parts that have gray color and weigh less than 15.
- $\sigma_{\text{Color='Gray'} \wedge \text{Weight<15}}$ (*part*)

part:

Part-id	Part-name	Color	Weight
12	Nut	Gray	12
15	Bolt	Orange	17
5	Screw	Blue	17
7	Screw	Gray	17
8	Screw	Gray	12

$\sigma_{\text{Color='Gray'} \wedge \text{Weight<15}}$ (*part*)

Part-id	Part-name	Color	Weight
12	Nut	Gray	12
8	Screw	Gray	12

Selection (σ)

- More formally:
  - Let *F* be a boolean formula. Comparisons like {=, <, >, ≤, ≥, ≠} with boolean operators like {∧, ∨}.
  - Let *r* be a relation with relation scheme $R(A_1, A_2, \ldots, A_n)$.
  - Then,

$$\sigma_F(r) = \{t : t \in r \text{ and } t \text{ satisfies } F\}$$

Selection (σ)

- More formally:
  - Let *F* be a boolean formula. Comparisons like {=, <, >, ≤, ≥, ≠} with boolean operators like {∧, ∨}.
  - Let *r* be a relation with relation scheme $R(A_1, A_2, \ldots, A_n)$.
  - Then,

$$\sigma_F(r) = \{t : t \in r \text{ and } t \text{ satisfies } F\}$$

What are the attributes in the relation scheme for the resulting relation?

Selection (σ)

- More formally:
  - Let *F* be a boolean formula. Comparisons like {=, <, >, ≤, ≥, ≠} with boolean operators like {∧, ∨}.
  - Let *r* be a relation with relation scheme $R(A_1, A_2, \ldots, A_n)$.
  - Then,

$$\sigma_F(r) = \{t : t \in r \text{ and } t \text{ satisfies } F\}$$

What are the attributes in the relation scheme for the resulting relation?

The resulting relation will be defined over attributes $A_1, A_2, \ldots, A_n$.

Boolean Formulas

- A boolean formula *F* is a combination of atoms. For example:
  - $(\text{atom}_1 \vee \text{atom}_2) \wedge \text{atom}_3$
- Each atom is:
  - Comparison of attributes: $A_i \, \theta \, A_j$
  - Comparison of attributes and constants: $A_i \, \theta \, c$, where *c* is a constant member of the domain of A_i.
  - The comparisons θ can be any of the comparison operators like {=, <, >, ≤, ≥, ≠}

Composition of Operators

- Used for combining the operations
- Give me the names of suppliers that are located in Chicago.

Composition of Operators

- Give me the names of suppliers that are located in Chicago.
- $\Pi_{\text{Supplier-name}}(\sigma_{\text{City='Chicago'}}(\textit{supplier}))$

supplier:

Supplier-id	Supplier-name	City
123	ACME	Chicago
234	BBQ Inc.	New York
345	COLE	London
456	Dave's	Chicago

$\sigma_{\text{City='Chicago'}}(\textit{supplier})$

Supplier-id	Supplier-name	City
123	ACME	Chicago
456	Dave's	Chicago

$\Pi_{\text{Supplier-name}}(\sigma_{\text{City='Chicago'}}(\textit{supplier}))$

Supplier-name
ACME
Dave's

Difference (–)

- Binary operation in which given two relations $r(R)$ and $s(R)$ defined on the same schema R, then $(r - s)$ is also a relation on R that includes all tuples in r that are not in s.
- Formally: $r - s = \{t : t \in r, t \notin s\}$

Difference (–)

- Binary operation in which given two relations $r(R)$ and $s(R)$ defined on the same schema R, then $(r - s)$ is also a relation on R that includes all tuples in r that are not in s.
- Formally: $r - s = \{t : t \in r, t \notin s\}$

What are the attributes in the relation scheme for the resulting relation?

Difference (–)

- Binary operation in which given two relations $r(R)$ and $s(R)$ defined on the same schema R, then $(r - s)$ is also a relation on R that includes all tuples in r that are not in s.
- Formally: $r - s = \{t : t \in r, t \notin s\}$

What are the attributes in the relation scheme for the resulting relation?

The resulting relation will be defined over attributes all attributes of R.

Difference (–)

- Give me the suppliers for company A that are not suppliers for company B.

Suppliers for company A:

Supplier-id	Supplier-name	City
123	ACME	Chicago
234	BBQ Inc.	New York
345	COLE	London
456	Dave's	Chicago

Suppliers for company B:

Supplier-id	Supplier-name	City
123	ACME	Chicago
567	Emory	New York
345	COLE	London
678	Fatulo	Chicago

Difference (–)

- Give me the suppliers for company A that are not suppliers for company B.

Suppliers for company A:

Supplier-id	Supplier-name	City
123	ACME	Chicago
234	**BBQ Inc.**	**New York**
345	COLE	London
456	**Dave's**	**Chicago**

Suppliers for company B:

Supplier-id	Supplier-name	City
123	ACME	Chicago
567	Emory	New York
345	COLE	London
678	Fatulo	Chicago

Difference (–)

- Give me the suppliers for company A that are not suppliers for company B.

A – B

Supplier-id	Supplier-name	City
234	BBQ Inc.	New York
456	Dave's	Chicago

Union ($\cup$)

- Binary operator where given two relations $r(R)$ and $s(R)$ defined on the same schema R, returns a relation on schema R that contains any tuple that is in r or s.
- Formally: $r \cup s = \{t : t \in r \text{ or } t \in s\}$

Union ($\cup$)

- Binary operator where given two relations $r(R)$ and $s(R)$ defined on the same schema R, returns a relation on schema R that contains any tuple that is in r or s.
- Formally: $r \cup s = \{t : t \in r \text{ or } t \in s\}$

What are the attributes in the relation scheme for the resulting relation?

Union ($\cup$)

- Binary operator where given two relations $r(R)$ and $s(R)$ defined on the same schema R, returns a relation on schema R that contains any tuple that is in r or s.
- Formally: $r \cup s = \{t : t \in r \text{ or } t \in s\}$

What are the attributes in the relation scheme for the resulting relation?

The resulting relation will be defined over all attributes of R.

Union (∪)

- Give me all suppliers, including those of company A or company B.

Suppliers for company A:

Supplier-id	Supplier-name	City
123	ACME	Chicago
234	BBQ Inc.	New York
345	COLE	London
456	Dave's	Chicago

Suppliers for company B:

Supplier-id	Supplier-name	City
123	ACME	Chicago
567	Emory	New York
345	COLE	London
678	Fatulo	Chicago

Union (∪)

- Give me all suppliers, including those of company A or company B.

Suppliers for company A:

Supplier-id	Supplier-name	City
123	**ACME**	**Chicago**
234	**BBQ Inc.**	**New York**
345	**COLE**	**London**
456	**Dave's**	**Chicago**

Suppliers for company B:

Supplier-id	Supplier-name	City
123	ACME	Chicago
567	**Emory**	**New York**
345	COLE	London
678	**Fatulo**	**Chicago**

Union (∪)

- Give me all suppliers, including those of company A or company B.

A ∪ B

Supplier-id	Supplier-name	City
123	ACME	Chicago
234	BBQ Inc.	New York
345	COLE	London
456	Dave's	Chicago
567	Emory	New York
678	Fatulo	Chicago

Cartesian Product (×)

- Let $R(A_1,\ldots,A_n)$ and $S(B_1,\ldots,B_m)$ be two schemas.
- Let $r(R)$ and $s(S)$ be relations defined on schemas R and S respectively.
- Then $r \times s$ is a relation defined on the schema $T(R.A_1,\ldots,R.A_n,S.B_1,\ldots,S.B_m)$.
- $r \times s$ contains a tuple t for each pair of tuples, t_1 and t_2, such that $t_1 \in r$ and $t_2 \in s$.

Cartesian Product (×)

- Retrieve all pairs of parts, suppliers. For each part its potential supplier.
- Supplier(Supplier-id,Supplier-name,City)
  Part (Part-id,Part-name,Color,Weight)

Supplier-id	Supplier-name	City
123	ACME	Chicago
234	BBQ Inc.	New York
345	COLE	London
456	Dave's	Chicago

Part-id	Part-name	Color	Weight
12	Nut	Gray	12
15	Bolt	Orange	17
5	Screw	Blue	17

- Supplier × Part

Supplier-id	Supplier-name	City	Part-id	Part-name	Color	Weight
123	ACME	Chicago	12	Nut	Gray	12
123	ACME	Chicago	15	Bolt	Orange	17
123	ACME	Chicago	5	Screw	Blue	17
234	BBQ Inc.	New York	12	Nut	Gray	12
234	BBQ Inc.	New York	15	Bolt	Orange	17
234	BBQ Inc.	New York	5	Screw	Blue	17
345	COLE	London	12	Nut	Gray	12
345	COLE	London	15	Bolt	Orange	17
345	COLE	London	5	Screw	Blue	17
456	Dave's	Chicago	12	Nut	Gray	12
456	Dave's	Chicago	15	Bolt	Orange	17
456	Dave's	Chicago	5	Screw	Blue	17

Cartesian Product (×)

- More formally:

$$r \times s = \{t : t(R.A_1, \ldots, R.A_n) \in r \text{ and } t(S.B_1, \ldots, S.B_m) \in s\}$$

Cartesian Product (×)

- More formally:

$$r \times s = \{t : t(R.A_1, \ldots, R.A_n) \in r \text{ and } t(S.B_1, \ldots, S.B_m) \in s\}$$

What are the attributes in the relation scheme for the resulting relation?

Cartesian Product (×)

- More formally:

$$r \times s = \{t : t(R.A_1, \ldots, R.A_n) \in r \text{ and } t(S.B_1, \ldots, S.B_m) \in s\}$$

What are the attributes in the relation scheme for the resulting relation?

The resulting relation will be defined over attributes $R.A_1, R.A_2, \ldots, R.A_n, S.B_1, \ldots, S.B_m$.

