Topic 1: Data models and query languages
Unit 3: Relational Algebra (RA)
Lecture 6

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp23)

https://northeastern-datalab.github.io/cs7240/sp23/

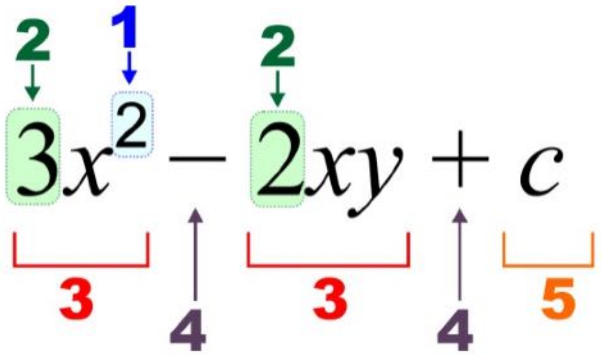
1/27/2023

Algebra and the connection to logic and queries

- Algebra
- Relational Algebra
  - Operators
  - Independence
  - Power of algebra: optimizations
- Equivalence RA and safe RC (Codd's theorem)

What is “Algebra”?

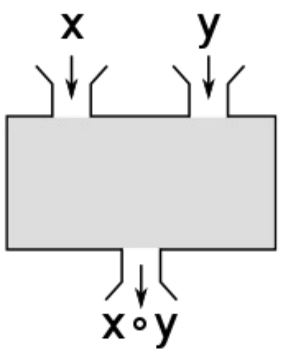
- Algebra is the study of mathematical symbols and the rules for manipulating these symbols
  - e.g., Linear Algebra
  - e.g., Relational Algebra
  - e.g., Boolean Algebra
  - e.g., Elementary algebra
  - e.g., Abstract algebra (groups, rings, fields, ...)



1 – Exponent (power), 2 – coefficient, 3 – term, 4 – operator, 5 – constant, x, y - variables

Picture source: https://en.wikipedia.org/wiki/Algebraic_expression
Also watch "What is abstract algebra?", Socratica, 2016: https://www.youtube.com/watch?v=IP7nW_hKB7I

What is “Abstract Algebra”?

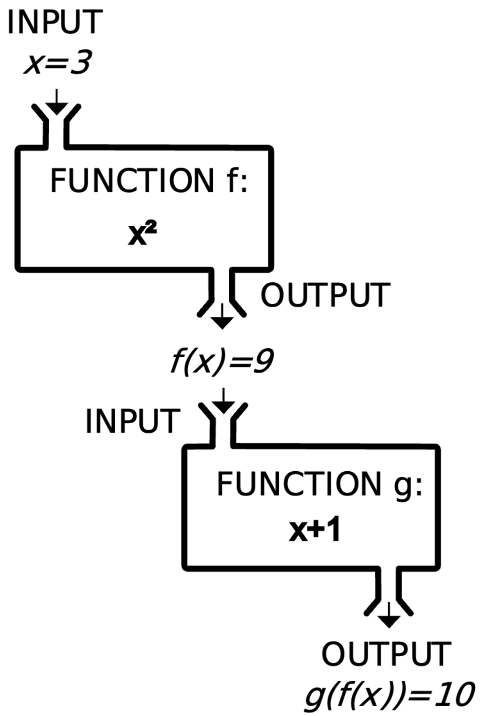


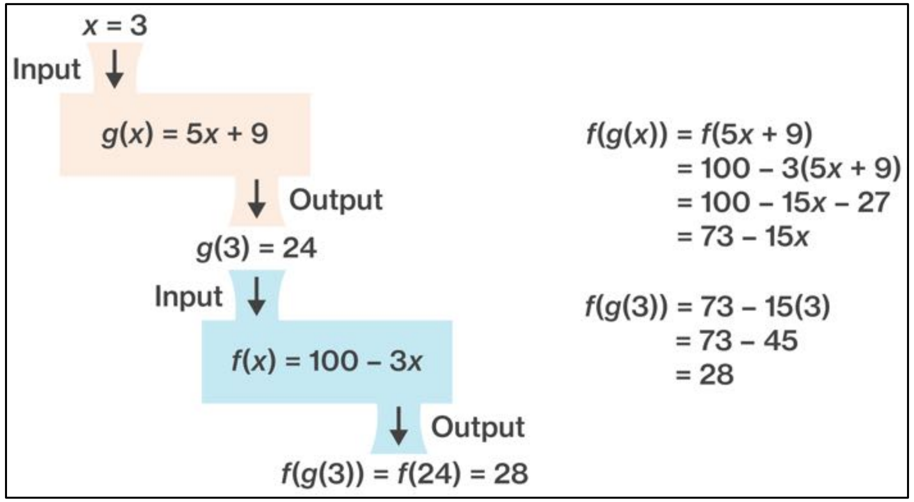
- Abstract algebra: studies algebraic structures, which consist of:
  - A domain (i.e. a set of elements)
  - A collection of operators
    - each of arity d; maps a domain of sequences $(x_1,\ldots,x_d)$ to an element y of its codomain (usually that is also the domain)
  - A set of axioms (or identities) that these operators must satisfy.
    - e.g. commutativity: $x \oplus y \equiv y \oplus x$ or $\oplus(x,y) \equiv \oplus(y, x)$ or $op(x,y) \equiv op(y,x)$

