Updated 2/8/2020

T1: Data models and query languages
L3: Relational calculus, relational algebra

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp20)

https://northeastern-datalab.github.io/cs7240/sp20/

1/14/2020

Where we are

Topic 1: Data models and query languages

- **Lecture 1 (Tue 1/7):** Course introduction, SQL refresher
  - Introduction, SQL
- **Lecture 2 (Fri 1/10):** Logic & relational calculus
  - SQL continued, Logic & relational calculus
- **Lecture 3 (Tue 1/14):** Relational Calculus, Relational algebra
- **Lecture 4 (Fri 1/17):** Codd's theorem, Datalog
- **Lecture 5 (Tue 1/21):** Stable model semantics, Information theory & normal forms
- **Lecture 6 (Fri 1/24): (A1 due)** Alternative data models

Queries and the connection to logic and algebra

- Why logic?
  - A crash course on FOL
- Relational Calculus
  - Syntax and Semantics
  - Domain Independence and Safety
- Relational Algebra
  - Operators
  - Independence
  - Power of algebra: optimizations
- Equivalence RC and RA

Bringing in the Domain

- Let $\mathbf{S}$ be a schema, D a database over $\mathbf{S}$, and Q an RC query over $\mathbf{S}$
- D gives an interpretation for the underlying FOL
  - Predicates $\longrightarrow$ relations; constants copied; no functions

- Not yet! We need to answer first: *What is the domain?*

- The *active domain* **ADom** (of D and Q) is the set of all the values that occur in either D or Q
- The query Q is evaluated over D with respect to a domain **Dom** that contains the active domain ($\mathbf{Dom} \supseteq \mathbf{ADom}$)
- Denote by $Q^{\mathbf{Dom}}(D)$ the result of evaluating Q over D relative to the domain **Dom**

Domain Independence

- Let $\mathbf{S}$ be a schema, and let Q be an RC query over $\mathbf{S}$
- We say that Q is domain independent if for every database D over $\mathbf{S}$ and every two domains $\mathbf{Dom1}$ and $\mathbf{Dom2}$ that contain the active domain, we have:

$$Q^{\mathbf{Dom1}}(D) = Q^{\mathbf{Dom2}}(D) = Q^{\mathbf{ADom}}(D)$$

Bad News...

- We would like be able to tell whether a given RA query is domain independent, and then reject "bad queries"
- Alas, this problem is undecidable!
  - That is, there is no algorithm that takes as input an RC query and returns true iff the query is domain independent

First observed in "Di Paola. The Recursive Unsolvability of the Decision Problem for the Class of Definite Formulas, JACM 1969. https://doi.org/10.1145/321510.321524"

Good News

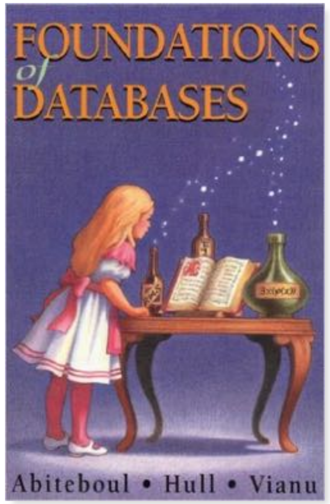
Domain-independent RC has an effective syntax; that is:

  - A syntactic restriction of RC in which every query is domain independent
  - Restricted queries are said to be *safe*

- Safety can be tested automatically (and efficiently)
  - Most importantly, for every domain independent RC query there exists an equivalent safe RC query!

Safety

- We do not formally define the safe syntax in this course
- Details on the safe syntax can be found in Ch 5.4 of [Alice'95]: Foundations of Databases by Abiteboul, Hull and Vianu
  - Example:
    - In $\exists x\ \varphi$, the variable x should be guarded by φ
    - Every variable x_i is guarded by $R(x_1,\ldots,x_k)$
    - In $\varphi \wedge (x=y)$, the variable x is guarded if and only if either x or y is guarded by φ
    - ... and so on



An accessible overview of issues involving safety and issues can be found in
Topor. Safety and Domain Independence. Encyclopedia of Database Systems. https://doi.org/10.1007/978-0-387-39940-9_1255

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

$\{ (x) \mid \neg\text{Person}(x, \text{'female'}, \text{'Canada'}) \}$?

$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \wedge y=z] \}$?

$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \wedge y \neq z] \}$?

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

Example fixes: $\ldots \land \exists y,z.\text{Person}(x,y,z)$

$\ldots \land \exists y.\text{Person}(x,y,\text{'Canada'})$

$\{ (x) \mid \neg\text{Person}(x, \text{'female'}, \text{'Canada'}) \}$

Not DI

x could be also 'Canada' or 'female' or ...

$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \land y=z] \}$

?

$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \land y\neq z] \}$

?

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

Example fixes: … $\wedge\ \exists y,z.\text{Person}(x,y,z)$

… $\wedge\ \exists y.\text{Person}(x,y,\text{'Canada'})$

$\{ (x) \mid \neg\text{Person}(x, \text{'female'}, \text{'Canada'}) \}$ — Not DI

x could be also 'Canada' or 'female' or …

$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \wedge y=z] \}$ — DI

same as $\{(x,y) \mid \text{Spouse}(x,y)\} = \text{Spouse}(x,y)$

$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \wedge y\neq z] \}$ — ?

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

Example fixes: ... $\wedge\ \exists y,z.\text{Person}(x,y,z)$
... $\wedge\ \exists y.\text{Person}(x,y,\text{'Canada'})$

$$\{\ (x) \mid \neg\text{Person}(x, \text{'female'}, \text{'Canada'})\ \}$$

Not DI

x could be also 'Canada' or 'female' or ...

$$\{\ (x,y) \mid \exists z\ [\text{Spouse}(x,z) \wedge y=z]\ \}$$

DI

same as $\{(x,y) \mid \text{Spouse}(x,y)\}$

$$\{\ (x,y) \mid \exists z\ [\text{Spouse}(x,z) \wedge y\neq z]\ \}$$

?

D: Spouse('Alice','Bob')
Dom_1={'Alice','Bob'}
Dom_2={'Alice','Bob','Charly'}

Dom $\supseteq$ ADom

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

Example fixes: $\ldots \wedge \exists y,z.\text{Person}(x,y,z)$

$\ldots \wedge \exists y.\text{Person}(x,y,\text{'Canada'})$

$\{ (x) \mid \neg\text{Person}(x, \text{'female'}, \text{'Canada'}) \}$ **Not DI**

x could be also 'Canada' or 'female' or ...

$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \wedge y=z] \}$ **DI**

same as $\{(x,y) \mid \text{Spouse}(x,y)\}$

$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \wedge y \neq z] \}$ **Not DI**

D: Spouse('Alice','Bob')

Dom_1={'Alice','Bob'} → {('Alice','Alice')}

Dom_2={'Alice','Bob','Charly'} → {('Alice','Alice'), ('Alice','Charly')}

Dom ⊇ ADom

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

$$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \land \exists y\ [\neg \text{Likes}(x,y)] \}$$?

$$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \land \forall y\ [\neg \text{Likes}(x,y)] \}$$?

$$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \land \forall y\ [\neg \text{Likes}(x,y)] \land \exists y\ [\neg \text{Likes}(x,y)] \}$$?

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

D

Person('Alice', 'female', 'Canada')	Likes('Alice', 'Beate')
Person('Beate', 'female', 'Canada')	Likes('Alice', 'Cecile')
Person('Cecile', 'female', 'Canada')	Likes('Alice', 'Alice')

ADom = ?

$$\{\ (x)\ |\ \exists z,w\ \text{Person}(x,z,w) \land \exists y\ [\neg \text{Likes}(x,y)]\ \}$$

$$\{\ (x)\ |\ \exists z,w\ \text{Person}(x,z,w) \land \forall y\ [\neg \text{Likes}(x,y)]\ \}$$

$$\{\ (x)\ |\ \exists z,w\ \text{Person}(x,z,w) \land \forall y\ [\neg \text{Likes}(x,y)] \land \exists y\ [\neg \text{Likes}(x,y)]\ \}$$

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

D

Person('Alice', 'female', 'Canada')	Likes('Alice', 'Beate')
Person('Beate', 'female', 'Canada')	Likes('Alice', 'Cecile')
Person('Cecile', 'female', 'Canada')	Likes('Alice', 'Alice')

ADom = {'Alice', 'Beate', 'Cecile', 'female', 'Canada'}

$$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \land \exists y\ [\neg \text{Likes}(x,y)] \}$$

$$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \land \forall y\ [\neg \text{Likes}(x,y)] \}$$

$$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \land \forall y\ [\neg \text{Likes}(x,y)] \land \exists y\ [\neg \text{Likes}(x,y)] \}$$

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

D

Person('Alice', 'Alice', 'Alice')	Likes('Alice', 'Beate')
Person('Beate', 'Beate', 'Beate')	Likes('Alice', 'Cecile')
Person('Cecile', 'Beate', 'Beate')	Likes('Alice', 'Alice')

ADom = ?

$$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \wedge \exists y\ [\neg \text{Likes}(x,y)] \}$$

$$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \wedge \forall y\ [\neg \text{Likes}(x,y)] \}$$

$$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \wedge \forall y\ [\neg \text{Likes}(x,y)] \wedge \exists y\ [\neg \text{Likes}(x,y)] \}$$

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

D

Person('Alice', 'Alice', 'Alice')	Likes('Alice', 'Beate')
Person('Beate', 'Beate', 'Beate')	Likes('Alice', 'Cecile')
Person('Cecile', 'Beate', 'Beate')	Likes('Alice', 'Alice')

ADom = {'Alice', 'Beate', 'Cecile')

Dom = {'Alice', 'Beate', 'Cecile', 'Dora')

$\{ (x) \mid \exists z,w\ Person(x,z,w) \land \exists y\ [\neg Likes(x,y)] \}$?

$\{ (x) \mid \exists z,w\ Person(x,z,w) \land \forall y\ [\neg Likes(x,y)] \}$?

$\{ (x) \mid \exists z,w\ Person(x,z,w) \land \forall y\ [\neg Likes(x,y)] \land \exists y\ [\neg Likes(x,y)] \}$?

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

D

Person('Alice', 'Alice', 'Alice')
Person('Beate', 'Beate', 'Beate')
Person('Cecile', 'Beate', 'Beate')

Likes('Alice', 'Beate')
Likes('Alice', 'Cecile')
Likes('Alice', 'Alice')

ADom = {'Alice', 'Beate', 'Cecile')

Dom = {'Alice', 'Beate', 'Cecile', 'Dora')

Example fix: ?