Relational Algebra Operations

- *Order* of tuples in a relation is irrelevant for all operations.
- Properties of some fundamental operators:
  - $r \cup s = s \cup r$
  - $r - s \neq s - r$
  - $r \times s = s \times r$
  - $(r \cup s) \cup v = r \cup (s \cup v)$
  - $(r - s) - v \neq r - (s - v)$

Relational Algebra

- The same query may have different Relational Algebra expressions.
- Example: Retrieve parts that are ‘green’ or weigh more than 15.
  - $\sigma_{Color=\text{‘green’} \vee Weight>15}(Part)$
  - What is an alternative Relational Algebra expression?

Relational Algebra

- The same query may have different Relational Algebra expressions.
- Example: Retrieve parts that are ‘green’ or weigh more than 15.
  - $\sigma_{Color=\text{‘green’} \vee Weight>15}(Part)$
  - $\sigma_{Color=\text{‘green’}}(Part) \cup \sigma_{Weight>15}(Part)$

Composite Operators

- These will be relational algebra operations that can be expressed equivalently by using the fundamental operators we've seen already.
- Examples:
  - Intersection, $\cap$
  - Natural Join, $\bowtie$
  - Division, $\div$

Intersection ($\cap$)

- Binary operator where given two relations $r(R)$ and $s(R)$ defined on the same schema R, **returns a relation on schema *R*** that contains any tuple that is in ***r* and *s***.
- Formally: $r \cap s = \{t : t \in r \text{ and } t \in s\}$
- Also, $r \cap s = r - (r - s)$
- Like union, intersection is also commutative.

Intersection (∩)

- Give me all suppliers that supply to both company A and B.

Suppliers for company A:

Supplier-id	Supplier-name	City
123	ACME	Chicago
234	BBQ Inc.	New York
345	COLE	London
456	Dave's	Chicago

Suppliers for company B:

Supplier-id	Supplier-name	City
123	ACME	Chicago
567	Emory	New York
345	COLE	London
678	Fatulo	Chicago

Intersection (∩)

- Give me all suppliers that supply to both company A and B.

Suppliers for company A:

Supplier-id	Supplier-name	City
123	**ACME**	**Chicago**
234	BBQ Inc.	New York
345	**COLE**	**London**
456	Dave's	Chicago

Suppliers for company B:

Supplier-id	Supplier-name	City
123	**ACME**	**Chicago**
567	Emory	New York
345	**COLE**	**London**
678	Fatulo	Chicago

Intersection (∩)

- Give me all suppliers that supply to both company A and B.

A ∩ B

Supplier-id	Supplier-name	City
123	ACME	Chicago
345	COLE	London

Join ($\bowtie_\theta$)

- Also known as theta join.
- Binary operation that allows us to combine a selection and a Cartesian product.
- Syntax: $r \bowtie_\theta s$, where θ is a boolean formula
- Example: For every student, retrieve the enrollment records in which the grade is higher than their GPA.

Join ($\bowtie_\theta$)

- Example: For every student, retrieve the enrollment records in which the grade is higher than their GPA.

Student:

Student-id	Name	GPA
23	Alice	B
25	Bob	B
28	Charlie	A

Enrollment:

Student-id	Course#	Grade
23	480	A
23	580	A
25	230	B+
25	250	C
28	480	A
28	230	A
28	580	A+

Join ($\bowtie_\theta$)

- Example: For every student, retrieve the enrollment records in which the grade is higher than their GPA.

Student $\bowtie_{\text{Student.GPA<Enrollment.grade} \wedge \text{Student.student-id=Enrollment.student-id}}$ Enrollment

Student. student-id	Name	GPA	Enrollment. student-id	Course#	Grade
23	Alice	B	23	480	A
23	Alice	B	23	580	A
25	Bob	B	25	230	B+
28	Charlie	A	28	580	A+

Join ($\bowtie_\theta$)

- More formally:

$r \bowtie_\theta s = \{t : t(R.A_1, \ldots, R.A_n) \in R$ and
$t(S.B_1, \ldots, S.B_m) \in S$ and t
satisfies the formula $\theta\}$

- $r \bowtie_\theta s = \sigma_\theta(r \times s)$

Join ($\bowtie_\theta$)

- More formally:

$$r \bowtie_\theta s = \{t : t(R.A_1, \ldots, R.A_n) \in R \text{ and } t(S.B_1, \ldots, S.B_m) \in S \text{ and } t \text{ satisfies the formula } \theta\}$$

- $r \bowtie_\theta s = \sigma_\theta(r \times s)$

What are the attributes in the relation scheme for the resulting relation?

Join ($\bowtie_\theta$)

- More formally:

$r \bowtie_\theta s = \{t : t(R.A_1, \ldots, R.A_n) \in R$ and
$t(S.B_1, \ldots, S.B_m) \in S$ and t
satisfies the formula $\theta\}$

- $r \bowtie_\theta s = \sigma_\theta(r \times s)$

What are the attributes in the relation scheme for the resulting relation?

The resulting relation will be defined over attributes
$R.A_1, R.A_2, \ldots, R.A_n, S.B1, \ldots, S.B_m$.

Join ($\bowtie_\theta$)

Student $\bowtie_{\text{Student.GPA<Enrollment.grade} \wedge \text{ Student.student-id=Enrollment.student-id}}$ Enrollment

Student. student-id	Name	GPA	Enrollment. student-id	Course#	Grade
23	Alice	B	23	480	A
23	Alice	B	23	580	A
25	Bob	B	25	230	B+
28	Charlie	A	28	580	A+

Is there any problem with this result?

Natural Join ($\bowtie$)

- Binary operation that joins two relations by their common attributes.
- Let $R = \{A_1, A_2, \ldots, A_n\}$ and $S = \{B_1, B_2, \ldots, B_m\}$ and $r(R)$ and $s(S)$ be relations based on those attribute sets respectively.
- ***r*** $\bowtie$ ***s* is a relation defined on attribute set** $\boldsymbol{R \cup S}$.
- Returns the joined tuples in which the common attributes of R and S are equal.

Natural Join (⋈)

- Example: For every student retrieve its enrollment records.

Student:

Student-id	Name	GPA
23	Alice	B
25	Bob	B
28	Charlie	A

Enrollment:

Student-id	Course#	Grade
23	480	A
23	580	A
25	230	B+
25	250	C
28	480	A
28	230	A
28	580	A+

Common attribute is **Student-id**

Natural Join ($\bowtie$)

- Example: For every student retrieve its enrollment records.

Student $\bowtie$ Enrollment

Student-id	Name	GPA	Course#	Grade
23	Alice	B	480	A
23	Alice	B	580	A
25	Bob	B	230	B+
25	Bob	B	250	C
28	Charlie	A	480	A
28	Charlie	A	230	A
28	Charlie	A	580	A+

Natural Join ($\bowtie$)

- Formally:

$r \bowtie s = \{t : t(R) \in r$ and $t(S) \in s$ and
t has scheme $R \cup S\}$

- Let $R \cap S = \{C_1, C_2, \ldots, C_k\}$, then:

$$r \bowtie s = \Pi_{R \cup S}(\sigma_{R.C_1 = S.C_1 \wedge R.C_2 = S.C_2 \wedge \cdots \wedge R.C_k = S.C_k}(r \times s))$$

- Also,

$$r \bowtie s = \Pi_{R \cup S}(r \bowtie_{R.C_1 = S.C_1 \wedge R.C_2 = S.C_2 \wedge \cdots \wedge R.C_k = S.C_k} s)$$

Natural Join ($\bowtie$)

- Some special properties of the natural join:

$$r \bowtie r = r$$

$$r \bowtie (s \bowtie v) = (r \bowtie s) \bowtie v$$

$$r \bowtie s = s \bowtie r$$

Natural Join ($\bowtie$)

- What do we get if we project the result over the attributes of Student?

Student $\bowtie$ Enrollment

Student-id	Name	GPA	Course#	Grade
23	**Alice**	**B**	480	A
23	**Alice**	**B**	580	A
25	**Bob**	**B**	230	B+
25	**Bob**	**B**	250	C
28	**Charlie**	**A**	480	A
28	**Charlie**	**A**	230	A
28	**Charlie**	**A**	580	A+

Natural Join ($\bowtie$)

- What do we get if we project the result over the attributes of Student?
- We won't necessarily get Student back.

Natural Join (⋈)

- What do we get if we project the result over the attributes of Student?
- We won't necessarily get Student back.

Student′ :

Student-id	Name	GPA
23	Alice	B
25	Bob	B
28	Charlie	A
30	Dan	C

What will be the result now?

Enrollment:

Student-id	Course#	Grade
23	480	A
23	580	A
25	230	B+
25	250	C
28	480	A
28	230	A
28	580	A+

Natural Join ($\bowtie$)

- What do we get if we project the result over the attributes of Student?
- We won't necessarily get Student back.

Student′ $\bowtie$ Enrollment

Student-id	Name	GPA	Course#	Grade
23	**Alice**	**B**	480	A
23	**Alice**	**B**	580	A
25	**Bob**	**B**	230	B+
25	**Bob**	**B**	250	C
28	**Charlie**	**A**	480	A
28	**Charlie**	**A**	230	A
28	**Charlie**	**A**	580	A+

Natural Join ($\bowtie$)

- Let r be defined over $R(A_1,A_2,\ldots,A_n)$ and s be define over $S(B_1,B_2,\ldots,B_m)$.
- For the Cartesian Product:
  - $\Pi_R(r \times s) = r$
- For the Natural Join
  - $\Pi_R(r \bowtie s) \neq r$
  - $\Pi_R(r \bowtie s) \subseteq r$

Natural Join ($\bowtie$)

- Theorem: $\Pi_R(r \bowtie s) \subseteq r$

Natural Join ($\bowtie$)

- Theorem: $\Pi_R(r \bowtie s) \subseteq r$
- Proof:
  - Let $t \in \Pi_R(r \bowtie s)$

Natural Join ($\bowtie$)

- Theorem: $\Pi_R(r \bowtie s) \subseteq r$
- Proof:
  - Let $t \in \Pi_R(r \bowtie s)$
  - Then $\exists$ a $t' \in r \bowtie s$ such that $t'(R) = t$ (by def. of Π)

Natural Join ($\bowtie$)

- Theorem: $\Pi_R(r \bowtie s) \subseteq r$
- Proof:
  - Let $t \in \Pi_R(r \bowtie s)$
  - Then $\exists$ a $t' \in r \bowtie s$ such that $t'(R) = t$ (by def. of Π)
  - Also, $t'(\mathrm{R}) \in r$ (by def. of $\bowtie$)

Natural Join ($\bowtie$)

- Theorem: $\Pi_R(r \bowtie s) \subseteq r$
- Proof:
  - Let $t \in \Pi_R(r \bowtie s)$
  - Then $\exists$ a $t' \in r \bowtie s$ such that $t'(R) = t$ (by def. of Π)
  - Also, $t'(\mathrm{R}) \in r$ (by def. of $\bowtie$)
  - Then $t \in r$.