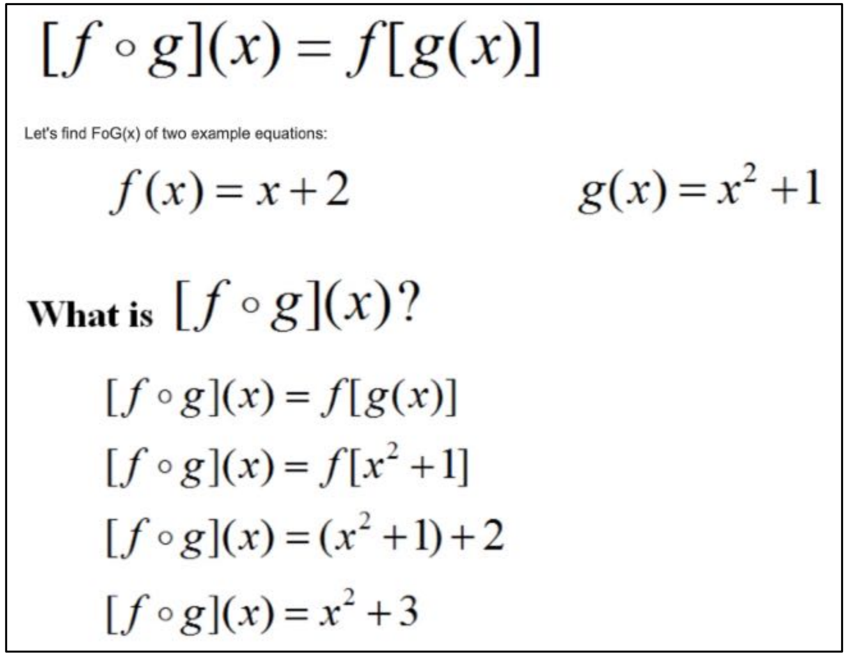
- Examples:
  - Boolean algebra: $(\{true,false\},\{\wedge,\vee,\neg\})$
  - Ring of integers: $(\mathbb{Z},\{+,\cdot\})$ ← ring: set equipped with two binary operations with certain properties like distributivity of multiplication over addition
  - Relational algebra

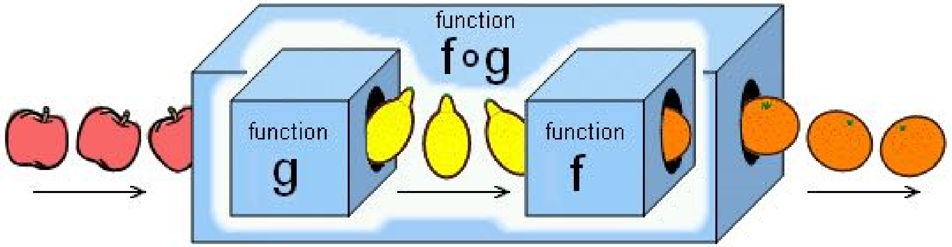
- The definition of an operator allows for composition:
  - e.g. $op_1\big(op_2(x),op_1(y,op_4(x,z))\big)$