{ (x) |∃z,w Person(x,z,w) ∧ ∃y [¬Likes(x,y)] }

Alice is in the output if Dom ⊃ ADom (Dora is in Dom)

Not DI

{ (x) |∃z,w Person(x,z,w) ∧ ∀y [¬Likes(x,y)] }

?

{ (x) |∃z,w Person(x,z,w) ∧ ∀y [¬Likes(x,y)] ∧ ∃y [¬Likes(x,y)] }

?

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

D

Person('Alice', 'Alice', 'Alice')	Likes('Alice', 'Beate')
Person('Beate', 'Beate', 'Beate')	Likes('Alice', 'Cecile')
Person('Cecile', 'Beate', 'Beate')	Likes('Alice', 'Alice')

ADom = {'Alice', 'Beate', 'Cecile')

Dom = {'Alice', 'Beate', 'Cecile', 'Dora')

Example fix: $\ldots \land \exists u,v\ [\text{Person}(y,u,v)]$

$$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \land \exists y\ [\neg \text{Likes}(x,y)] \}$$

Not DI

Alice is in the output if Dom ⊃ ADom (Dora is in Dom)

$$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \land \forall y\ [\neg \text{Likes}(x,y)] \}$$

?

$$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \land \forall y\ [\neg \text{Likes}(x,y)] \land \exists y\ [\neg \text{Likes}(x,y)] \}$$

?

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

D

Person('Alice', 'Alice', 'Alice')
Person('Beate', 'Beate', 'Beate')
Person('Cecile', 'Beate', 'Beate')

Likes('Alice', 'Beate')
Likes('Alice', 'Cecile')

ADom = {'Alice', 'Beate', 'Cecile')
Dom = {'Alice', 'Beate', 'Cecile', 'Dora')

Example fix: … ∧ ∃u,v [Person(y,u,v)]

{ (x) |∃z,w Person(x,z,w) ∧ ∃y [¬Likes(x,y)] } — Not DI

Alice is in the output if Dom ⊃ ADom (Dora is in Dom)

{ (x) |∃z,w Person(x,z,w) ∧ ∀y [¬Likes(x,y)] } — DI

x never occurs in Likes(x,_): Beate, Cecile

{ (x) |∃z,w Person(x,z,w) ∧ ∀y [¬Likes(x,y)] ∧ ∃y [¬Likes(x,y)] } — ?

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

D

Person('Alice', 'Alice', 'Alice')
Person('Beate', 'Beate', 'Beate')
Person('Cecile', 'Beate', 'Beate')

Likes('Alice', 'Beate')
Likes('Alice', 'Cecile')

ADom = {'Alice', 'Beate', 'Cecile'}

Dom = {'Alice', 'Beate', 'Cecile', 'Dora'}

Example fix: ... $\land \exists u,v$ [Person(y,u,v)]

$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \land \exists y\ [\neg \text{Likes}(x,y)] \}$ — **Not DI**

Alice is in the output if Dom ⊃ ADom (Dora is in Dom)

$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \land \forall y\ [\neg \text{Likes}(x,y)] \}$ — **DI**

x never occurs in Likes(x,_): Beate, Cecile

$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \land \forall y\ [\neg \text{Likes}(x,y)] \land \exists y\ [\neg \text{Likes}(x,y)] \}$ — **DI**

implication (absorption) if Dom ≠ ∅, which is necessary for there to be Person(x,_,_)

What is the meaning of the following expressions?

$\{\ x \mid \exists y.\ R(x)\ \}$?

$\{\ x \mid x \geq 10\ \}$?

$\{\ x \mid \forall y\ R(x,y)\ \}$?

What is the meaning of the following expressions?

$\{\, x \mid \exists y.\ R(x) \,\}$ logically equivalent to $\{\, x \mid R(x) \,\} = R(x)$

$\{\, x \mid x \geq 10 \,\}$?

$\{\, x \mid \forall y\ R(x,y) \,\}$?

What is the meaning of the following expressions?

$\{\, x \mid \exists y.\ R(x) \,\}$ logically equivalent to $\{ x \mid R(x) \} = R(x)$

$\{\, x \mid x \geq 10 \,\}$ What if Dom=$\mathbb{N}$? $\{\, x \mid A(x) \wedge x \geq 10 \,\}$

$\{\, x \mid \forall y\ R(x,y) \,\}$?

What is the meaning of the following expressions?

$\{\, x \mid \exists y.\ R(x) \,\}$ — logically equivalent to $\{\, x \mid R(x) \,\} = R(x)$

$\{\, x \mid x \geq 10 \,\}$ — What if Dom=ℕ?

$\{\, x \mid A(x) \wedge x \geq 10 \,\}$

$\{\, x \mid \forall y\ R(x,y) \,\}$

D: R('a','a')
ADom={'a'}
Dom={'a','Chile'}

$\{\, x \mid \forall y\ [A(y) \rightarrow R(x,y)] \,\}$

what if relation A is empty?

What is the meaning of the following expressions?

$\{ x \mid \exists y.\ R(x) \}$

logically equivalent to $\{ x \mid R(x) \} = R(x)$

$\{ x \mid x \geq 10 \}$

What if Dom=ℕ?

$\{ x \mid A(x) \wedge x \geq 10 \}$

$\{ x \mid \forall y\ R(x,y) \}$

D: R('a','a')
ADom={'a'}
Dom={'a','Chile'}

$\{ x \mid \forall y\ [A(y) \rightarrow R(x,y)] \}$

what if relation A is empty?

1. always true for A=∅

$\{ x \mid \forall y\ [\neg A(y) \vee R(x,y)] \}$

What is the meaning of the following expressions?

$\{ x \mid \exists y.\ R(x) \}$ logically equivalent to $\{ x \mid R(x) \} = R(x)$

$\{ x \mid x \geq 10 \}$ What if Dom=ℕ? $\{ x \mid A(x) \land x \geq 10 \}$

$\{ x \mid \forall y\ R(x,y) \}$

D: R('a','a')
ADom={'a'}
Dom={'a','Chile'}

$\{ x \mid \forall y\ [A(y) \rightarrow R(x,y)] \}$

what if relation A is empty?

1. always true for A=∅

$\{ x \mid \forall y\ [\neg A(y) \lor R(x,y)] \}$

2. alternative way to see that

Neutral element for ∀ is true

Σ: $0 + x = x$

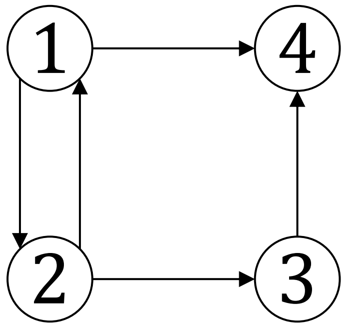
$\prod$: $1 \cdot x = x$

$\lor$: $\text{FALSE} \lor x = x$ $\exists$: $x_1 \lor x_2 \lor \ldots \lor \text{FALSE}$

$\land$: $\text{TRUE} \land x = x$ $\forall$: $x_1 \land x_2 \land \ldots \land \text{TRUE}$

MIN: $\text{MIN}(\infty, x) = x$

Example: Querying a Graph



What do these queries return ?

$$\{\, x \mid \exists y.\ E(x,y) \,\}$$

?

$$\{\, x \mid \exists y,z,u.[E(x,y) \wedge E(y,z) \wedge E(z,u)] \}$$

?

$$\{\, (x,y) \mid \forall z.[E(x,z) \rightarrow E(y,z)] \}$$

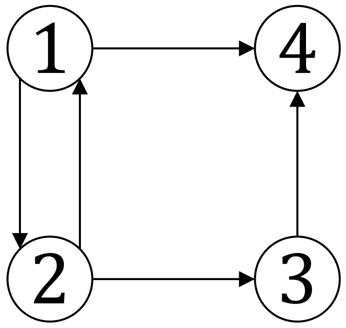
?

E:

1	2
2	1
2	3
1	4
3	4

E encodes the directed edges of a graph

Example: Querying a Graph



What do these queries return ?

$$\{\ x \mid \exists y.\ E(x,y)\ \}$$

Nodes that have at least one child: {1,2,3}

E:

1	2
2	1
2	3
1	4
3	4

$$\{\ x \mid \exists y,z,u.[E(x,y) \land E(y,z) \land E(z,u)]\}$$

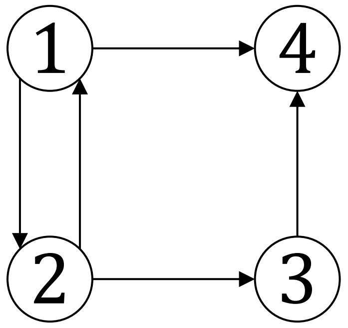
?

$$\{\ (x,y) \mid \forall z.[E(x,z) \rightarrow E(y,z)]\}$$

?

E encodes the directed edges of a graph

Example: Querying a Graph



What do these queries return ?

$$\{\, x \mid \exists y.\ E(x,y) \,\}$$

Nodes that have at least one child: {1,2,3}

$$\{\, x \mid \exists y,z,u.[E(x,y) \wedge E(y,z) \wedge E(z,u)] \}$$

Nodes that have a great-grand-child: {1,2}

$$\{\, (x,y) \mid \forall z.[E(x,z) \rightarrow E(y,z)] \}$$

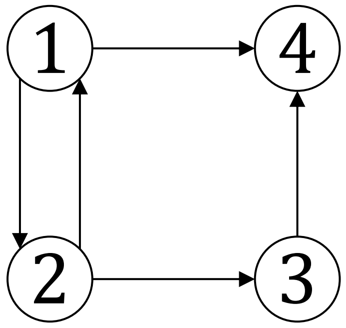
?

E:

1	2
2	1
2	3
1	4
3	4

E encodes the directed edges of a graph

Example: Querying a Graph



E:

1	2
2	1
2	3
1	4
3	4

E encodes the directed edges of a graph

What do these queries return ?

$$\{\ x \mid \exists y.\ E(x,y)\ \}$$

Nodes that have at least one child: {1,2,3}

$$\{\ x \mid \exists y,z,u.[E(x,y) \land E(y,z) \land E(z,u)]\}$$

Nodes that have a great-grand-child: {1,2}

$\nexists z.[E(x,z) \land \neg E(x,z)]$

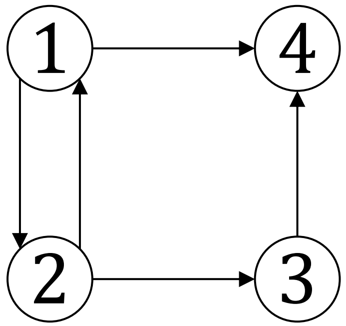
$$\{\ (x,y) \mid \forall z.[E(x,z) \rightarrow E(y,z)]\}$$

Every child of x is a child of y.