Division (÷)

- Let $r(R)$ and $s(S)$ be relations such that $S \subseteq R$.
- Then $r \div s$ consists of every tuple t in $\Pi_{R-S}(r)$ that satisfies the following condition: If every tuple in s is appended to t, then the resulting tuple is in r.
- **The result is defined over the scheme *R-S.***

Division (÷)

Numerator:

Supplier-id	Part-id
1	1
1	2
1	3
1	4
1	5
1	6
2	1
2	2
3	2
4	2
4	4
4	5

Denominator:

Part-id
1

Intuitively: Numerator ÷ Denominator

Should return the suppliers that supply all parts in the denominator.

Division (÷)

Numerator:

Supplier-id	Part-id
1	1
1	2
1	3
1	4
1	5
1	6
2	1
2	2
3	2
4	2
4	4
4	5

Denominator:

Part-id
1

Intuitively: Numerator ÷ Denominator

Should return the suppliers that supply all parts in the denominator.

What is the result?

Division (÷)

Numerator:

Supplier-id	Part-id
1	1
1	2
1	3
1	4
1	5
1	6
2	1
2	2
3	2
4	2
4	4
4	5

Denominator:

Part-id
1

Result:

Supplier-id
1
2

Division (÷)

Numerator:

Supplier-id	Part-id
1	1
1	2
1	3
1	4
1	5
1	6
2	1
2	2
3	2
4	2
4	4
4	5

Denominator:

Part-id
2
4

Intuitively: Numerator ÷ Denominator

Should return the suppliers that supply all parts in the denominator.

What is the result?

Division (÷)

Numerator:

Supplier-id	Part-id
1	1
1	2
1	3
1	4
1	5
1	6
2	1
2	2
3	2
4	2
4	4
4	5

Denominator:

Part-id
2
4

Result:

Supplier-id
1
4

Division (÷)

Numerator:

Supplier-id	Part-id
1	1
1	2
1	3
1	4
1	5
1	6
2	1
2	2
3	2
4	2
4	4
4	5

Denominator:

Part-id
1
2
3
4
5
6

Intuitively: Numerator ÷ Denominator

Should return the suppliers that supply all parts in the denominator.

What is the result?

Division (÷)

Numerator:

Supplier-id	Part-id
1	1
1	2
1	3
1	4
1	5
1	6
2	1
2	2
3	2
4	2
4	4
4	5

Denominator:

Part-id
1
2
3
4
5
6

Result:

Supplier-id
1

Division (÷)

- What are the student-id's of students who took both 480 and 580, and got an A in both?

Enrollment:

Student-id	Course#	Grade
23	480	A
23	580	A
25	230	B+
25	250	C
28	480	A
28	230	A
28	580	A

We want to compute: Enrollment ÷ *s* to answer this query.

What should *s* be?

Division (÷)

- What are the student-id's of students who took both 480 and 580, and got an A in both?

Enrollment:

Student-id	Course#	Grade
23	480	A
23	580	A
25	230	B+
25	250	C
28	480	A
28	230	A
28	580	A

s:

Course#	Grade
480	A
580	A

Division (÷)

- What are the student-id's of students who took both 480 and 580, and got an A in both?

Enrollment:

Student-id	Course#	Grade
23	480	A
23	580	A
25	230	B+
25	250	C
28	480	A
28	230	A
28	580	A

s:

Course#	Grade
480	A
580	A

Enrollment ÷ *s*

Student-id
23
28

Division (÷)

- What are the student-id's of students who took both 480 and 580, and got an A in both?

Enrollment:

Student-id	Course#	Grade
23	480	A
23	580	A
25	230	B+
25	250	C
28	480	A
28	230	A
28	580	A

What's an alternative algebra expression that would answer this query without division?

Division (÷)

- Formally:

$r \div s = \{t : \forall u \in s, \exists v \in r$ such that

$v(R–S) = t$ and

$v(S) = u\}$

- $r \div s$ is defined over attributes $R–S$

Division (÷)

- Formally:

$r \div s = \{t : \forall u \in s, \exists v \in r$ such that

$v(R–S) = t$ and

$v(S) = u\}$

- $r \div s$ is defined over attributes $R–S$
- Also,

$r \div s = \Pi_{R–S}(r) – \Pi_{R–S}((\Pi_{R–S}(r) \times s) – r)$

Division (÷)

- Formally:

$r \div s = \{t : \forall u \in s, \exists v \in r$ such that

$v(R–S) = t$ and

$v(S) = u\}$

- $r \div s$ is defined over attributes $R–S$
- Also,

$r \div s = \Pi_{R–S}(r) – \mathbf{\Pi_{R–S}((\Pi_{R–S}(r) \times s) – r)}$

The set of tuples t for which there is a tuple $u \in s$ such that $t \cdot u \notin r$

Division

- Special Remarks
  - $(r \times s) \div s = r$
  - $r \div s$ is the maximal subset $d \subseteq \Pi_{R\text{-}S}(r)$ such that $d \times s \subseteq r$

Examples #1

- Relational Schema
  - order(order#,retailer-name,item,quantity)
  - supplier(supp#,supp-name,item,price)

Examples #1

- Relational Schema
  - order(order#,retailer-name,item,quantity)
  - supplier(supp#,supp-name,item,price)
- Query
  - Retrieve the name of suppliers that supply every item that was endorsed by ‘Jewel’

Examples #1

- Relational Schema
  - order(order#,retailer-name,item,quantity)
  - supplier(supp#,supp-name,item,price)
- Query
  - Retrieve the name of suppliers that supply every item that was endorsed by ‘Jewel’
- Relational Algebra Query

$$\Pi_{\text{supp-name,item}}(X) \div \Pi_{\text{item}}(Y)$$

Examples #1

- Relational Schema
  - order(order#,retailer-name,item,quantity)
  - supplier(supp#,supp-name,item,price)
- Query
  - Retrieve the name of suppliers that supply **every item** that was endorsed by ‘Jewel’
- Relational Algebra Query
- $\Pi_{\text{supp-name,item}}(\text{supplier}) \div \Pi_{\text{item}}(\sigma_{\text{retailer-name='Jewel'}}(\text{order}))$

Examples #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)

Examples #2, Query #1

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #1: Retrieve the names of suppliers that have shipments of part #2.
- Steps to take?

Examples #2, Query #1

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #1: Retrieve the names of suppliers that have shipments of part #2.
- Steps to take?
  - Select the shipments of part #2.

Examples #2, Query #1

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #1: Retrieve the names of suppliers that have shipments of part #2.
- Steps to take?
  - Select the shipments of part #2.
  - Link these shipments with the supplier information (to get the supplier names).

Examples #2, Query #1

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #1: Retrieve the names of suppliers that have shipments of part #2.
- Steps to take?
  - Select the shipments of part #2.
  - Link these shipments with the supplier information (to get the supplier names).
  - Extract the supplier names.

Examples #2, Query #1

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #1: Retrieve the names of suppliers that have shipments of part #2.

Step 1: Select the shipments of Part #2.

Examples #2, Query #1

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #1: Retrieve the names of suppliers that have shipments of part #2.

Step 1: Select the shipments of Part #2.

$$\sigma_{\text{part-id}=\text{'2'}}(\text{supplies})$$

Examples #2, Query #1

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #1: Retrieve the names of suppliers that have shipments of part #2.

Step 2: Link these shipments with the supplier information (to get the supplier names).

$$\sigma_{\text{part-id}=\text{'2'}}(\text{supplies})$$

Examples #2, Query #1

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #1: Retrieve the names of suppliers that have shipments of part #2.

Step 2: Link these shipments with the supplier information (to get the supplier names).

$$\sigma_{\text{part-id='2'}}(\text{supplies}) \bowtie \text{supplier}$$

Examples #2, Query #1

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #1: Retrieve the names of suppliers that have shipments of part #2.

Step 3: Extract the supplier names.

$$\sigma_{\text{part-id='2'}}(\text{supplies}) \bowtie \text{supplier}$$

Examples #2, Query #1

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #1: Retrieve the names of suppliers that have shipments of part #2.

Step 3: Extract the supplier names.

$$\Pi_{\text{supp-name}}(\sigma_{\text{part-id='2'}}(\text{supplies}) \bowtie \text{supplier})$$

Examples #2, Query #1

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #1: Retrieve the names of suppliers that have shipments of part #2.

$$\Pi_{\text{supp-name}}(\sigma_{\text{part-id='2'}}(\text{supplies}) \bowtie \text{supplier})$$

Examples #2, Query #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #2: Retrieve the names of suppliers that have shipments of at least one ‘red’ part.
- Steps to take?

Examples #2, Query #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #2: Retrieve the names of suppliers that have shipments of at least one ‘red’ part.
- Steps to take?
  - Extract all parts that are red.

Examples #2, Query #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #2: Retrieve the names of suppliers that have shipments of at least one ‘red’ part.
- Steps to take?
  - Extract all parts that are red.
  - Get the part numbers for those red parts.