Function composition

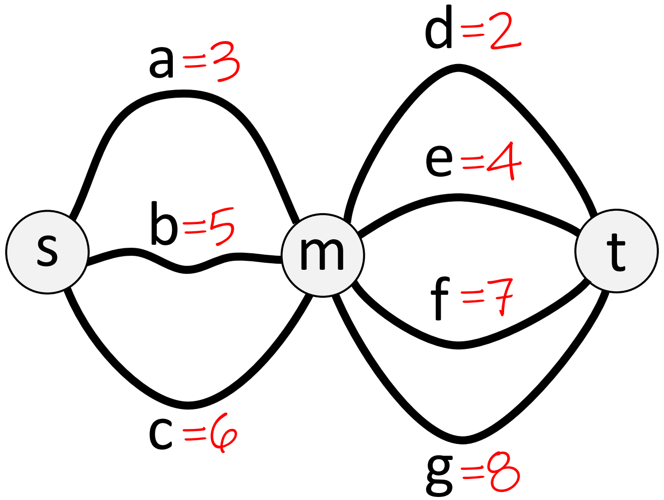








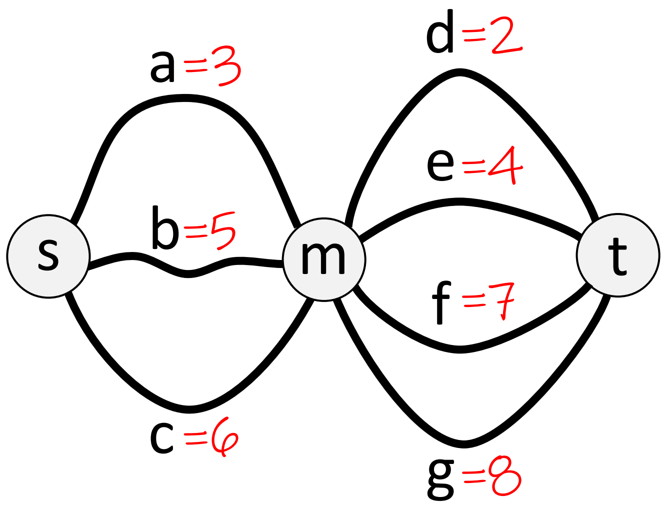
Distributivity = efficient factorization



What is the shortest path from s to t?

?

Distributivity = efficient factorization



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

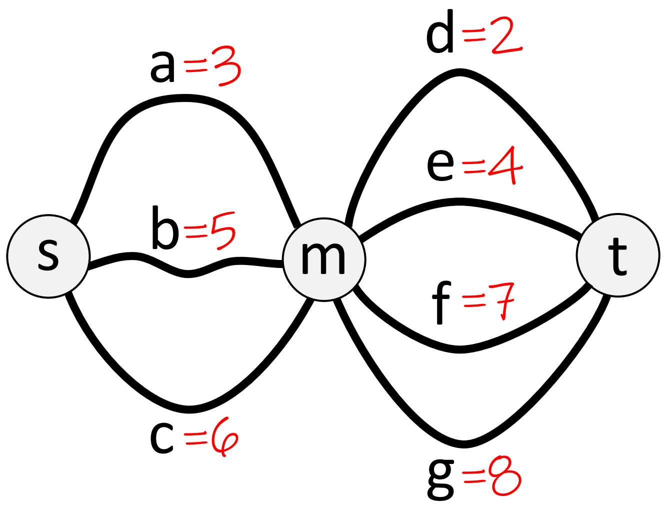
min[3+2, 3+4, 3+7, 3+8, ..., 6+8]

?

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)]

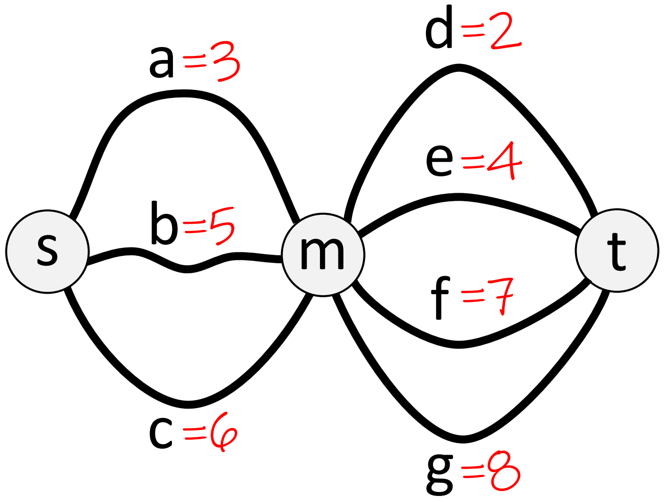
(+ distributes over min)

Distributivity = efficient factorization

(Tropical semiring)

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

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]

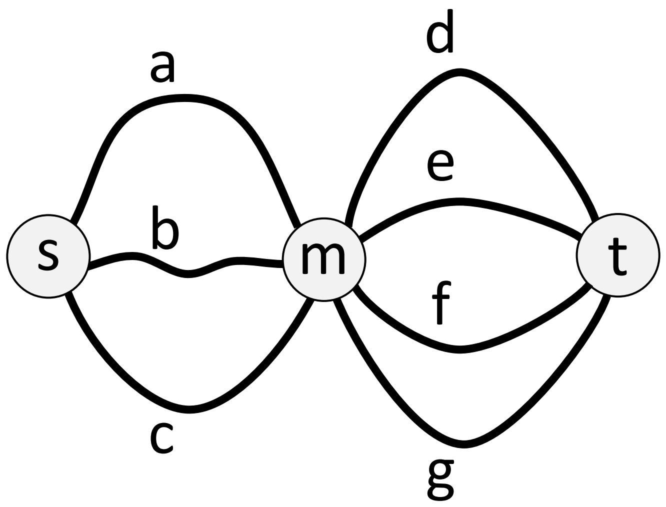
What is the shortest path from s to t?