Which of the following tuples fulfill the condition?

(1,1) (4,4) (1,3) (3,1) (4,1)

Example: Querying a Graph



What do these queries return ?

$$\{\ x \mid \exists y.\ E(x,y)\ \}$$

Nodes that have at least one child: {1,2,3}

$$\{\ x \mid \exists y,z,u.[E(x,y) \land E(y,z) \land E(z,u)]\}$$

Nodes that have a great-grand-child: {1,2}

E:

1	2
2	1
2	3
1	4
3	4

E encodes the directed edges of a graph

$\nexists z.[E(x,z) \land \neg E(x,z)]$

$$\{\ (x,y) \mid \forall z.[E(x,z) \rightarrow E(y,z)]\}$$

Every child of x is a child of y.

Which of the following tuples fulfill the condition?

(1,1) (4,4) ~~(1,3)~~ (3,1) (4,1)

{(1,1),(2,2),(3,1),(3,3),(4,1), (4,2), (4,3)}

The person/bar/drinks schema

Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink)

331

What does this query compute?

$$\{\, x \mid \forall y.[\text{Frequents}(x,y) \rightarrow \exists z.[\text{Serves}(y,z) \wedge \text{Likes}(x,z)]] \,\}$$

Schema adapted from Jeff Ullman's drinkers/bars/beers example to avoid attributes with same first letters

The person/bar/drinks schema

Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink)

What does this query compute?

$$\{\, x \mid \forall y.[\text{Frequents}(x,y) \rightarrow \exists z.[\text{Serves}(y,z) \wedge \text{Likes}(x,z)]] \,\}$$

Find drinkers that frequent only bars that serves some beer they like.

Careful! This query is not domain independent. Why?
Challenge: write this query without the ∀ quantifier!

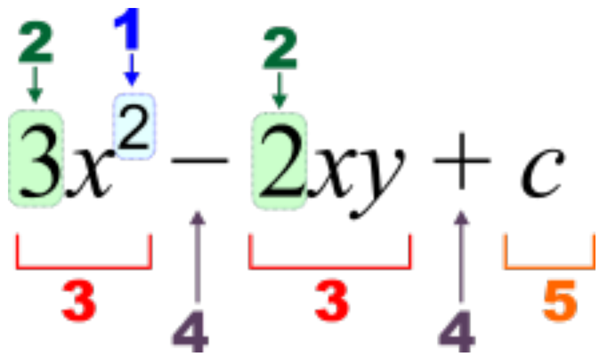
Schema adapted from Jeff Ullman's drinkers/bars/beers example to avoid attributes with same first letters

Queries and the connection to logic and algebra

- Why logic?
  - A crash course on FOL
- Relational Calculus
  - Syntax and Semantics
  - Domain Independence and Safety
- Relational Algebra
  - Operators
  - Independence
  - Power of algebra: optimizations
- Equivalence RC and RA

Algebra

- Algebra is the study of mathematical symbols and the rules for manipulating these symbols
- e.g., Linear Algebra
- e.g., Relational Algebra
- e.g., Boolean Algebra
- e.g., Abstract algebra (groups, rings, fields, ...)
- e.g., Elementary algebra



What is "Algebra"?

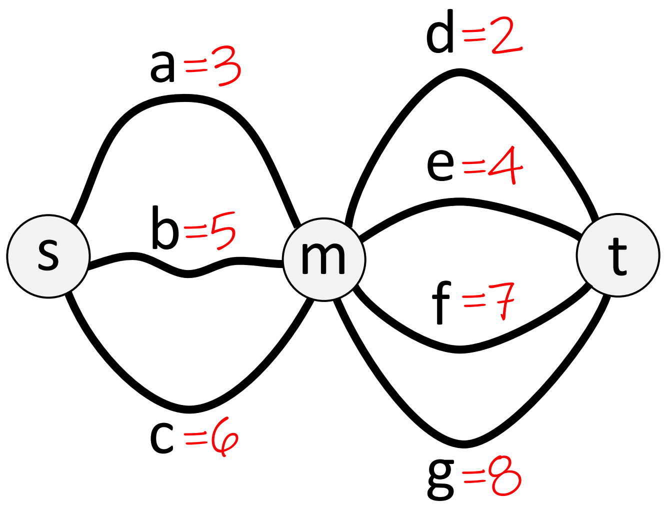
- An *abstract algebra* consists of:
  - A class of *elements*
  - A collection of *operators*
- Each operator:
  - Has an *arity* d
  - Has a *domain* of sequences $(e_1,\ldots,e_d)$ of elements
  - Maps every sequence in its domain to an element e
- The definition of an operator allows for composition:

$$o_1\big(o_2(x),o_1(y,o_4(x,z))\big)$$

- Examples:
  - Ring of integers: $(\mathbb{Z},\{+,\cdot\})$
  - Boolean algebra: $(\{\text{true},\text{false}\},\{\wedge,\vee,\neg\})$
  - Relational algebra

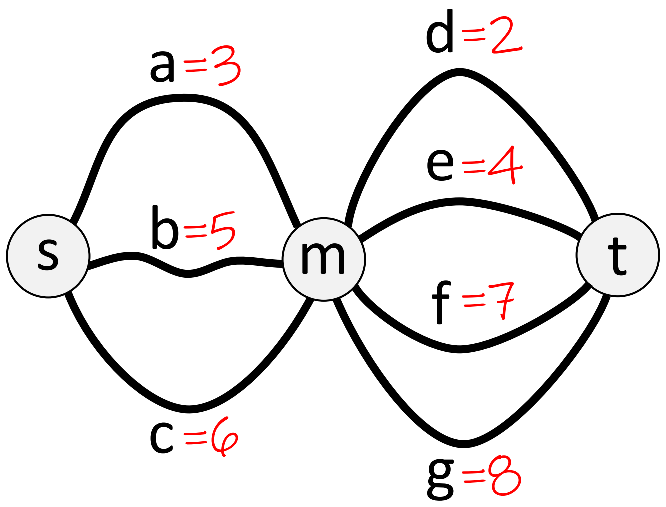
set equipped with two binary operations with certain properties like distributivity of multiplication over addition

Distributivity = efficient factorization



What is the shortest path from s to t?

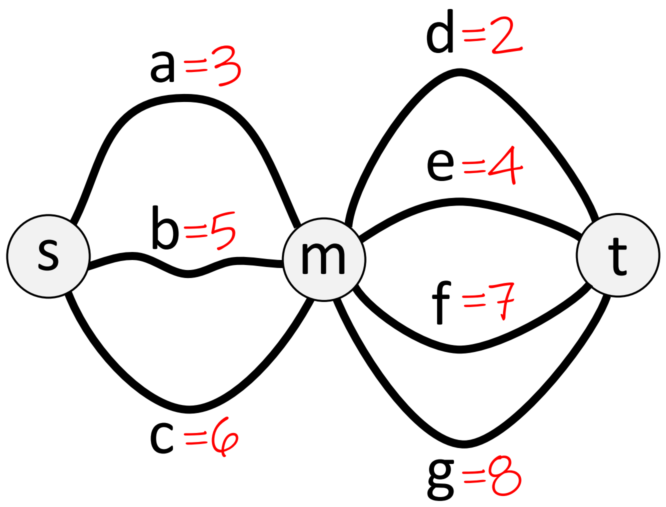
Distributivity = efficient factorization



What is the shortest path from s to t?

Answer: 5 = 3 + 2

Distributivity = efficient factorization



min [a + d, a + e, a + f, a + g, ..., c + g]

min[3+2, 3+4, 3+7, 3+8, ..., 6+8]

= min [a, b, c] + min [d, e, f, g]

min[3,5,6] + min[2,4,7,8]

What is the shortest path from s to t?

Answer: 5 = 3 + 2

min[x,y]+z = min[(x+z), (y+z)]

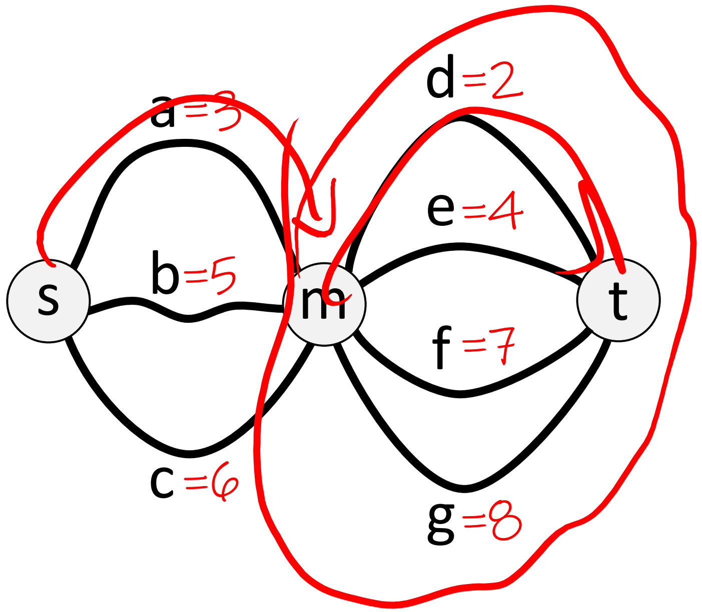
(+ distributes over min)

Distributivity = efficient factorization

(Tropical semiring)

- Semiring ($\mathbb{R}^\infty$,min,+,∞,0)

Principle of optimality from Dynamic Programming: *irrespective of the initial state and decision, an optimal solution continues optimally from the resulting state*



min [a + d, a + e, a + f, a + g, ..., c + g]

min[3+2, 3+4, 3+7, 3+8, ..., 6+8]

= min [a, b, c] + min [d, e, f, g]

min[3,5,6] + min[2,4,7,8]

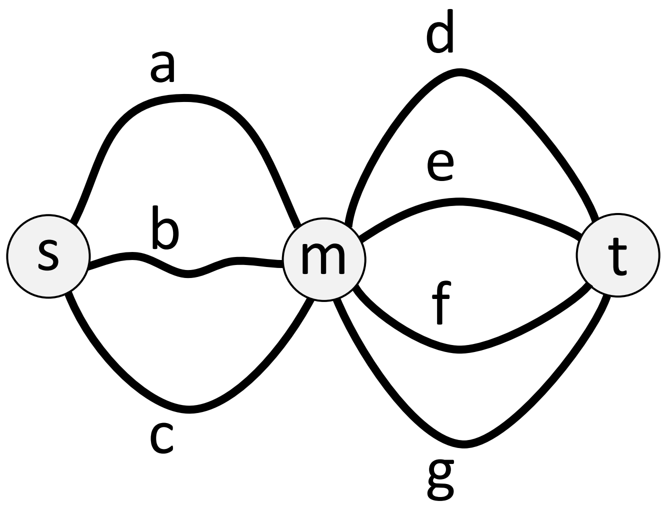
What is the shortest path from s to t?