Examples #2, Query #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #2: Retrieve the names of suppliers that have shipments of at least one ‘red’ part.
- Steps to take?
  - Extract all parts that are red.
  - Get the part numbers for those red parts.
  - Extract all shipments of those red parts.

Examples #2, Query #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #2: Retrieve the names of suppliers that have shipments of at least one ‘red’ part.
- Steps to take?
  - Extract all parts that are red.
  - Get the part numbers for those red parts.
  - Extract all shipments of those red parts.
  - Get the suppliers of those shipments.

Examples #2, Query #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #2: Retrieve the names of suppliers that have shipments of at least one ‘red’ part.
- Steps to take?
  - Extract all parts that are red.
  - Get the part numbers for those red parts.
  - Extract all shipments of those red parts.
  - Get the suppliers of those shipments.
  - Get the supplier names of those suppliers.

Examples #2, Query #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #2: Retrieve the names of suppliers that have shipments of at least one ‘red’ part.

Step 1: Extract all parts that are red.

Examples #2, Query #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #2: Retrieve the names of suppliers that have shipments of at least one ‘red’ part.

Step 1: Extract all parts that are red.

$$\sigma_{color=\text{‘red’}}(parts)$$

Examples #2, Query #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #2: Retrieve the names of suppliers that have shipments of at least one ‘red’ part.

Step 2: Get the part numbers for those red parts.

$$\sigma_{color=\text{‘red’}}(parts)$$

Examples #2, Query #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #2: Retrieve the names of suppliers that have shipments of at least one ‘red’ part.

Step 2: Get the part numbers for those red parts.

$$\Pi_{\text{part-id}}(\sigma_{\text{color='red'}}(\text{parts}))$$

Examples #2, Query #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #2: Retrieve the names of suppliers that have shipments of at least one ‘red’ part.

Step 3: Extract all shipments of those red parts.

$$\Pi_{\text{part-id}}(\sigma_{\text{color='red'}}(\text{parts}))$$

Examples #2, Query #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #2: Retrieve the names of suppliers that have shipments of at least one ‘red’ part.

Step 3: Extract all shipments of those red parts.

$$\Pi_{\text{part-id}}(\sigma_{\text{color='red'}}(\text{parts})) \bowtie \text{supplies}$$

Examples #2, Query #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #2: Retrieve the names of suppliers that have shipments of at least one ‘red’ part.

Step 4: Get the suppliers of those shipments.

$$\Pi_{\text{part-id}}(\sigma_{\text{color='red'}}(\text{parts})) \bowtie \text{supplies}$$

Examples #2, Query #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #2: Retrieve the names of suppliers that have shipments of at least one ‘red’ part.

Step 4: Get the suppliers of those shipments.

$$\text{supplier} \bowtie \Pi_{\text{supp-id}}(\Pi_{\text{part-id}}(\sigma_{\text{color='red'}}(\text{parts})) \bowtie \text{supplies})$$

Examples #2, Query #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #2: Retrieve the names of suppliers that have shipments of at least one ‘red’ part.

Step 5: Get the supplier names of those suppliers.

$$\Pi_{\text{supp-name}}(\text{supplier} \bowtie \Pi_{\text{supp-id}}(\Pi_{\text{part-id}}(\sigma_{\text{color='red'}}(\text{parts})) \bowtie \text{supplies}))$$

Examples #2, Query #2

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #2: Retrieve the names of suppliers that have shipments of at least one ‘red’ part.

$$\Pi_{\text{supp-name}}(\text{supplier} \bowtie \Pi_{\text{supp-id}}(\Pi_{\text{part-id}}(\sigma_{\text{color='red'}}(\text{parts})) \bowtie \text{supplies}))$$

Examples #2, Query #3

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #3: Give the names of suppliers that have shipments of all the parts.
- Steps to take?

Examples #2, Query #3

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #3: Give the names of suppliers that have shipments of all the parts.
- Steps to take?
  - Division between the shipments and all the part-id's (this will give us supplier-id's of such suppliers).

Examples #2, Query #3

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #3: Give the names of suppliers that have shipments of all the parts.
- Steps to take?
  - Division between the shipments and all the part-id's (this will give us supplier-id's of such suppliers).
  - Use those supplier-id's to get their names.

Examples #2, Query #3

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #3: Give the names of suppliers that have shipments of all the parts.

Step 1: Division between the shipments and all the part-id's (this will give us supplier-id's of such suppliers).

Examples #2, Query #3

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #3: Give the names of suppliers that have shipments of all the parts.

Step 1: Division between the shipments and all the part-id's (this will give us supplier-id's of such suppliers).

$$\Pi_{\text{supp-id,part-id}}(\text{supplies}) \div \Pi_{\text{part-id}}(\text{part})$$

Examples #2, Query #3

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #3: Give the names of suppliers that have shipments of all the parts.

Step 2: Use those supplier-id's to get their names.

$$(\Pi_{\text{supp-id,part-id}}(\text{supplies}) \div \Pi_{\text{part-id}}(\text{part}))$$

Examples #2, Query #3

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #3: Give the names of suppliers that have shipments of all the parts.

Step 2: Use those supplier-id's to get their names.

$$(\Pi_{\text{supp-id,part-id}}(\text{supplies}) \div \Pi_{\text{part-id}}(\text{part})) \bowtie \text{supplier}$$

Examples #2, Query #3

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #3: Give the names of suppliers that have shipments of all the parts.

Step 2: Use those supplier-id's to get their names.

$$\Pi_{\text{supp-name}}((\Pi_{\text{supp-id,part-id}}(\text{supplies}) \div \Pi_{\text{part-id}}(\text{part})) \bowtie \text{supplier})$$

Examples #2, Query #3

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #3: Give the names of suppliers that have shipments of all the parts.

$$\Pi_{\text{supp-name}}((\Pi_{\text{supp-id,part-id}}(\text{supplies}) \div \Pi_{\text{part-id}}(\text{part})) \bowtie \text{supplier})$$

Examples #2, Query #4

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #4: Names of suppliers that do not have shipments of part #2.
- Steps to take?

Examples #2, Query #4

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #4: Names of suppliers that do not have shipments of part #2.
- Steps to take?
  - Extract all suppliers that do ship part #2.

Examples #2, Query #4

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #4: Names of suppliers that do not have shipments of part #2.
- Steps to take?
  - Extract all suppliers that do ship part #2.
  - Using those, compute the ones that do not ship part #2.

Examples #2, Query #4

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #4: Names of suppliers that do not have shipments of part #2.
- Steps to take?
  - Extract all suppliers that do ship part #2.
  - Using those, compute the ones that do not ship part #2.
  - From those suppliers, get their names.

Examples #2, Query #4

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #4: Names of suppliers that do not have shipments of part #2.

Step 1: Extract all suppliers that do ship part #2.

Examples #2, Query #4

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #4: Names of suppliers that do not have shipments of part #2.

Step 1: Extract all suppliers that do ship part #2.

$$\sigma_{\text{part-id}=\text{'2'}}(\text{supplies})$$

Examples #2, Query #4

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #4: Names of suppliers that do not have shipments of part #2.

Step 1: Extract all suppliers that do ship part #2.

$$\Pi_{\text{supp-id}}(\sigma_{\text{part-id}=\text{'2'}}(\text{supplies}))$$

Examples #2, Query #4

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #4: Names of suppliers that do not have shipments of part #2.

Step 2: Using those, compute the ones that do not ship part #2.

$$\Pi_{\text{supp-id}}(\sigma_{\text{part-id='2'}}(\text{supplies}))$$

Examples #2, Query #4

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #4: Names of suppliers that do not have shipments of part #2.

Step 2: Using those, compute the ones that do not ship part #2.

$$\Pi_{\text{supp-id}}(\text{supplier}) - \Pi_{\text{supp-id}}(\sigma_{\text{part-id}=\text{'2'}}(\text{supplies}))$$

Examples #2, Query #4

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #4: Names of suppliers that do not have shipments of part #2.

Step 3: From those suppliers, get their names.

$$\Pi_{\text{supp-id}}(\text{supplier}) - \Pi_{\text{supp-id}}(\sigma_{\text{part-id}=\text{'2'}}(\text{supplies}))$$

Examples #2, Query #4

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #4: Names of suppliers that do not have shipments of part #2.

Step 3: From those suppliers, get their names.

$$\text{supplier} \bowtie (\Pi_{\text{supp-id}}(\text{supplier}) - \Pi_{\text{supp-id}}(\sigma_{\text{part-id}=\text{'2'}}(\text{supplies})))$$

Examples #2, Query #4

- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #4: Names of suppliers that do not have shipments of part #2.

Step 3: From those suppliers, get their names.