Answer: 5 = 3 + 2

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

(+ distributes over min)

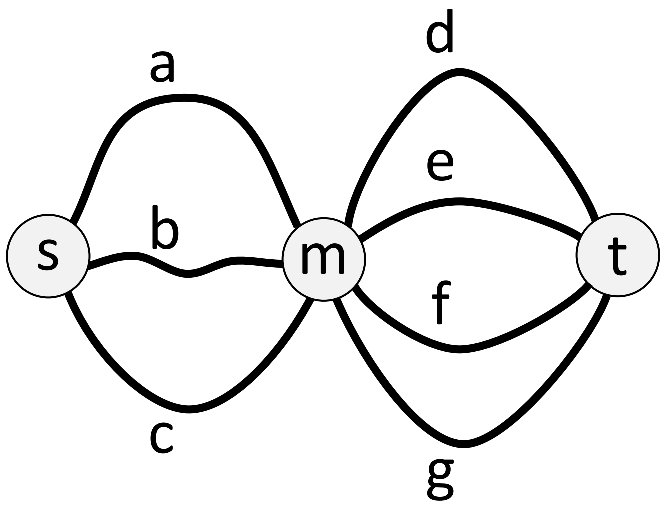
Distributivity = efficient factorization



How many paths are
there from s to t?

?

Distributivity = efficient factorization



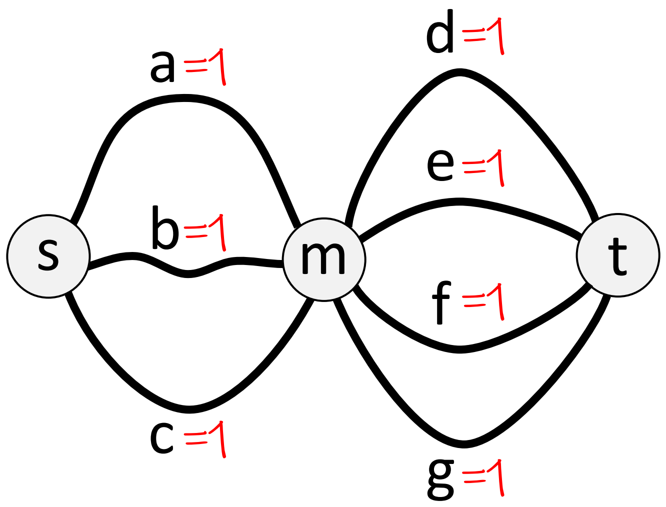
How many paths are there from s to t?

Answer: 12 = 3 · 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 · 4

count [a·d, a·e, a · f, a · g, ..., c · g]

count[1·1, 1·1, 1·1, 1·1, ..., 1·1] (12)

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

count[1,1,1] · count[1,1,1,1]

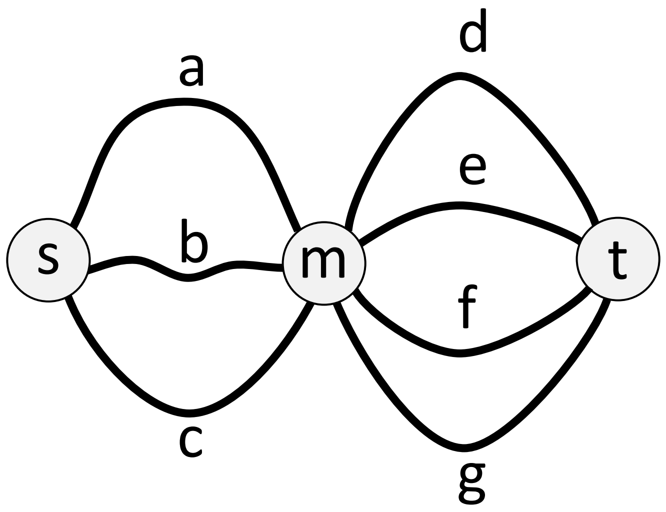
+[x,y] · z = +[x·z,y·z]

(· distributes over +)

Distributivity = efficient factorization

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

Semirings generalize this idea



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

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

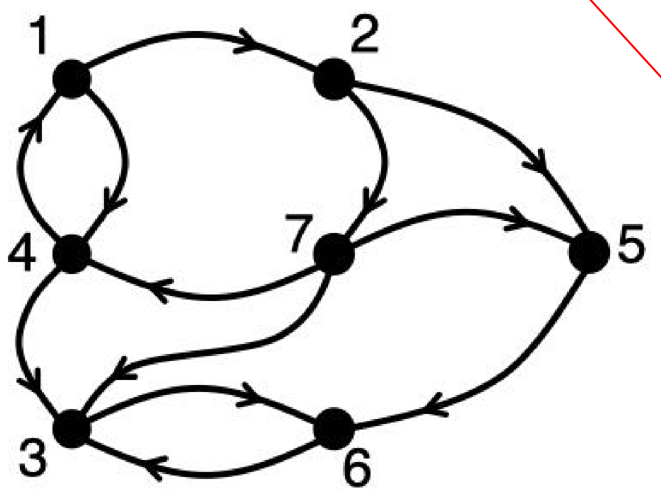
$\bigoplus[x, y] \otimes z = \bigoplus[x \otimes z, y \otimes z]$