Answer: 5 = 3 + 2

min[x,y]+z = min[(x+z), (y+z)]

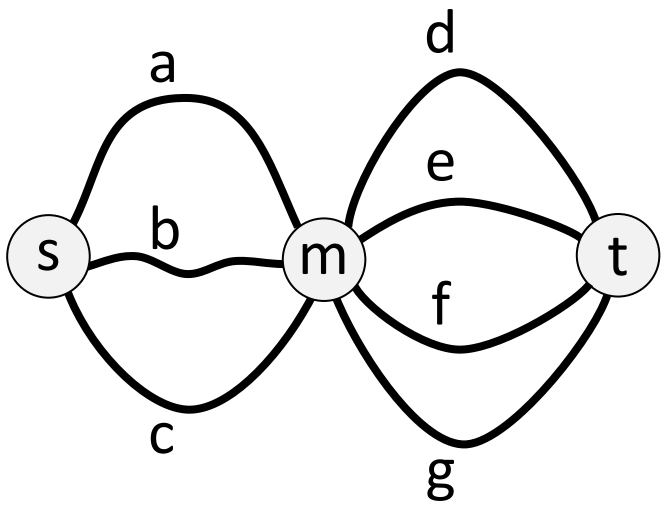
(+ distributes over min)

Distributivity = efficient factorization



How many paths are there from s to t?

Distributivity = efficient factorization



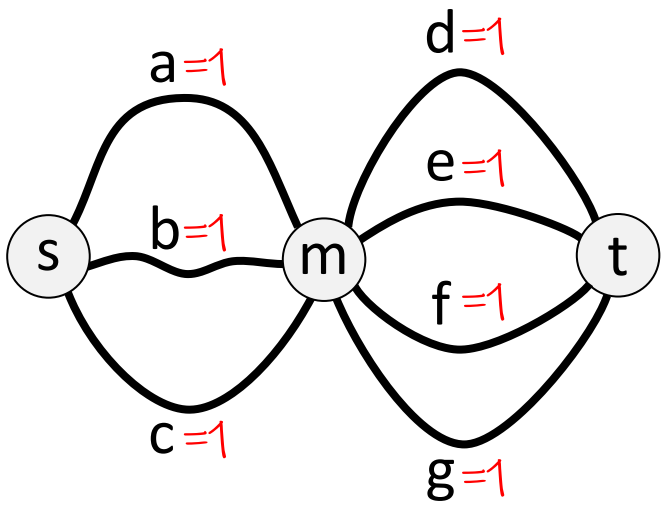
How many paths are there from s to t?

Answer: $12 = 3 \cdot 4$

Distributivity = efficient factorization

(Ring of real numbers)

- Semiring $(\mathbb{R},+,\cdot,0,1)$



How many paths are there from s to t?

Answer: $12 = 3 \cdot 4$

count $[a{\cdot}d, a{\cdot}e, a \cdot f, a \cdot g, \ldots, c \cdot g]$

count$[\underbrace{1{\cdot}1, 1{\cdot}1, 1{\cdot}1, 1{\cdot}1, \ldots, 1{\cdot}1}_{12}]$

= count [a, b, c] · count [d, e, f, g]

count[1,1,1] · count[1,1,1,1]

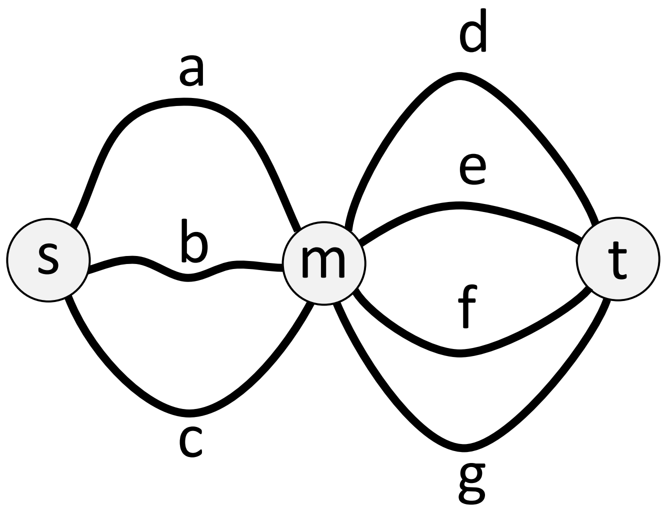
$+[x,y] \cdot z = +[x{\cdot}z, y{\cdot}z]$

(· distributes over +)

Distributivity = efficient factorization

- Semiring $(S, \oplus, \otimes, 0, 1)$

Semirings generalize this idea



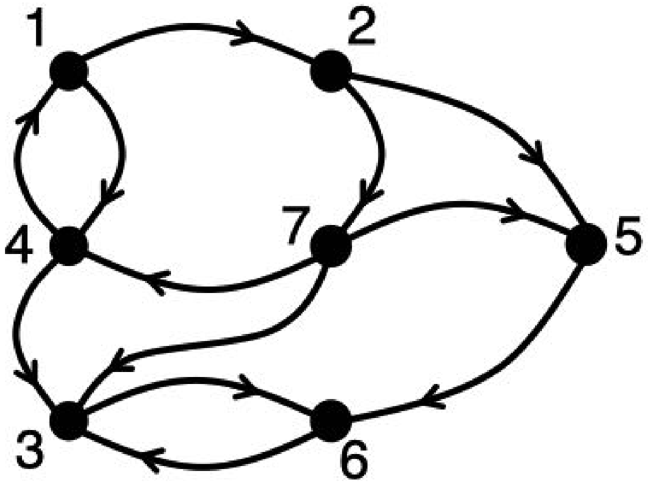
$\bigoplus[a \otimes d,\ a \otimes e,\ a \otimes f,\ a \otimes g,\ \ldots,\ c \otimes g]$

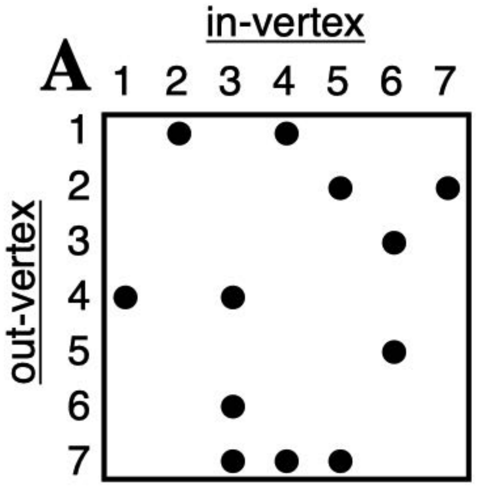
$= \bigoplus[a, b, c] \quad \otimes \quad \bigoplus[d, e, f, g]$

$\bigoplus[x, y] \otimes z = \bigoplus[x \otimes z, y \otimes z]$

($\otimes$ distributes over $\oplus$)

Matrix multiplication

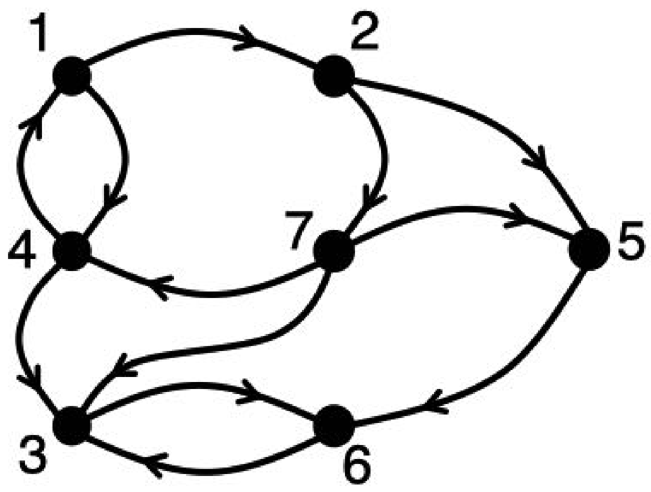


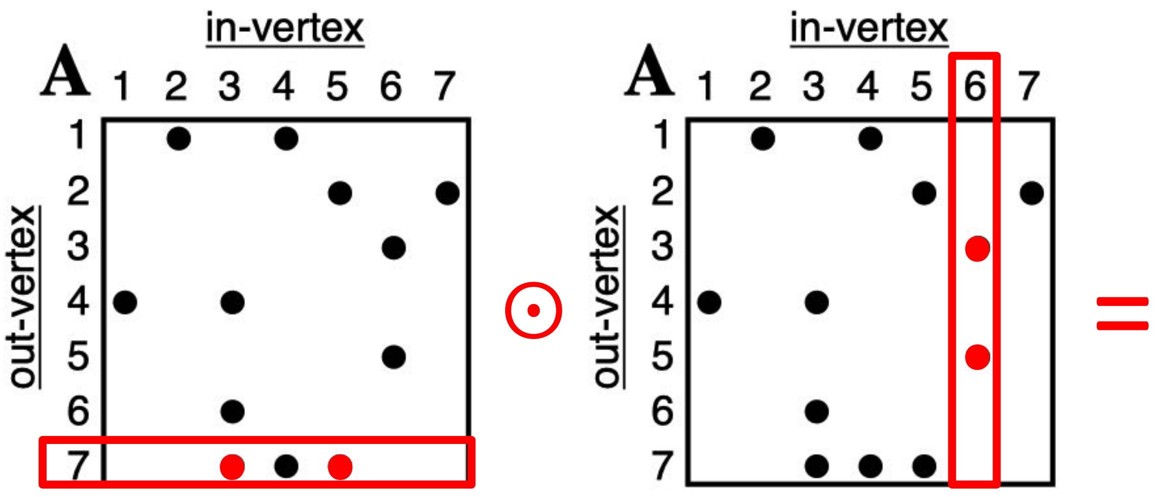


How many paths are
there from 7 to 6?