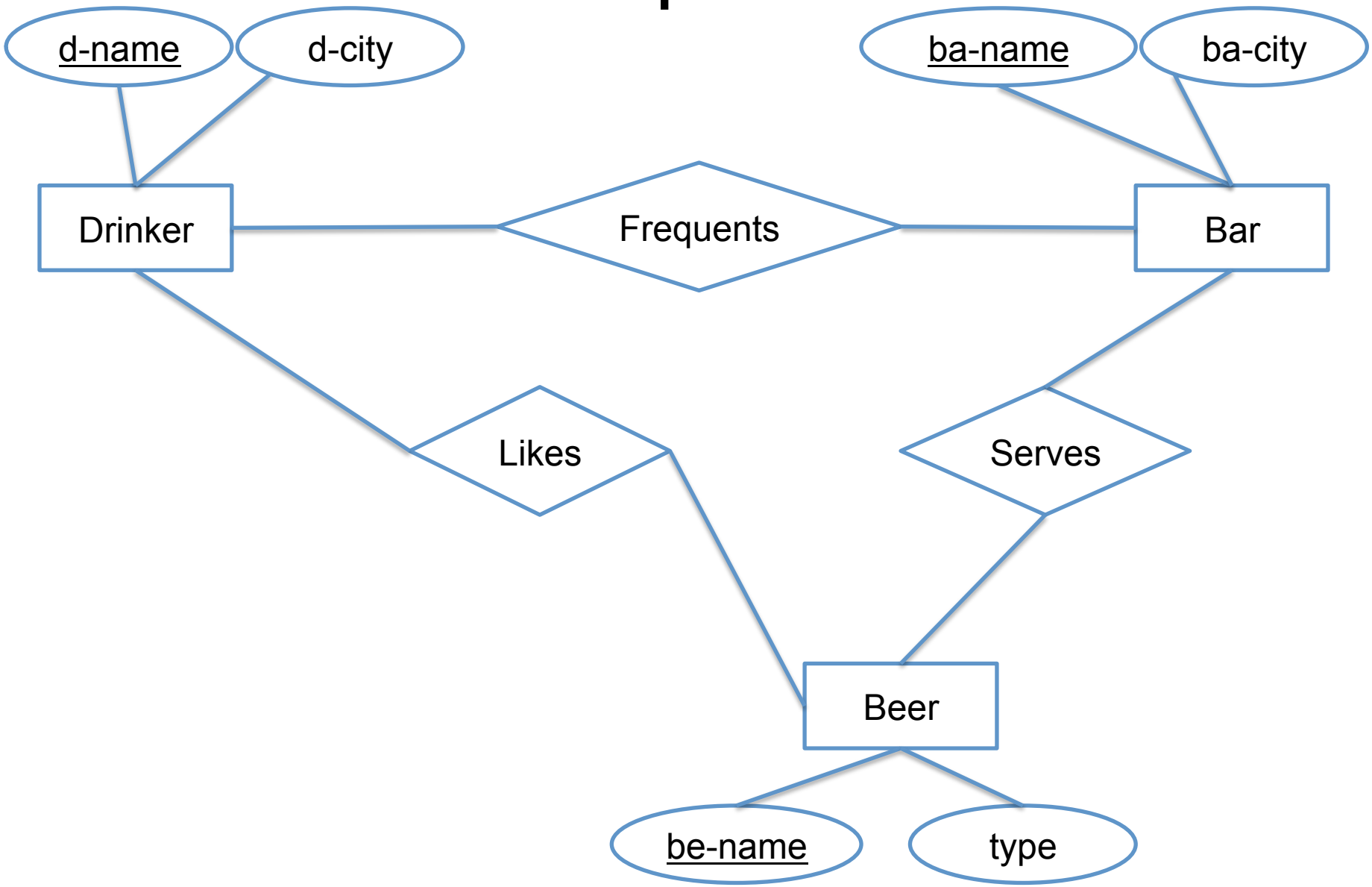
$$\Pi_{\text{supp-name}}(\text{supplier} \bowtie (\Pi_{\text{supp-id}}(\text{supplier}) - \Pi_{\text{supp-id}}(\sigma_{\text{part-id='2'}}(\text{supplies}))))$$

Examples #2, Query #4

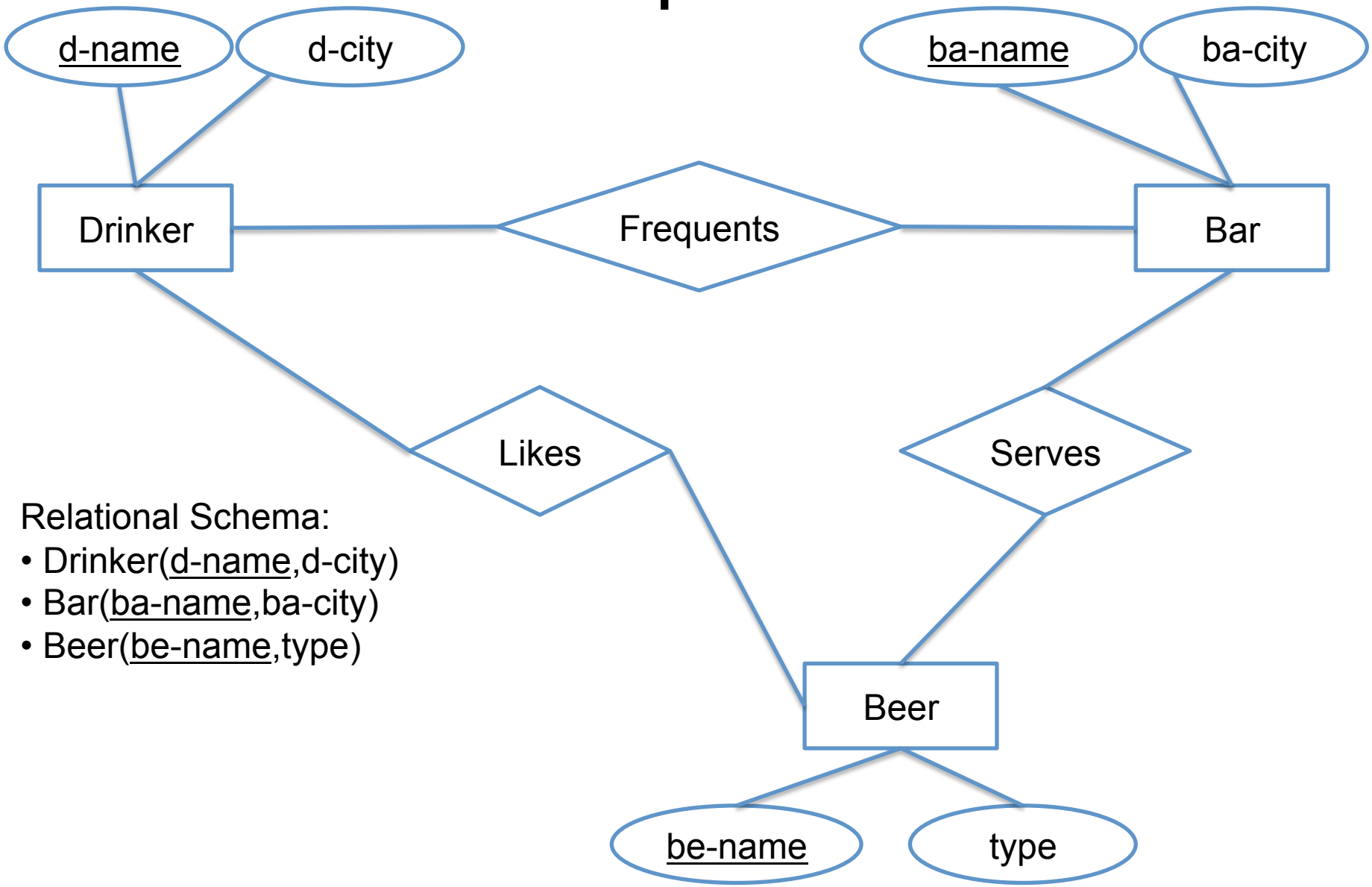
- Relational Schema
  - *Part*(*part-id, part-name, color*)
  - *Supplier*(*supp-id,supp-name,address*)
  - *Supplies*(*supp-id,part-id,quantity*)
- Query #4: Names of suppliers that do not have shipments of part #2.

$$\Pi_{\text{supp-name}}(\text{supplier} \bowtie (\Pi_{\text{supp-id}}(\text{supplier}) - \Pi_{\text{supp-id}}(\sigma_{\text{part-id='2'}}(\text{supplies}))))$$

Examples #3


d-name
d-city
ba-name
ba-city
Drinker
Frequents
Bar
Likes
Serves
Beer
be-name
type

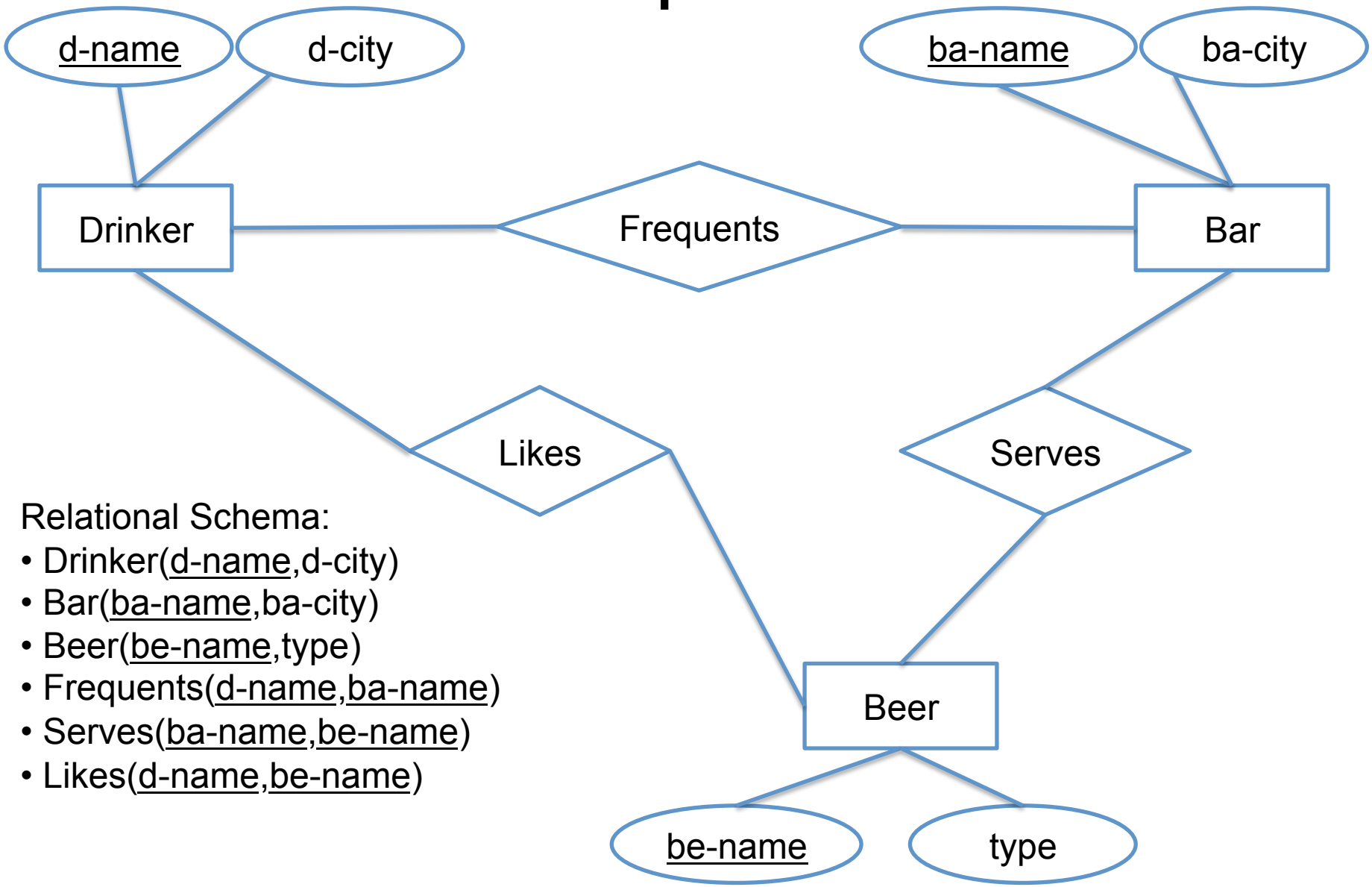
Examples #3



Relational Schema:

- Drinker(d-name,d-city)
- Bar(ba-name,ba-city)
- Beer(be-name,type)

Examples #3



Relational Schema:

- Drinker(d-name,d-city)
- Bar(ba-name,ba-city)
- Beer(be-name,type)
- Frequents(d-name,ba-name)
- Serves(ba-name,be-name)
- Likes(d-name,be-name)

Example #3, Query #1

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #1:

Which bars serve a beer that Joe likes?

Example #3, Query #1

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Steps to take?

Query #1:

Which bars serve a beer that Joe likes?

Example #3, Query #1

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #1:

Which bars serve a beer that Joe likes?

Steps to take?

1. Retrieve the beers that Joe likes.

Example #3, Query #1

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #1:

Which bars serve a beer that Joe likes?

Steps to take?

1. Retrieve the beers that Joe likes.
2. Find out which bars serve each of those beers.

Example #3, Query #1

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #1:

Which bars serve a beer that Joe likes?

Steps to take?

1. Retrieve the beers that Joe likes.
2. Find out which bars serve each of those beers.
3. Extract the bar names for those bars.

Example #3, Query #1

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #1:

Which bars serve a beer that Joe likes?

Step 1: Retrieve the beers that Joe likes.

Example #3, Query #1

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #1:

Which bars serve a beer that Joe likes?

Step 1: Retrieve the beers that Joe likes.

$$\sigma_{\text{d-name='Joe'}}(\text{likes})$$

Example #3, Query #1

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #1:

Which bars serve a beer that Joe likes?

Step 2: Find out which bars serve each of those beers.

$$\sigma_{\text{d-name='Joe'}}(\text{likes})$$

Example #3, Query #1

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #1:

Which bars serve a beer that Joe likes?

Step 2: Find out which bars serve each of those beers.

$$\sigma_{\text{d-name='Joe'}}(\text{likes}) \bowtie \text{serves}$$

Example #3, Query #1

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #1:

Which bars serve a beer that Joe likes?

Step 3: Extract the bar names for those bars.

$$\sigma_{\text{d-name='Joe'}}(\text{likes}) \bowtie \text{serves}$$

Example #3, Query #1

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #1:

Which bars serve a beer that Joe likes?

Step 3: Extract the bar names for those bars.

$$\Pi_{\text{ba-name}}(\sigma_{\text{d-name='Joe'}}(\text{likes}) \bowtie \text{serves})$$

Example #3, Query #1

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #1:

Which bars serve a beer that Joe likes?

$$\Pi_{\text{ba-name}}(\sigma_{\text{d-name='Joe'}}(\text{likes}) \bowtie \text{serves})$$

Example #3, Query #1

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #1:

Which bars serve a beer that Joe likes?

Are these two queries equivalent?

$$\Pi_{\text{ba-name}}(\sigma_{\text{d-name='Joe'}}(\text{likes} \bowtie \text{serves}))$$

$$\Pi_{\text{ba-name}}(\sigma_{\text{d-name='Joe'}}(\text{likes}) \bowtie \text{serves})$$

Example #3, Query #2

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #2:

Which drinkers frequent all the bars that serve ‘Bud’?

Example #3, Query #2

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Steps to take?

Query #2:

Which drinkers frequent all the bars that serve ‘Bud’?

Example #3, Query #2

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #2:

Which drinkers frequent all the bars that serve ‘Bud’?

Steps to take?

1. Compute all bars that serve Bud

Example #3, Query #2

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #2:

Which drinkers frequent all the bars that serve ‘Bud’?

Steps to take?

1. Compute all bars that serve Bud
2. Find out all drinkers that frequent all of those

Example #3, Query #2

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #2:

Which drinkers frequent all the bars that serve ‘Bud’?

Step 1: Compute all bars that serve Bud

Example #3, Query #2

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #2:

Which drinkers frequent all the bars that serve 'Bud'?

Step 1: Compute all bars that serve Bud

$$\sigma_{\text{be-name='Bud'}}(\text{serves})$$

Example #3, Query #2

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #2:

Which drinkers frequent all the bars that serve 'Bud'?

Step 1: Compute all bars that serve Bud

$$\Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves}))$$

Example #3, Query #2

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #2:

Which drinkers frequent all the bars that serve 'Bud'?

Step 2: Find out all drinkers that frequent all of those

$$\Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves}))$$

Example #3, Query #2

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #2:

Which drinkers frequent all the bars that serve 'Bud'?

Step 2: Find out all drinkers that frequent all of those

$$\text{frequents} \div \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves}))$$

Example #3, Query #2

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #2:

Which drinkers frequent all the bars that serve ‘Bud’?

$$\text{frequents} \div \Pi_{\text{ba-name}}(\sigma_{\text{be-name}=\text{‘Bud’}}(\text{serves}))$$

Example #3, Query #2

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #2:

Which drinkers frequent all the bars that serve ‘Bud’?

Alternative without division:

Example #3, Query #2

Relational Schema:

- DRINKER(<u>d-name</u>,d-city)
- BAR(<u>ba-name</u>,ba-city)
- BEER(<u>be-name</u>,type)
- FREQUENTS(<u>d-name</u>,<u>ba-name</u>)
- SERVES(<u>ba-name</u>,<u>be-name</u>)
- LIKES(<u>d-name</u>,<u>be-name</u>)

Query #2:

Which drinkers frequent all the bars that serve 'Bud'?

Alternative without division:

$$\Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})$$

Example #3, Query #2

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #2:

Which drinkers frequent all the bars that serve 'Bud'?

Alternative without division:

$$\Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})$$

All possible drinker,bar pairs

Example #3, Query #2

Relational Schema:

- DRINKER(<u>d-name</u>,d-city)
- BAR(<u>ba-name</u>,ba-city)
- BEER(<u>be-name</u>,type)
- FREQUENTS(<u>d-name</u>,<u>ba-name</u>)
- SERVES(<u>ba-name</u>,<u>be-name</u>)
- LIKES(<u>d-name</u>,<u>be-name</u>)

Query #2:

Which drinkers frequent all the bars that serve ‘Bud’?

Alternative without division:

$$(\Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})) - \text{frequents}$$

Example #3, Query #2

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #2:

Which drinkers frequent all the bars that serve 'Bud'?

Alternative without division:

$$(\Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})) - \text{frequents}$$

Drinkers with the bars that they don't frequent.

Example #3, Query #2

Relational Schema:

- DRINKER(<u>d-name</u>,d-city)
- BAR(<u>ba-name</u>,ba-city)
- BEER(<u>be-name</u>,type)
- FREQUENTS(<u>d-name</u>,<u>ba-name</u>)
- SERVES(<u>ba-name</u>,<u>be-name</u>)
- LIKES(<u>d-name</u>,<u>be-name</u>)

Query #2:

Which drinkers frequent all the bars that serve 'Bud'?

Alternative without division:

$$((\Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})) - \text{frequents}) \bowtie \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves}))$$

Example #3, Query #2

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #2:

Which drinkers frequent all the bars that serve 'Bud'?

Alternative without division:

$$((\Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})) - \text{frequents}) \bowtie \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves}))$$

Drinkers with the bars that serve Bud that they don't frequent.

Example #3, Query #2

Relational Schema:

- DRINKER(<u>d-name</u>,d-city)
- BAR(<u>ba-name</u>,ba-city)
- BEER(<u>be-name</u>,type)
- FREQUENTS(<u>d-name</u>,<u>ba-name</u>)
- SERVES(<u>ba-name</u>,<u>be-name</u>)
- LIKES(<u>d-name</u>,<u>be-name</u>)

Query #2:

Which drinkers frequent all the bars that serve 'Bud'?

Alternative without division:

$$\Pi_{\text{d-name}} (((\Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})) - \text{frequents}) \bowtie \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves})))$$

Example #3, Query #2

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #2:

Which drinkers frequent all the bars that serve 'Bud'?

Alternative without division:

$$\Pi_{\text{d-name}} (((\Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})) - \text{frequents}) \bowtie \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves})))$$

Drinkers that don't frequent some bar that serves Bud.

Example #3, Query #2

Relational Schema:

- DRINKER(<u>d-name</u>,d-city)
- BAR(<u>ba-name</u>,ba-city)
- BEER(<u>be-name</u>,type)
- FREQUENTS(<u>d-name</u>,<u>ba-name</u>)
- SERVES(<u>ba-name</u>,<u>be-name</u>)
- LIKES(<u>d-name</u>,<u>be-name</u>)

Query #2:

Which drinkers frequent all the bars that serve 'Bud'?

Alternative without division:

$$\Pi_{\text{d-name}}(\text{frequents}) - \Pi_{\text{d-name}}\,(((\Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})) - \text{frequents}) \bowtie \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves})))$$

Drinkers that frequent all bars that serve 'Bud'.