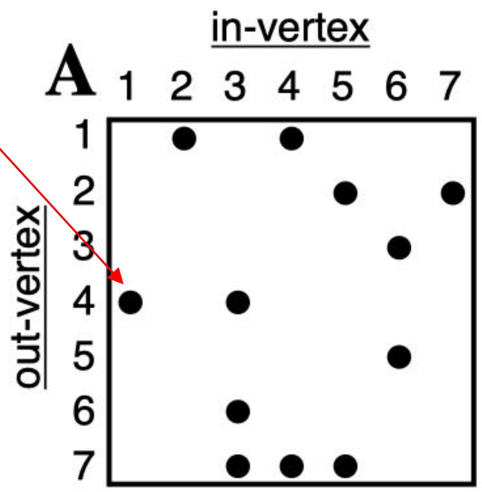
($\otimes$ distributes over $\oplus$)

Matrix multiplication

A... Adjacency matrix, or Arcs

think of dots as "1"s





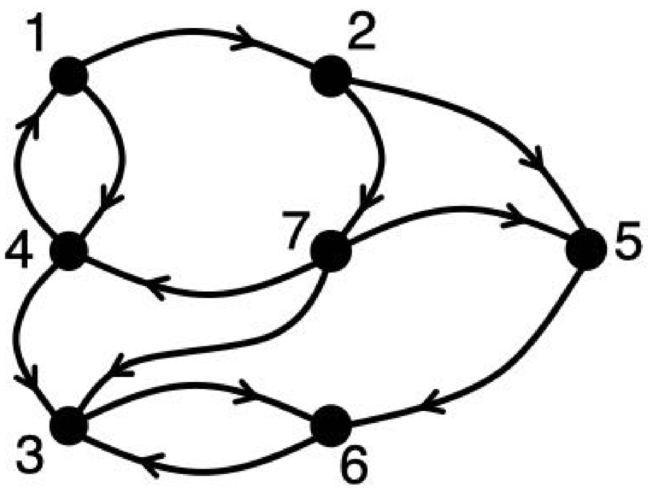
How many paths of length 2 are there from 7 to 6?

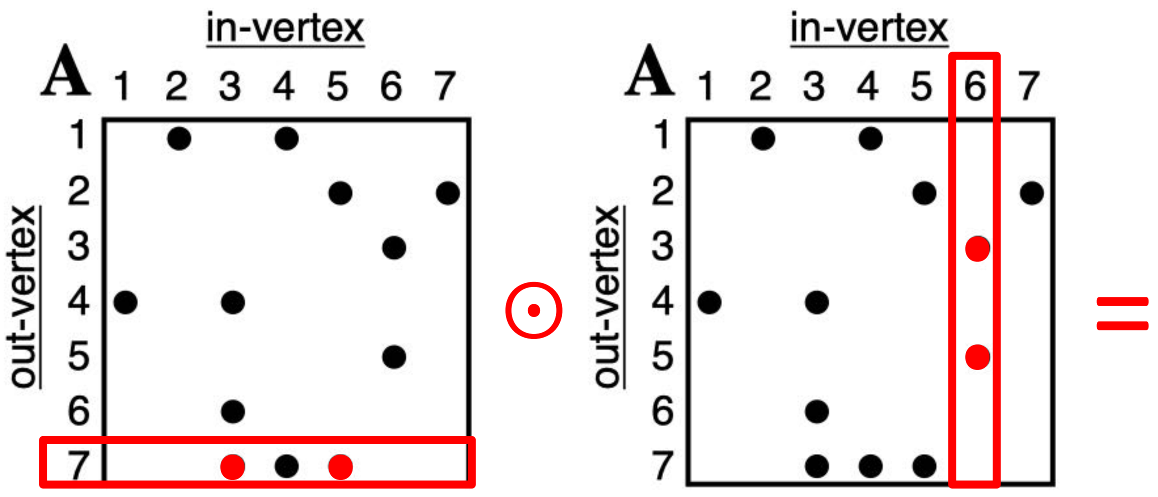
?

Example graph taken from "Kepner, Gilbert. Graph algorithms in the language of linear algebra, 2011" https://doi.org/10.1137/1.9780898719918