Example graph taken from "Kepner, Gilbert. Graph algorithms in the language of linear algebra, 2011"

Matrix multiplication



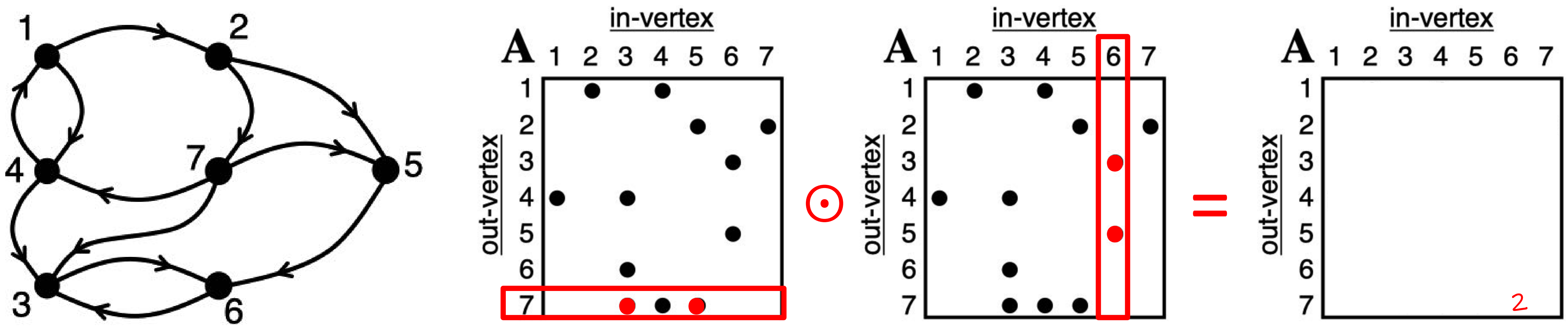


matrix multiplication

How many paths are there from 7 to 6?

Example graph taken from "Kepner, Gilbert. Graph algorithms in the language of linear algebra, 2011"

Matrix multiplication



matrix
multiplication

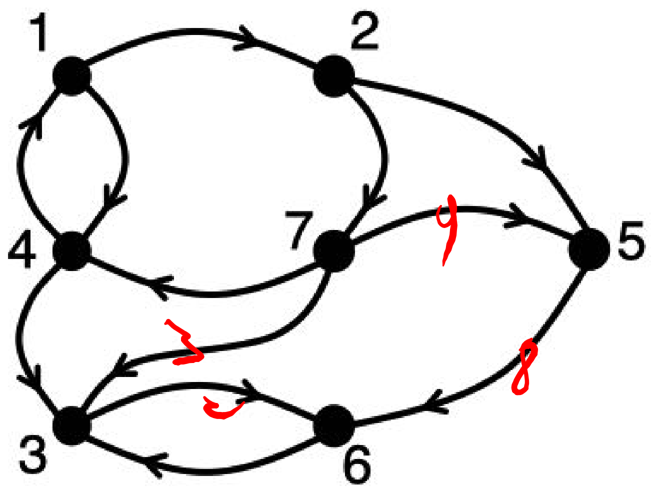
$= 0 \cdot 0 + 0 \cdot 0 + 1 \cdot 1$
$+ 1 \cdot 0 + 1 \cdot 1 + \ldots$

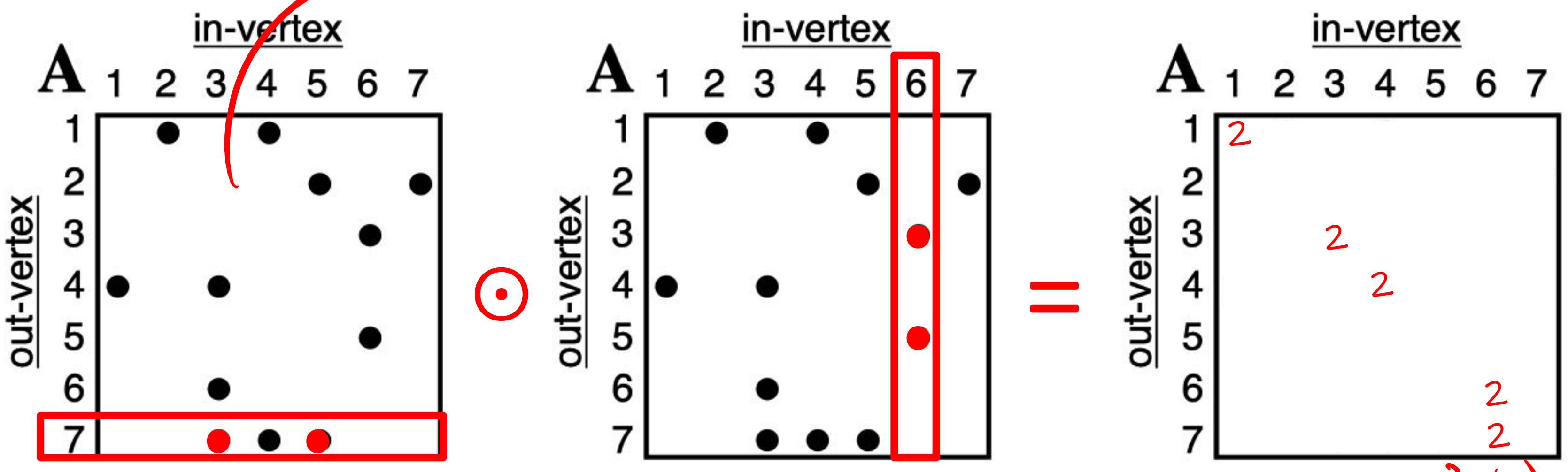
How many paths are
there from 7 to 6?

Example graph taken from "Kepner, Gilbert. Graph algorithms in the language of linear algebra, 2011"

Matrix multiplication

only diagonals and 7→6 are shown





matrix multiplication

= 0·0 + 0·0 + 1·1 + 1·0 + 1·1 + ...

How many paths are there from 7 to 6?

Shortest Path from 7 to 6 ?

Example graph taken from "Kepner, Gilbert. Graph algorithms in the language of linear algebra, 2011"

The Relational Algebra

- In the relational algebra (RA) the elements are relations
  - Recall: pairs (s,r)
- RA has 6 *primitive operators*:
  - Unary: projection, selection, renaming
  - Binary: union, difference, Cartesian product
- Each of the six is essential (*independent*)—we cannot define it using the others
  - We will see what exactly this means and how this can be proved
- In practice, we allow many more useful operators that can be defined by the primitive ones
  - For example, *intersection* via union and difference

RA vs Other QLs

- Some subtle (yet important) differences between RA and other languages
  - *Can tables have duplicate records?*
    - (RA vs. SQL)
  - *Are missing (NULL) values allowed?*
    - (RA vs. SQL)
  - *Is there any order among records?*
    - (RA vs. SQL)
  - *Is the answer dependent on the domain from which values are taken (not just the DB)?*
    - (RA vs. RC)

Recall: Virtues of the relational model

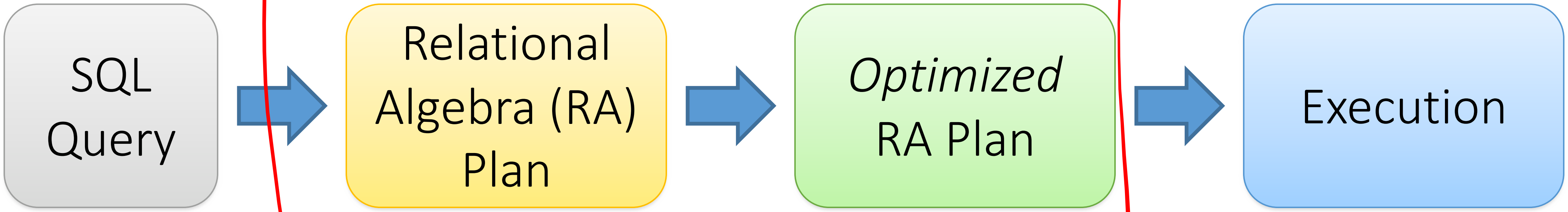
- Physical independence (logical too), Declarative

- Simple, elegant clean: Everything is a relation

- Why did it take multiple years?
  - Doubted it could be done efficiently.

RDBMS Architecture

- How does a SQL engine work ?



Declarative query (from user)

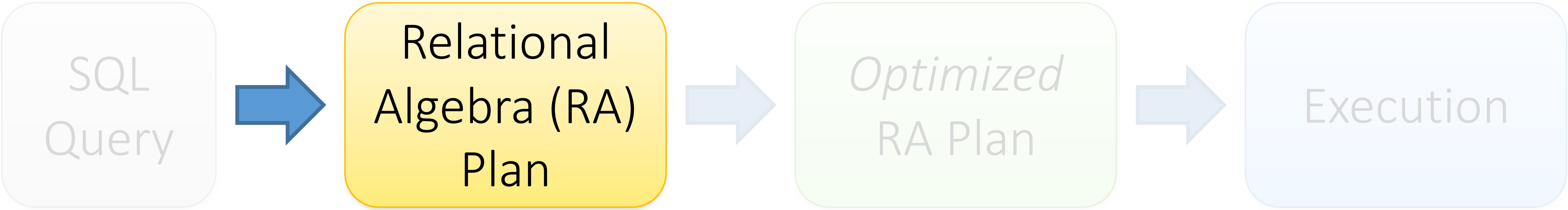
Translate to relational algebra expression

Find logically equivalent- but more efficient- RA expression

Execute each operator of the optimized plan!

RDBMS Architecture

- How does a SQL engine work ?



Relational Algebra allows us to translate declarative (SQL) queries into precise and optimizable expressions!

Queries and the connection to logic and algebra

- Why logic?
  - A crash course on FOL
- Relational Calculus
  - Syntax and Semantics
  - Domain Independence and Safety
- Relational Algebra
  - Operators
  - Independence
  - Power of algebra: optimizations
- Equivalence RC and RA

Relational Algebra (RA)

- Five basic operators:
  1. Selection: σ
  2. Projection: Π
  3. Cartesian Product: $\times$
  4. Union: $\cup$
  5. Difference*: $-$
- Auxiliary operators (sometimes counted as basic):
  6. Renaming: ρ
- Derived
  7. Intersection / complement
  8. Joins $\bowtie$ (natural,equi-join, theta join, semi-join)
  9. Division

- Extended RA
  1. Duplicate elimination δ
  2. Grouping and aggregation γ
  3. Sorting τ

All operators take in 1 or more relations as inputs and return another relation

* Relational difference $-$ is also sometimes written as \ like set difference.
$-$ is used by [Ramakrishnan+'03] and [Garcia-Molina+2014] and [Elmasri+'15]

Keep in mind: RA operates on sets!

- RDBMSs use **multisets**, however in relational algebra formalism we will consider sets!