Example #3, Query #2

Relational Schema:

- DRINKER(<u>d-name</u>,d-city)
- BAR(<u>ba-name</u>,ba-city)
- BEER(<u>be-name</u>,type)
- FREQUENTS(<u>d-name</u>,<u>ba-name</u>)
- SERVES(<u>ba-name</u>,<u>be-name</u>)
- LIKES(<u>d-name</u>,<u>be-name</u>)

Query #2:

Which drinkers frequent all the bars that serve 'Bud'?

Alternative without division:

$$\Pi_{\text{d-name}}(\text{frequents}) - \Pi_{\text{d-name}}\big((\Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})) - \text{frequents}\big) \bowtie \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves}))$$

Example #3, Query #3

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #3:

Which drinkers frequent a bar that serves a beer they like?

Example #3, Query #3

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Steps to take?

Query #3:

Which drinkers frequent a bar that serves a beer they like?

Example #3, Query #3

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #3:

Which drinkers frequent a bar that serves a beer they like?

Steps to take?

1. Get drinkers with the beers they like and the bars serve those beers.

Example #3, Query #3

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #3:

Which drinkers frequent a bar that serves a beer they like?

Steps to take?

1. Get drinkers with the beers they like and the bars serve those beers.
2. Of those, also get only the bars that they also frequent.

Example #3, Query #3

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #3:

Which drinkers frequent a bar that serves a beer they like?

Steps to take?

1. Get drinkers with the beers they like and the bars that serve those beers.
2. Of those, also get only the bars that they also frequent.
3. Extract the drinkers.

Example #3, Query #3

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #3:

Which drinkers frequent a bar that serves a beer they like?

Step 1: Get drinkers with the beers they like and the bars that serve those beers.

Example #3, Query #3

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #3:

Which drinkers frequent a bar that serves a beer they like?

Step 1: Get drinkers with the beers they like and the bars that serve those beers.

likes $\bowtie$ serves

Example #3, Query #3

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #3:

Which drinkers frequent a bar that serves a beer they like?

Step 2: Of those, also get only the bars that they also frequent.

$$likes \bowtie serves$$

Example #3, Query #3

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #3:

Which drinkers frequent a bar that serves a beer they like?

Step 2: Of those, also get only the bars that they also frequent.

$$likes \bowtie serves \bowtie frequents$$

Example #3, Query #3

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #3:

Which drinkers frequent a bar that serves a beer they like?

Step 3: Extract the drinkers.

$$likes \bowtie serves \bowtie frequents$$

Example #3, Query #3

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #3:

Which drinkers frequent a bar that serves a beer they like?

Step 3: Extract the drinkers.

$$\Pi_{\text{d-name}}(\text{likes} \bowtie \text{serves} \bowtie \text{frequents})$$

Example #3, Query #3

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #3:

Which drinkers frequent a bar that serves a beer they like?

$$\Pi_{\text{d-name}}(\text{likes} \bowtie \text{serves} \bowtie \text{frequents})$$

Example #3, Query #4

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #4:

Which drinkers don't frequent any bar that serves a beer they like?

Example #3, Query #4

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Steps to take?

Query #4:

Which drinkers don't frequent any bar that serves a beer they like?

Example #3, Query #4

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #4:

Which drinkers don't frequent any bar that serves a beer they like?

Steps to take?

1. Compute the drinkers that do frequent a bar that serves a beer they like.

Example #3, Query #4

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #4:

Which drinkers don't frequent any bar that serves a beer they like?

Steps to take?

1. Compute the drinkers that do frequent a bar that serves a beer they like.
2. Compute the opposite of those.

Example #3, Query #4

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #4:

Which drinkers don't frequent any bar that serves a beer they like?

Step 1: Compute the drinkers that do frequent a bar that serves a beer they like.

Example #3, Query #4

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #4:

Which drinkers don't frequent any bar that serves a beer they like?

Step 1: Compute the drinkers that do frequent a bar that serves a beer they like.

$$\Pi_{\text{d-name}}(\text{likes} \bowtie \text{serves} \bowtie \text{frequents})$$

Example #3, Query #4

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #4:

Which drinkers don't frequent any bar that serves a beer they like?

Step 2: Compute the opposite of those.

$$\Pi_{\text{d-name}}(\text{likes} \bowtie \text{serves} \bowtie \text{frequents})$$

Example #3, Query #4

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #4:

Which drinkers don't frequent any bar that serves a beer they like?

Step 2: Compute the opposite of those.

$$\Pi_{\text{d-name}}(\text{drinker}) - \Pi_{\text{d-name}}(\text{likes} \bowtie \text{serves} \bowtie \text{frequents})$$

Example #3, Query #4

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #4:

Which drinkers don't frequent any bar that serves a beer they like?

$$\Pi_{\text{d-name}}(\text{drinker}) - \Pi_{\text{d-name}}(\text{likes} \bowtie \text{serves} \bowtie \text{frequents})$$

Example #3, Query #5

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #5:

Which drinkers frequent only bars that serve some beer they like?

Example #3, Query #5

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Steps to take?

Query #5:

Which drinkers frequent only bars that serve some beer they like?

Example #3, Query #5

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Steps to take?

Query #5:

Which drinkers frequent only bars that serve some beer they like?

Opposite of: Drinkers that frequent some bar that don't serve a beer they like.

Example #3, Query #5

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #5:

Which drinkers frequent only bars that serve some beer they like?

Opposite of: Drinkers that frequent some bar that don't serve a beer they like.

Steps to take?

1. Compute drinkers with bars that serve a beer they like.

Example #3, Query #5

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #5:

Which drinkers frequent only bars that serve some beer they like?

Opposite of: Drinkers that frequent some bar that don't serve a beer they like.

Steps to take?

1. Compute drinkers with bars that serve a beer they like.
2. Take those out of frequents

Example #3, Query #5

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #5:

Which drinkers frequent only bars that serve some beer they like?

Opposite of: Drinkers that frequent some bar that don't serve a beer they like.

Steps to take?

1. Compute drinkers with bars that serve a beer they like.
2. Take those out of frequents (gives us drinkers that frequent a bar that doesn't serve a beer they like).

Example #3, Query #5

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #5:

Which drinkers frequent only bars that serve some beer they like?

Opposite of: Drinkers that frequent some bar that don't serve a beer they like.

Steps to take?

1. Compute drinkers with bars that serve a beer they like.
2. Take those out of frequents (gives us drinkers that frequent a bar that doesn't serve a beer they like).
3. Take those out of drinker.

Example #3, Query #5

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #5:

Which drinkers frequent only bars that serve some beer they like?

Step 1: Compute drinkers with bars that serve a beer they like.

Example #3, Query #5

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #5:

Which drinkers frequent only bars that serve some beer they like?

Step 1: Compute drinkers with bars that serve a beer they like.

likes $\bowtie$ serves

Example #3, Query #5

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #5:

Which drinkers frequent only bars that serve some beer they like?

Step 1: Compute drinkers with bars that serve a beer they like.

$$\Pi_{\text{d-name,ba-name}}(\text{likes} \bowtie \text{serves})$$

Example #3, Query #5

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #5:

Which drinkers frequent only bars that serve some beer they like?

Step 2: Take those out of frequents

$$\Pi_{\text{d-name,ba-name}}(\text{likes} \bowtie \text{serves})$$

Example #3, Query #5

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #5:

Which drinkers frequent only bars that serve some beer they like?

Step 2: Take those out of frequents

$$\text{frequents} - \Pi_{\text{d-name,ba-name}}(\text{likes} \bowtie \text{serves})$$

Example #3, Query #5

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #5:

Which drinkers frequent only bars that serve some beer they like?

Step 2: Take those out of frequents

$$\Pi_{\text{d-name}}(\text{frequents} - \Pi_{\text{d-name,ba-name}}(\text{likes} \bowtie \text{serves}))$$

Example #3, Query #5

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #5:

Which drinkers frequent only bars that serve some beer they like?

Step 3: Take those out of drinker.

$$\Pi_{\text{d-name}}(\text{frequents} - \Pi_{\text{d-name,ba-name}}(\text{likes} \bowtie \text{serves}))$$

Example #3, Query #5

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #5:

Which drinkers frequent only bars that serve some beer they like?

Step 3: Take those out of drinker.

$$\Pi_{\text{d-name}}(\text{drinker}) - \Pi_{\text{d-name}}(\text{frequents} - \Pi_{\text{d-name,ba-name}}(\text{likes} \bowtie \text{serves}))$$

Example #3, Query #5

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #5:

Which drinkers frequent only bars that serve some beer they like?

$$\Pi_{\text{d-name}}(\text{drinker}) - \Pi_{\text{d-name}}(\text{frequents} - \Pi_{\text{d-name,ba-name}}(\text{likes} \bowtie \text{serves}))$$

Example #3, Query #6

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #6:

Which drinkers frequent all the bars in Chicago and NY.

Example #3, Query #6

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Steps to take?

Query #6:

Which drinkers frequent all the bars in Chicago and NY.

Example #3, Query #6

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #6:

Which drinkers frequent all the bars in Chicago and NY.

Steps to take?

1. Compute all bars in Chicago and New York.

Example #3, Query #6

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #6:

Which drinkers frequent all the bars in Chicago and NY.

Steps to take?

1. Compute all bars in Chicago and New York.
2. Compute the drinkers that frequent all those bars.

Example #3, Query #6

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #6:

Which drinkers frequent all the bars in Chicago and NY.

Step 1: Compute all bars in Chicago and New York.

Example #3, Query #6

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #6:

Which drinkers frequent all the bars in Chicago and NY.

Step 1: Compute all bars in Chicago and New York.

$$\sigma_{\text{city='Chicago'} \vee \text{city='NY'}}(\text{bar})$$

Example #3, Query #6

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #6:

Which drinkers frequent all the bars in Chicago and NY.

Step 1: Compute all bars in Chicago and New York.

$$\Pi_{\text{ba-name}}(\sigma_{\text{city='Chicago'} \vee \text{city='NY'}}(\text{bar}))$$

Example #3, Query #6

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #6:

Which drinkers frequent all the bars in Chicago and NY.