Matrix multiplication

A... Adjacency matrix, or Arcs





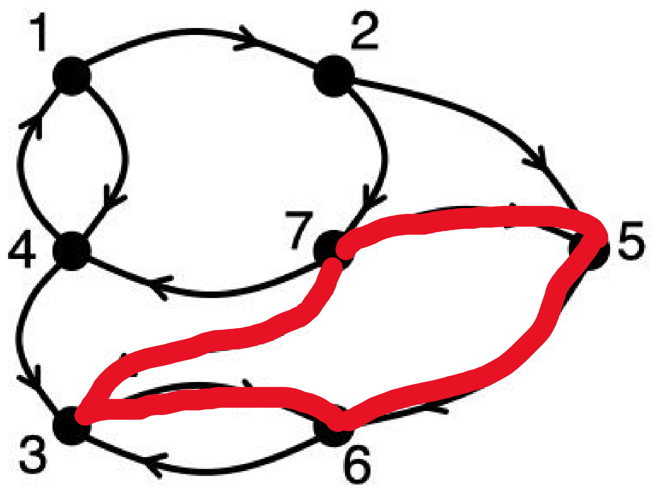
matrix
multiplication

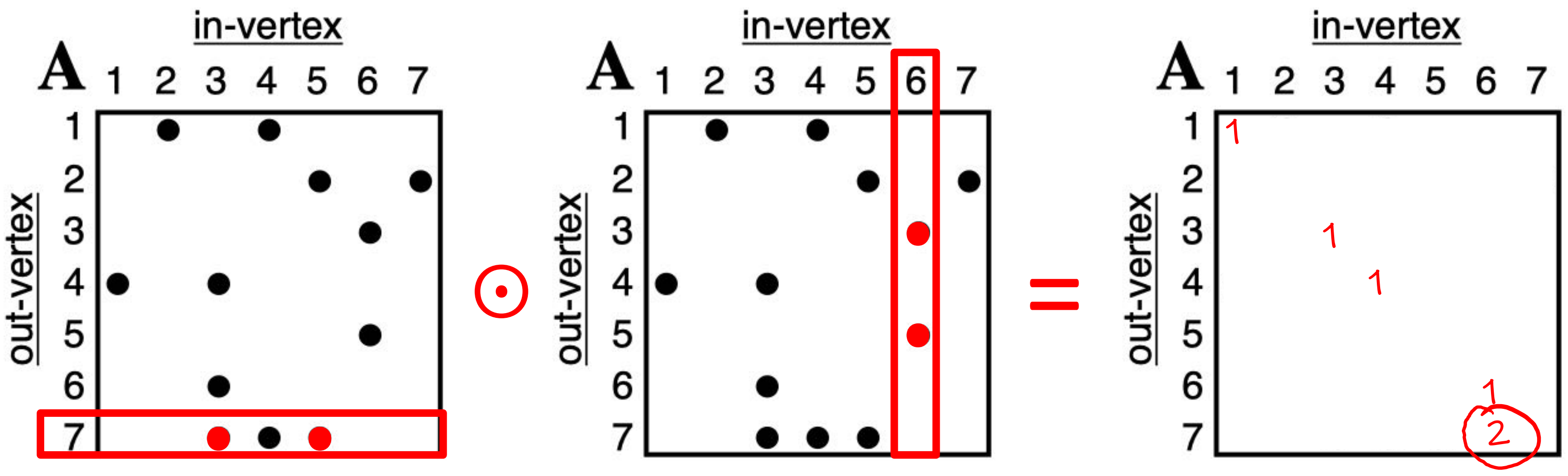
How many paths of
length 2 are there
from 7 to 6?

Matrix multiplication

A... Adjacency matrix, or Arcs

only diagonals and 7→6 are shown





matrix multiplication

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

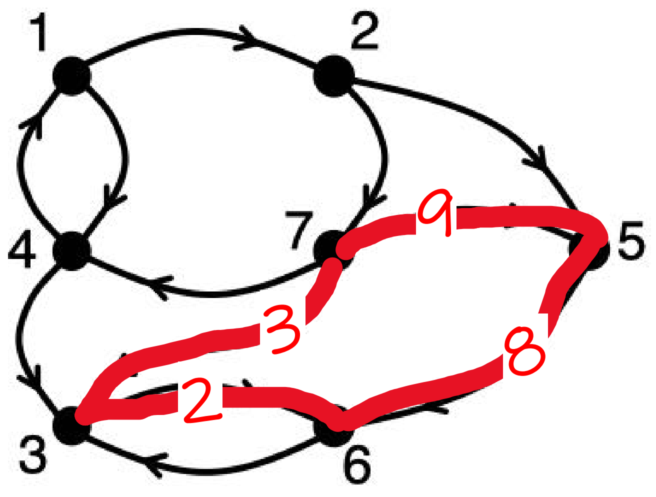
How many paths of length 2 are there from 7 to 6?