- Also: we will consider the **named perspective**, where every attribute must have a unique name
  - →attribute order does not matter…

Now on to the basic RA operators…

1. Selection (σ)

- Returns all tuples which satisfy a condition
- Notation: $\sigma_c(R)$
- Examples
  - $\sigma_{Salary > 40000}$ (Employee)
  - $\sigma_{name = \text{“Smith”}}$ (Employee)
- The condition c can be =, <, ≤, >, ≥, <> combined with AND, OR, NOT

```
Students(sid,sname,gpa)
```

SQL:

```
SELECT *
FROM Students
WHERE gpa > 3.5;
```

RA:

$$\sigma_{gpa > 3.5}(Students)$$

Another example:

SSN	Name	Salary
1234545	John	20000
5423341	Smith	60000
4352342	Fred	50000

$\sigma_{Salary > 40000}$ (Employee)

SSN	Name	Salary
5423341	Smith	60000
4352342	Fred	50000

2. Projection (Π)

- Eliminates columns, then removes duplicates (set perspective!)
- Notation: $\Pi_{A1,\dots,An}(R)$
- Example: project social-security number and names:
  - $\Pi_{SSN,\ Name}$ (Employee)
  - Output schema: Answer(SSN, Name)

```
Students(sid,sname,gpa)
```

SQL:

```
SELECT DISTINCT
   sname,
   gpa
FROM Students;
```

RA:

$$\Pi_{sname,gpa}(Students)$$

Another example:

SSN	Name	Salary
1234545	John	20000
5423341	John	60000
4352342	John	20000

Π_{SSN}(Employee)

SSN
1234545
5423341
4352342

Another example:

SSN	Name	Salary
1234545	John	20000
5423341	John	60000
4352342	John	20000

$\Pi_{\text{Name,Salary}}$ (Employee)

Name	Salary
John	20000
John	60000

Another example:

SSN	Name	Salary
1234545	John	20000
5423341	John	60000
4352342	John	20000

$\Pi_{\text{Name,Salary}}$ (Employee)

Which is more efficient?

Bag semantics

Name	Salary
John	20000
John	60000
John	20000

Set semantics

Name	Salary
John	20000
John	60000

Composing RA Operators

"commuting operators"

Patient

no	name	zip	disease
1	p1	98125	flu
2	p2	98125	heart
3	p3	98120	lung
4	p4	98120	heart

$\pi_{zip,disease}$(Patient)

zip	disease
98125	flu
98125	heart
98120	lung
98120	heart

$\sigma_{disease='heart'}$(Patient)

no	name	zip	disease
2	p2	98125	heart
4	p4	98120	heart

$\sigma_{disease='heart'}(\pi_{zip,disease}(\text{Patient}))$

zip	disease
98125	heart
98120	heart

$\pi_{zip,disease}(\sigma_{disease='heart'}(\text{Patient}))$

Logical Equivalece of RA Plans

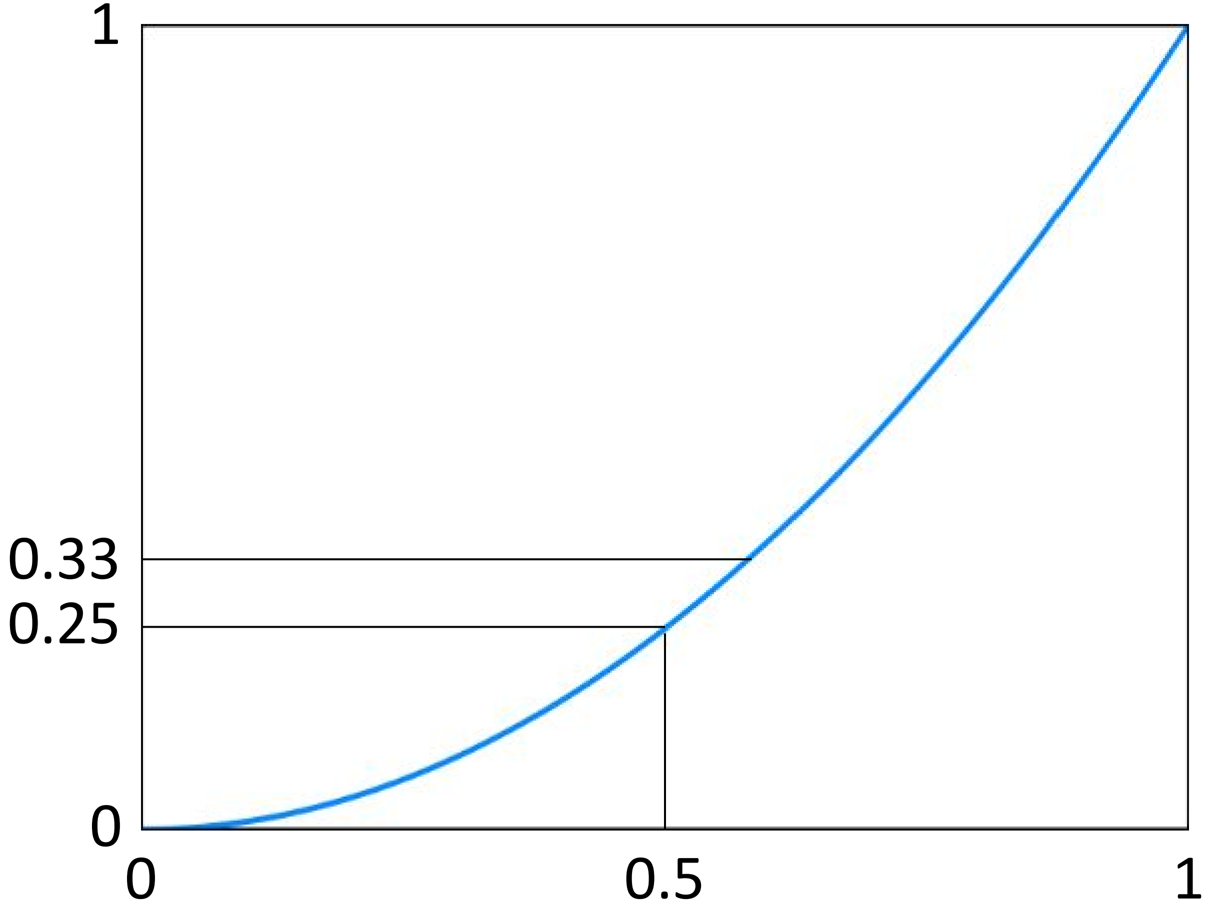
- Given relations R(A,B) and S(B,C):

  - Here, projection & selection commute:
    - $\sigma_{A=5}(\Pi_A(R)) = \Pi_A(\sigma_{A=5}(R))$

  - What about here?
    - $\sigma_{A=5}(\Pi_B(R))\ ?= \Pi_B(\sigma_{A=5}(R))$

Commuting functions: a digression

- Do functions commute with taking the expectation?
  - E[f(x)] = f(E[x]) ?
- Only for linear functions
  - Thus f(x)=ax + b
  - E[ax+b] = a E[x] + b
- Jensen's inequality for convex f
  - E[f(x)] ≥ f(E[x])
- Example f(x) = x^2
  - Assume $0 \leq x \leq 1$
  - f(E[x]) = f(0.5) = 0.25
  - $E[f(x)] = \frac{\int_0^1 f(x)}{1-0} = \frac{x^3}{3}\Big|_0^1 = 0.33$



Ratio of averages != average of ratios

	S	N	N/S
U1	1	2	2/1 = 2
U2	2	1	1/2 = 0.5
		ϕ	1.25

RA Operators are Compositional!

```
Students(sid,sname,gpa)
```

```
SELECT DISTINCT
   sname,
   gpa
FROM Students
WHERE gpa > 3.5;
```

How do we represent this query in RA?

$$\Pi_{sname,gpa}(\sigma_{gpa>3.5}(Students))$$

$$\Updownarrow$$

$$\sigma_{gpa>3.5}(\Pi_{sname,gpa}(Students))$$

Are these logically equivalent?

3. Cross-Product (×)

- Each tuple in R1 with each tuple in R2
- Notation: R1 × R2
- Example:
  - Employee × Dependents
- Rare in practice; mainly used to express joins

```
Students(sid,sname,gpa)
People(ssn,pname,address)
```

SQL:

```
SELECT *
FROM Students, People;
```

RA:

$$Students \times People$$

Another example:

People

ssn	pname	address
1234545	John	216 Rosse
5423341	Bob	217 Rosse

×

Students

sid	sname	gpa
001	John	3.4
002	Bob	1.3

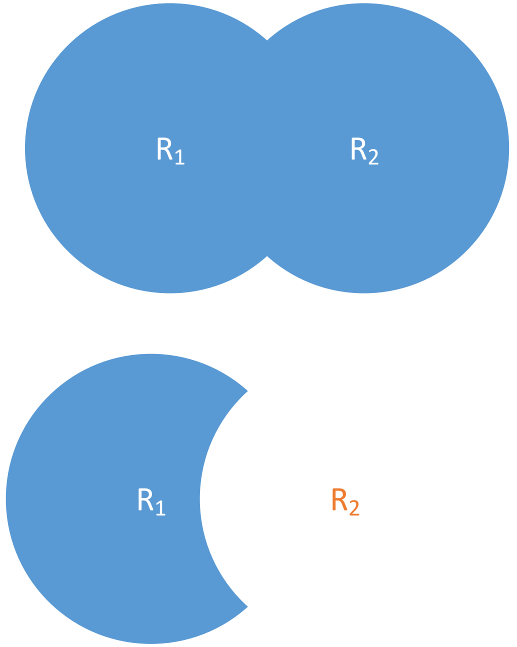
$$Students \times People$$

ssn	pname	address	sid	sname	gpa
1234545	John	216 Rosse	001	John	3.4
5423341	Bob	217 Rosse	001	John	3.4
1234545	John	216 Rosse	002	Bob	1.3
5423341	Bob	216 Rosse	002	Bob	1.3

4. Union (∪) and 5. Difference (–)

Students (<u>neuid</u>, fname, lname)
Faculty (<u>neuid</u>, fname, lname, college)

R1 ∪ R2
R1 – R2



- Examples:
  - ActiveEmployees ∪ RetiredEmployees
  - AllEmployees – RetiredEmployees

Only make sense if R1, R2 have the same schema

What do they mean over bags ?

6. Renaming (ρ)

- Changes the schema, not the instance
- A ‘special’ operator- neither basic nor derived
- Notation: $\rho_{B1,\ldots,Bn}(R)$

- Note: this is <u>shorthand</u> for the proper form (since <u>names, not order</u> matters!):
  - $\rho_{A1 \rightarrow B1,\ldots,An \rightarrow Bn}(R)$

```
Students(sid,sname,gpa)
```

SQL:

```
SELECT
  sid AS studId,
  sname AS name,
  gpa AS gradePtAvg
FROM Students;
```

RA:

$$\rho_{studId,name,gradePtAvg}(Students)$$

We care about this operator *because* we are working in a *named perspective*

Another example:

Students

sid	sname	gpa
001	John	3.4
002	Bob	1.3

$$\rho_{studId,name,gradePtAvg}(Students)$$

Students

studId	name	gradePtAvg
001	John	3.4
002	Bob	1.3

Why renaming

R

A	B
1	2
3	4

S

B	C	D
2	5	6
4	7	8
9	10	11

R × S

A	R.B	S.B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

$\rho_{B \to E}$(R) × S

A	E	B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

What if we have R × R?