Step 2: Compute the drinkers that frequent all those bars.

$$\Pi_{\text{ba-name}}(\sigma_{\text{city='Chicago'} \vee \text{city='NY'}}(\text{bar}))$$

Example #3, Query #6

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #6:

Which drinkers frequent all the bars in Chicago and NY.

Step 2: Compute the drinkers that frequent all those bars.

$$\text{frequents} \div \Pi_{\text{ba-name}}(\sigma_{\text{city='Chicago'} \vee \text{city='NY'}}(\text{bar}))$$

Example #3, Query #6

Relational Schema:

- DRINKER(d-name,d-city)
- BAR(ba-name,ba-city)
- BEER(be-name,type)
- FREQUENTS(d-name,ba-name)
- SERVES(ba-name,be-name)
- LIKES(d-name,be-name)

Query #6:

Which drinkers frequent all the bars in Chicago and NY.

$$\text{frequents} \div \Pi_{\text{ba-name}}(\sigma_{\text{city='Chicago'} \vee \text{city='NY'}}(\text{bar}))$$

Rename (ρ)

- Syntax: $\rho_x(r)$ or $\rho_x(E)$, where E is a relational algebra expression.
- Renames r into x.
- Returns r under the name x.

Retrieve the students who got the same grade as student #194.

Scheme: ENROL(student-id,course,grade)

Rename (ρ)

- Syntax: $\rho_x(r)$ or $\rho_x(E)$, where E is a relational algebra expression.
- Renames r into x.
- Returns r under the name x.

Retrieve the students who got the same grade as student #194.

Scheme: ENROL(student-id,course,grade)

$$\Pi_{\text{enrol.student-id}}(\sigma_{\text{enrol.grade=d.grade} \wedge \text{enrol.course=d.course}}(\text{enrol} \times \sigma_{\text{student-id='194'}}(\rho_d(\text{enrol}))))$$

Rename (ρ)

- EMPLOYEE(name,address,salary)
- Query: Retrieve the highest salary of any of the employees.
- Steps to take?

Rename (ρ)

- EMPLOYEE(name,address,salary)
- Query: Retrieve the highest salary of any of the employees.
- Steps to take?
  - Compute all salaries that are less than another salary

Rename (ρ)

- EMPLOYEE(name,address,salary)
- Query: Retrieve the highest salary of any of the employees.
- Steps to take?
  - Compute all salaries that are less than another salary
  - Take all of those out from all the employee salaries.

Rename (ρ)

- EMPLOYEE(name,address,salary)
- Query: Retrieve the highest salary of any of the employees.

Step 1: Compute all salaries that are less than another salary

Rename (ρ)

- EMPLOYEE(name,address,salary)
- Query: Retrieve the highest salary of any of the employees.

Step 1: Compute all salaries that are less than another salary

$$\Pi_{salary}(employee) - \Pi_{employee.salary}(\sigma_{employee.salary < d.salary}(employee \times \rho_{d}(employee)))$$

Rename (ρ)

- EMPLOYEE(name,address,salary)
- Query: Retrieve the highest salary of any of the employees.

Step 1: Compute all salaries that are less than another salary

$$\Pi_{salary}(employee) - \Pi_{employee.salary}(\sigma_{employee.salary < d.salary}(employee \times \rho_{d}(employee)))$$

Rename (ρ)

- EMPLOYEE(name,address,salary)
- Query: Retrieve the highest salary of any of the employees.

Step 1: Compute all salaries that are less than another salary

$$\Pi_{salary}(employee) - \Pi_{employee.salary}(\sigma_{employee.salary < d.salary}(employee \times \rho_{d}(employee)))$$

Rename (ρ)

- EMPLOYEE(name,address,salary)
- Query: Retrieve the highest salary of any of the employees.

Step 2: Take all of those out from all the employee salaries.

$$\Pi_{salary}(employee) - \Pi_{employee.salary}(\sigma_{employee.salary < d.salary}(employee \times \rho_{d}(employee)))$$

Rename (ρ)

- EMPLOYEE(name,address,salary)
- Query: Retrieve the highest salary of any of the employees.

Step 2: Take all of those out from all the employee salaries.

$$\Pi_{salary}(employee) - \Pi_{employee.salary}(\sigma_{employee.salary < d.salary}(employee \times \rho_{d}(employee)))$$

Rename (ρ)

- EMPLOYEE(name,address,salary)
- Query: Retrieve the highest salary of any of the employees.

$$\Pi_{salary}(employee) - \Pi_{employee.salary}(\sigma_{employee.salary < d.salary}(employee \times \rho_{d}(employee)))$$

Assignment (←)

- Doesn’t add any extra power to relational algebra.
- Just used to make complex queries easier to read by assigning expressions to temporary variables.

Assignment (←)

Which drinkers frequent all the bars that serve ‘Bud’?

$$\Pi_{\text{d-name}}(\text{drinker}) - \Pi_{\text{d-name}}\,(((\Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})) - \text{frequents}) \bowtie \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves})))$$

$$\textit{barsThatServeBud} \leftarrow \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves}))$$

$$\textit{allDrinkersAllBars} \leftarrow \Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})$$

$$\textit{drinkersBarsNotFreq} \leftarrow \textit{allDrinkersAllBars} - \text{frequents}$$

$$\textit{drinkerNotFreqBud} \leftarrow \Pi_{\text{d-name}}\,(\textit{drinkersBarsNotFreq}\ \textit{barsThatServeBud})$$

$$\textit{result} \leftarrow \Pi_{\text{d-name}}(\text{frequents}) - \textit{drinkerNotFreqBud}$$

Assignment (←)

Which drinkers frequent all the bars that serve ‘Bud’?

$$\Pi_{\text{d-name}}(\text{drinker}) - \Pi_{\text{d-name}}\,(((\Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})) - \text{frequents}) \bowtie \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves})))$$

$$\textit{barsThatServeBud} \leftarrow \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves}))$$

$$\textit{allDrinkersAllBars} \leftarrow \Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})$$

$$\textit{drinkersBarsNotFreq} \leftarrow \textit{allDrinkersAllBars} - \text{frequents}$$

$$\textit{drinkerNotFreqBud} \leftarrow \Pi_{\text{d-name}}\,(\textit{drinkersBarsNotFreq}\ \textit{barsThatServeBud})$$

$$\textit{result} \leftarrow \Pi_{\text{d-name}}(\text{frequents}) - \textit{drinkerNotFreqBud}$$

Assignment (←)

Which drinkers frequent all the bars that serve ‘Bud’?

$$\Pi_{\text{d-name}}(\text{drinker}) - \Pi_{\text{d-name}}\,(((\Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})) - \text{frequents}) \bowtie \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves})))$$

$$\textit{barsThatServeBud} \leftarrow \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves}))$$

$$\textit{allDrinkersAllBars} \leftarrow \Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})$$

$$\textit{drinkersBarsNotFreq} \leftarrow \textit{allDrinkersAllBars} - \text{frequents}$$

$$\textit{drinkerNotFreqBud} \leftarrow \Pi_{\text{d-name}}\,(\textit{drinkersBarsNotFreq}\ \textit{barsThatServeBud})$$

$$\textit{result} \leftarrow \Pi_{\text{d-name}}(\text{frequents}) - \textit{drinkerNotFreqBud}$$

Assignment (←)

Which drinkers frequent all the bars that serve ‘Bud’?

$$\Pi_{\text{d-name}}(\text{drinker}) - \Pi_{\text{d-name}}\big((\Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})) - \text{frequents}) \bowtie \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves}))\big)$$

$$\textit{barsThatServeBud} \leftarrow \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves}))$$

$$\textit{allDrinkersAllBars} \leftarrow \Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})$$

$$\textit{drinkersBarsNotFreq} \leftarrow \textit{allDrinkersAllBars} - \text{frequents}$$

$$\textit{drinkerNotFreqBud} \leftarrow \Pi_{\text{d-name}}(\textit{drinkersBarsNotFreq} \quad \textit{barsThatServeBud})$$

$$\textit{result} \leftarrow \Pi_{\text{d-name}}(\text{frequents}) - \textit{drinkerNotFreqBud}$$

Assignment (←)

Which drinkers frequent all the bars that serve ‘Bud’?

$$\Pi_{\text{d-name}}(\text{drinker}) - \Pi_{\text{d-name}}\big((\Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})) - \text{frequents}) \bowtie \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves}))\big)$$

$$\textit{barsThatServeBud} \leftarrow \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves}))$$
$$\textit{allDrinkersAllBars} \leftarrow \Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})$$
$$\textit{drinkersBarsNotFreq} \leftarrow \textit{allDrinkersAllBars} - \text{frequents}$$
$$\textit{drinkerNotFreqBud} \leftarrow \Pi_{\text{d-name}}(\textit{drinkersBarsNotFreq} \bowtie \textit{barsThatServeBud})$$
$$\textit{result} \leftarrow \Pi_{\text{d-name}}(\text{frequents}) - \textit{drinkerNotFreqBud}$$

Assignment (←)

Which drinkers frequent all the bars that serve ‘Bud’?

$$\Pi_{\text{d-name}}(\text{drinker}) - \Pi_{\text{d-name}}\,(((\Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})) - \text{frequents}) \bowtie \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves})))$$

$$\textit{barsThatServeBud} \leftarrow \Pi_{\text{ba-name}}(\sigma_{\text{be-name='Bud'}}(\text{serves}))$$

$$\textit{allDrinkersAllBars} \leftarrow \Pi_{\text{d-name}}(\text{drinker}) \times \Pi_{\text{ba-name}}(\text{bar})$$

$$\textit{drinkersBarsNotFreq} \leftarrow \textit{allDrinkersAllBars} - \text{frequents}$$

$$\textit{drinkerNotFreqBud} \leftarrow \Pi_{\text{d-name}}\,(\textit{drinkersBarsNotFreq} \bowtie \textit{barsThatServeBud})$$

$$\textit{result} \leftarrow \Pi_{\text{d-name}}(\text{frequents}) - \textit{drinkerNotFreqBud}$$