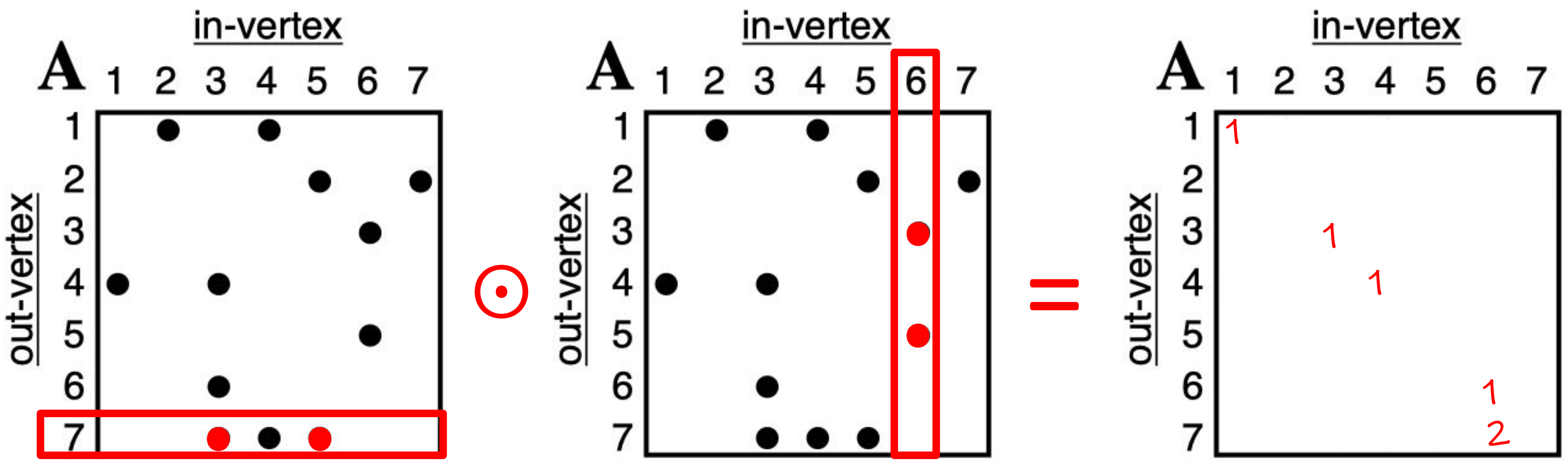
Example graph taken from "Kepner, Gilbert. Graph algorithms in the language of linear algebra, 2011" https://doi.org/10.1137/1.9780898719918

Matrix multiplication

A... Adjacency matrix, or Arcs

only diagonals and 7→6 are shown





matrix multiplication

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

How long is the "shortest path" (minimal sum of weights) from 7 to 6?

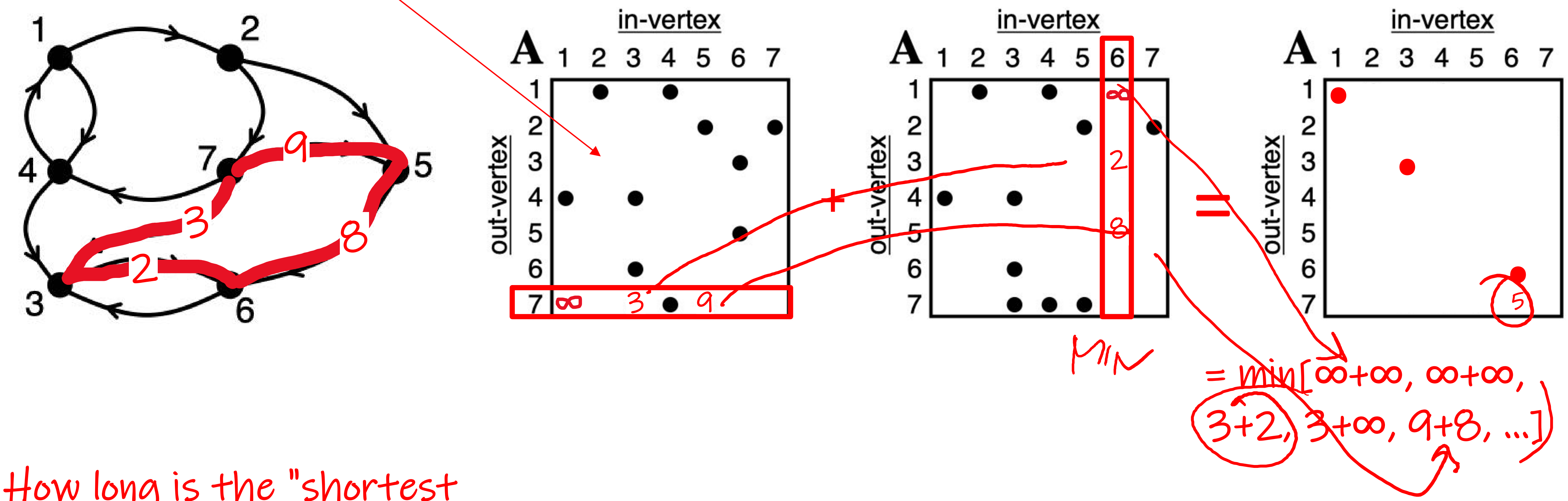
?

Matrix multiplication

Neutral element ∞ instead of 0

A... Adjacency matrix, or Arcs

only diagonals and 7→6 are shown



How long is the "shortest path" (minimal sum of weights) from 7 to 6?