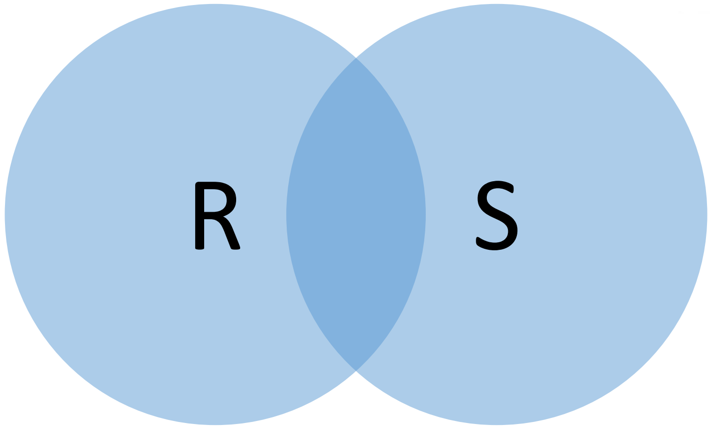
Implied Operators

- Derived relational operators
  - Not among the 5 basic operators (sometimes 6 if renaming counted)
  - Can be expressed in RA (implied)
  - Very common in practice
- Enhancing the available operator set with the implied operators is a good idea!
  - Easier to write queries
  - Easier to understand/maintain queries
  - Easier for DBMS to apply specialized optimizations

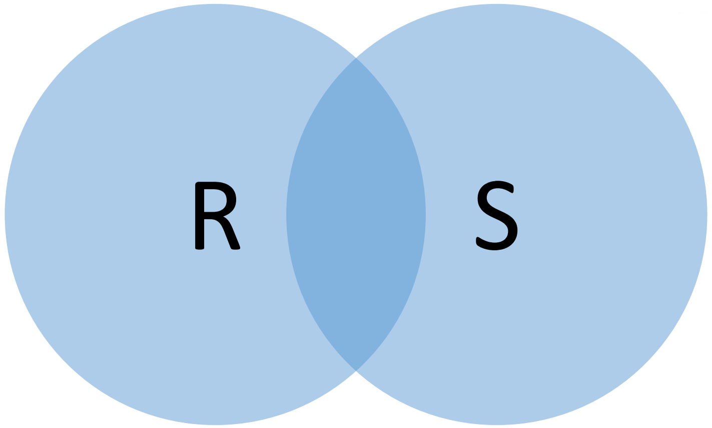
7. What about Intersection ∩?

- As derived operator using union and minus

?



7. What about Intersection ∩?



- As derived operator using union and minus

$$R \cap S = ((R \cup S) - (R - S)) - (S - R)$$

- Derived operator using minus only!

?

7. What about Intersection ∩?

- As derived operator using union and minus

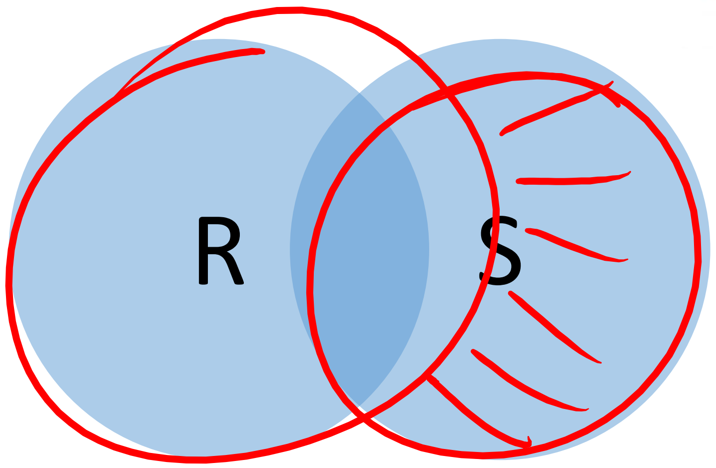
$$R \cap S = ((R \cup S) - (R - S)) - (S - R)$$

- Derived operator using minus only!

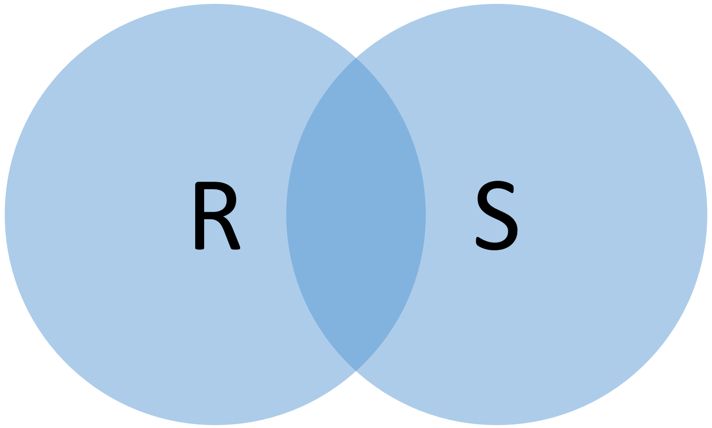
$$R \cap S = \quad R \quad - (S - R)$$

- Derived using join

?



7. What about Intersection ∩?



- As derived operator using union and minus

  $R \cap S = ((R \cup S) - (R - S)) - (S - R)$?

- Derived operator using minus only!

  $R \cap S = \quad R \quad - (S - R)$

- Derived using join

  $R \cap S = R \bowtie S$

- Example
  - UnionizedEmployees ∩ RetiredEmployees

8 Joins: Overview

- Natural join
- Theta-join
- Equi-join (most important)

8a. Natural Join (⋈)

- Notation: $R_1 \bowtie R_2$
- Joins R_1 and R_2 on equality of all shared attributes
  - If R_1 has attribute set A, and R_2 has attribute set B, and they share attributes A⋂B = C, can also be written: $R_1 \bowtie_C R_2$
- Our first example of a derived RA operator:
  - Meaning: $R_1 \bowtie R_2 = \Pi_{A \cup B}(\sigma_{R1.C=R2.C}(R_1 \times R_2))$
  - Meaning: $R_1 \bowtie R_2 = \Pi_{A \cup B}(\sigma_{C=D}(\rho_{C \to D}(R_1) \times R_2))$
  - Where:
    - The rename $\rho_{C \to D}$ renames the shared attributes in one of the relations
    - The selection $\sigma_{C=D}$ checks equality of the shared attributes
    - The projection $\Pi_{A \cup B}$ eliminates the duplicate common attributes

```
Students(sid,name,gpa)
People(ssn,name,address)
```

SQL:

```
SELECT DISTINCT
  ssid, S.name, gpa,
  ssn, address
FROM
  Students S,
  People P
WHERE S.name = P.name;
```

RA:

$$Students \bowtie People$$

An example

R

A	B
1	2
3	4

S

B	C	D
2	5	6
4	7	8
9	10	11

$R \bowtie S$

A	B	C	D
1	2	5	6
3	4	7	8

$R \bowtie S =$

$\Pi_{ABC}(\sigma_{R.B=S.B}(R \times S)) =$

$\Pi_{AR.BC}(\sigma_{R.B=S.B}(R \times S)) =$

$\Pi_{ABC}(\sigma_{B=E}(\rho_{B\to E}(R) \times S))$

$\rho_{B\to E}(R) \times S$

A	E	B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

Natural Join practice

- Given schemas R(A, B, C, D), S(A, C, E), what is the schema of R ⋈ S ?

- Given R(A, B, C), S(D, E), what is R ⋈ S ?

- Given R(A, B), S(A, B), what is R ⋈ S ?

8b. Theta Join ($\bowtie_\theta$)

- A join that involves a predicate

$$R1 \bowtie_\theta R2 \quad = \quad \sigma_\theta \,(R1 \times R2)$$

- Here θ can be any condition
- No projection in this case!
- Example

AnonPatient (age, zip, disease)
Voters (name, age, zip)

$$P \bowtie_{P.zip = V.zip \text{ and } P.age >= V.age - 1 \text{ and } P.age <= V.age + 1} V$$

> Note that natural join is a theta join + a projection.

```
Students(sid,sname,gpa)
People(ssn,pname,address)
```

SQL:

```
SELECT *
FROM
   Students,People
WHERE θ;
```

RA:

$$Students \bowtie_\theta People$$

8c. Equi-join ($\bowtie_{A=B}$)

- A theta join where q is an equality
- $R1 \bowtie_{A=B} R2 \;=\; \sigma_{A=B}(R1 \times R2)$
- Example:
  - Employee $\bowtie_{SSN=SSN}$ Dependents

Most common join in practice!

What is the connection with natural join?

```
Students(sid,sname,gpa)
People(ssn,pname,address)
```

SQL:

```
SELECT *
FROM
  Students S,
  People P
WHERE sname = pname;
```

RA:

$$S \bowtie_{sname=pname} P$$

Join Summary

- **Theta-join**: $R \bowtie_\theta S = \sigma_\theta (R \times S)$
  - Join of R and S with a join condition θ
  - Cross-product followed by selection θ
  - No projection
- **Equijoin**: $R \bowtie_\theta S = \sigma_\theta (R \times S)$
  - Join condition θ consists only of equalities
  - No projection
- **Natural join**: $R \bowtie S = \pi_A (\sigma_\theta (R \times S))$
  - Equality on **all** fields with same name in R and in S
  - Projection π_A drops all redundant attributes

Some Examples

Supplier(<u>sno</u>,sname,scity,sstate)
Part(<u>pno</u>,pname,psize,pcolor)
Supply(<u>sno,pno</u>,qty,price)

Name of supplier of parts with size greater than 10

$\pi_{sname}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{psize>10}\ (\text{Part}))$

Name of supplier of red parts or parts with size greater than 10

$\pi_{sname}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{psize>10}\ (\text{Part}) \cup \sigma_{pcolor='red'}\ (\text{Part})\ )\ )$

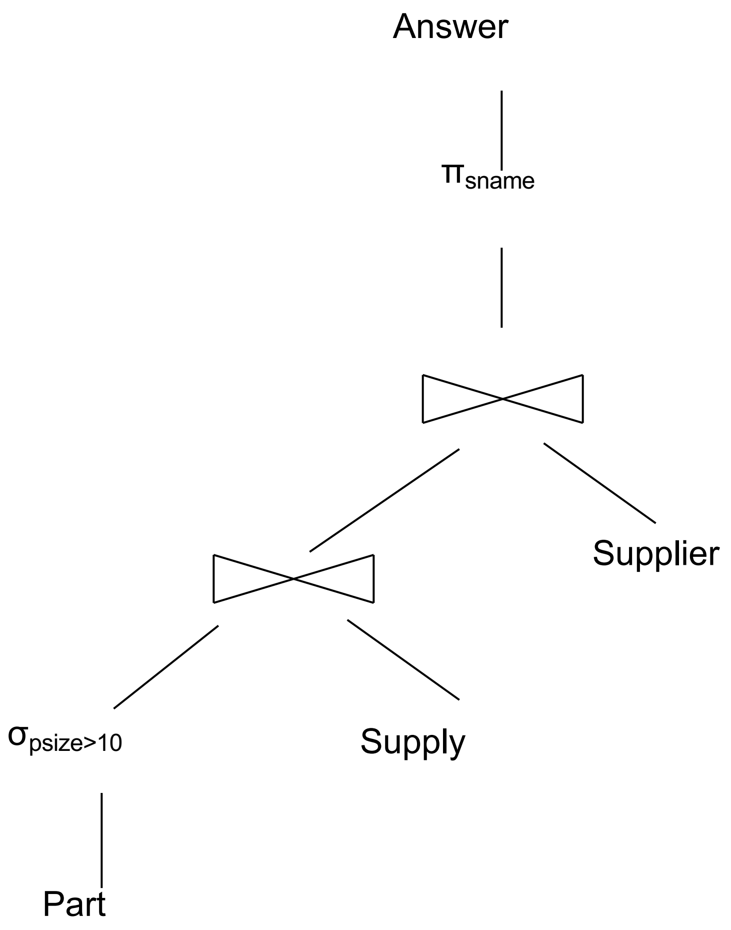
$\pi_{sname}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{psize>10\ \vee\ pcolor='red'}\ (\text{Part})\ )\ )$

Can be represented as trees as well

Representing RA Queries as Trees

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,qty,price)

π_{sname}(Supplier ⋈ Supply ⋈ ($\sigma_{psize>10}$ (Part))



Example: Converting SFW Query -> RA

```
Students(sid,name,gpa)
People(ssn,name,address)
```

```
SELECT DISTINCT
   gpa,
   address
FROM Students S,
     People P
WHERE gpa > 3.5 AND
   S.name = P.name;
```

$$\Pi_{gpa,address}(\sigma_{gpa>3.5}(S \bowtie P))$$

How do we represent this query in RA?

9. Division

- Consider two relations R(X,Y) and S(Y)
  - Here, X and Y are tuples of attributes

- R ÷ S is the relation T(X) that contains all the Xs that occur with *every* Y in S

Formal Definition

- **Legal input:** (R,S) such that R has all the attributes of S
- R÷S is the relation T with:
  - The header of R, with all attributes of S removed
  - Tuple set {t[**X**] | t[**X**,**Y**]∈R for every s[**Y**]∈S}
    - *This is an abuse of notation, since the attributes in* X *need not necessarily come before those of* Y

Questions

Studies

sid	student	course
1	Alice	AI
1	Alice	DB
2	Bob	DB
2	Bob	ML
3	Charly	AI
3	Charly	DB
3	Charly	ML

÷ *Course*

course
ML

= ?

÷

course
AI
DB
ML

= ?

Questions

recall set semantics for RA

Studies

sid	student	course
1	Alice	AI
1	Alice	DB
2	Bob	DB
2	Bob	ML
3	Charly	AI
3	Charly	DB
3	Charly	ML

÷

Course

course
ML

=

sid	student
2	Bob
3	Charly

÷

course
AI
DB
ML

=

sid	student
3	Charly

(RxS)÷S = ?

(RxS)÷R = ?

Questions

recall set semantics for RA

Studies

sid	student	course
1	Alice	AI
1	Alice	DB
2	Bob	DB
2	Bob	ML
3	Charly	AI
3	Charly	DB
3	Charly	ML

÷ *Course*

course
ML

=

sid	student
2	Bob
3	Charly

÷

course
AI
DB
ML

=

sid	student
3	Charly

R,S have disjoint attribute sets

(RxS)÷S = R

(RxS)÷R = S

Q: If R has 1000 tuples and S has 100 tuples, how many tuples can be in R÷S?

Q: If R has 1000 tuples and S has 1001 tuples, how many tuples can be in R÷S?

Questions

Studies

sid	student	course
1	Alice	AI
1	Alice	DB
2	Bob	DB
2	Bob	ML
3	Charly	AI
3	Charly	DB
3	Charly	ML

CourseType

course	type
AI	elective
DB	core
ML	core

Who took all core courses in RA?

?

Questions

Studies

sid	student	course
1	Alice	AI
1	Alice	DB
2	Bob	DB
2	Bob	ML
3	Charly	AI
3	Charly	DB
3	Charly	ML

CourseType

course	type
AI	elective
DB	core
ML	core

Who took all core courses in RA?

$$\text{Studies} \div \pi_{\text{course}} \sigma_{\text{type='core'}} \text{CourseType}$$

How to write R÷S in Primitive RA?

$$R(X,Y) \div S(Y)$$

?

$R \div S = Q$

R:

A	B
a	0
a	1
a	2
b	1

S:

B
1
2

Q:

A
a

How to write R÷S in Primitive RA?

$R(X,Y) \div S(Y)$

?

$R \div S = Q$

A	B
a	0
a	1
a	2
b	1
b	2

B
1
2

A
a

1 (rows a 1 through b 2)

2 (row b 2)

3 (value b)

4: {a} = {a,b} – {b}

How to write R÷S in Primitive RA?

$$R(X,Y) \div S(Y)$$

$$\pi_X R \times S$$

Each X of R w/ each Y of S

R ÷ S = Q

A	B
a	0
a	1
a	2
b	1
b	2

B
1
2

A
a

1

2

3

4: {a} = {a,b} – {b}

How to write R÷S in Primitive RA?

$$R(X,Y) \div S(Y)$$

$$(\pi_X R \times S) - R$$

Each X of R w/ each Y of S

(X,Y) s.t. X in R, Y in S, but (X,Y) not in R

R ÷ S = Q

R:

A	B
a	0
a	1
a	2
b	1
b	2

S:

B
1
2

Q:

A
a

1 (brace over rows a 1, a 2, b 1, b 2)

2 (row b 2)

3 (b circled)

4: {a} = {a,b} – {b}

How to write R÷S in Primitive RA?

$$R(X,Y) \div S(Y)$$

$$\pi_X\left(\left(\pi_X R \times S\right) - R\right)$$

Each X of R w/ each Y of S

(X,Y) s.t. X in R, Y in S, but (X,Y) not in R

Xs in R where for some Y in S, (X,Y) is not in R

R ÷ S = Q

R:

A	B
a	0
a	1
a	2
b	1
b	2

S:

B
1
2

Q:

A
a

1

2

3

4: {a} = {a,b} – {b}

How to write R÷S in Primitive RA?

$$R(X,Y) \div S(Y)$$

$$\pi_X R - \pi_X\Big(\big(\pi_X R \times S\big) - R\Big)$$

Each X of R w/ each Y of S

(X,Y) s.t. X in R, Y in S, but (X,Y) not in R

Xs in R where for some Y in S, (X,Y) is not in R

R÷S

$R \div S = Q$

R:

A	B
a	0
a	1
a	2
b	1
b	2

S:

B
1
2

Q:

A
a

1

2

3

4: {a} = {a,b} – {b}

R÷S in Primitive RA vs. RC

$$R(X,Y) \div S(Y)$$

In RA:

$$\pi_X R - \pi_X((\pi_X R \times S) - R)$$

In DRC: ?

R ÷ S = Q

R:

A	B
a	0
a	1
a	2
b	1
b	2

S:

B
1
2

Q:

A
a

R÷S in Primitive RA vs. RC

$$R(X,Y) \div S(Y)$$

R ÷ S = Q

R:

A	B
a	0
a	1
a	2
b	1
b	2

S:

B
1
2

Q:

A
a

In RA:

$$\pi_X R - \pi_X\big((\pi_X R \times S) - R\big)$$

In DRC: what if $S(Y)=\emptyset$?

$$\{\, X \mid \exists Z.[R(X,Z)] \land \forall Y.[S(Y) \rightarrow R(X,Y)] \,\}$$

? without universal quantification

R÷S in Primitive RA vs. RC

$$R(X,Y) \div S(Y)$$

R ÷ S = Q

R:

A	B
a	0
a	1
a	2
b	1
b	2

S:

B
1
2

Q:

A
a

In RA:

$$\pi_X R - \pi_X\big((\pi_X R \times S) - R\big)$$

In DRC: what if $S(Y)=\emptyset$?

$$\{\, X \mid \exists Z.[R(X,Z)] \land \forall Y.[S(Y) \rightarrow R(X,Y)] \,\}$$

$$\{\, X \mid \exists Z.[R(X,Z)] \land \nexists Y.[S(Y) \land \neg R(X,Y)] \,\}$$

In TRC:

?

R÷S in Primitive RA vs. RC

$$R(X,Y) \div S(Y)$$

R	
A	B
a	0
a	1
a	2
b	1
b	2

÷

S
B
1
2

=

Q
A
a

In RA:

$$\pi_X R - \pi_X\Big((\pi_X R \times S) - R\Big)$$

In DRC: what if S(Y)=∅ ?

$$\{ X \mid \exists Z.[R(X,Z)] \land \forall Y.[S(Y) \rightarrow R(X,Y)] \}$$

$$\{ X \mid \exists Z.[R(X,Z)] \land \nexists Y.[S(Y) \land \neg R(X,Y)] \}$$

? in SQL

In TRC:

$$\{ r.A \mid \exists r \in R.[\nexists s \in S.[\nexists r_2 \in R.[r_2.B = s.B \land r_2.A = r.A]] \}$$

R÷S in Primitive RA vs. RC

In SQL

```
SELECT DISTINCT R.A
FROM R
WHERE not exists (
        SELECT *
        FROM S
        WHERE not exists (
                SELECT *
                FROM R AS R2
                WHERE R2.B=S.B
                AND R2.A=R.A))
```

R ÷ S = Q

R:

A	B
a	0
a	1
a	2
b	1
b	2

S:

B
1
2

Q:

A
a

In TRC:

$$\{ r.A \mid \exists r \in R.[\, \nexists s \in S.[\nexists r_2 \in R.[r_2.B = s.B \wedge r_2.A = r.A)] \}$$