The Relational Algebra

Company

cid	CName	StockPrice	Country
1	GizmoWorks	25	USA
2	Canon	65	Japan
3	Hitachi	15	Japan

- In the relational algebra (RA) the elements are relations
  - A relation is a schema together with a finite set of tuples
- RA has 5 primitive operators:
  - Unary: projection, selection
  - Binary: union, difference, Cartesian product
- Each of the 5 is essential or "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 (thus also called derived operators)
  - For example, equi-joins via Cartesian product and selection

RA vs other Query Languages (QLs)

- There are some subtle (yet important) differences between RA and other QLs. In RA, ...
  - ... can tables have duplicate records?

    ?
  - ... are missing (NULL) values allowed?

    ?
  - ... is there any order among records?

    ?
  - ...is the answer dependent on the domain from which values are taken (not just the database at hand)?

    ?

RA vs other Query Languages (QLs)


- There are some subtle (yet important) differences between RA and other QLs. In RA, ...
  - ... 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 database at hand)?
    - (RA vs. unsafe RC)

Recall: Virtues of the relational model

- "Separation of concerns": Physical independence (logical too), Declarative
- Simple, elegant clean: Everything is a relation
- Why did it take multiple years?
  - Doubted it could be done efficiently.

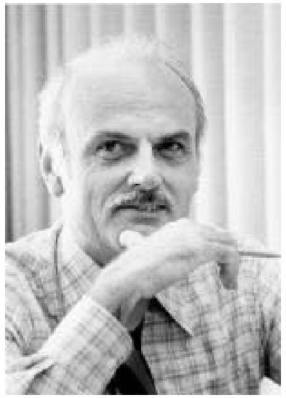
again in System R and in Eagle, the big project at Santa Teresa. Nevertheless, what kicked off this work was a key paper by Ted Codd – was it published in 1970 in CACM?

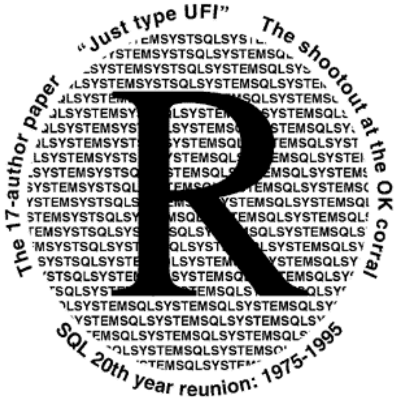
Mike Blasgen: Yes.

Irv Traiger: A couple of us from the Systems Department had tried to read it – couldn't make heads nor tails out of it. *[laughter]* At least back then, it seemed like a very badly written paper: some industrial motivation, and then right into the math. *[laughter]*

Bob Yost: I went over there with several other people – I was in the Advanced Systems Development Division – I remember going over there in about 1970 to see this because we were working with the IMS[8] guys at the time. We couldn't believe it; we thought it's going to take at least ten years before there's going to be anything. And it was ten years. *[laughter]*

Irv Traiger: So we had this 1970 paper; there were a couple of other papers that Ted had written after that; one on a language called DSL/Alpha[9], which was based on the predicate calculus. Glenn Bacon, who had the Systems Department, used to wonder how Ted could justify that everybody would be able to write this language that was based on mathematical predicate calculus, with universal quantifiers and existential quantifiers and variables and really, really hairy stuff.





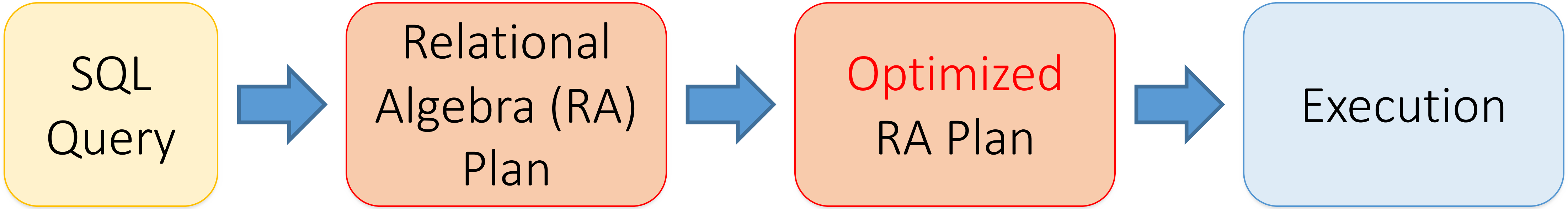
System R is a database system built as a research project at IBM San Jose Research (now IBM Almaden Research Center) in the 1970's. System R introduced the SQL language and also demonstrated that a relational system could provide good transaction processing performance.

See also: "The 1995 SQL Reunion: People, Projects, and Politic", edited by Paul McJones 1997. https://www.mcjones.org/System_R/SQL_Reunion_95/index.html

RDBMS Architecture

- How does a SQL engine work ?

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



Declarative query (from user)

Translate to relational algebra expression

Find logically equivalent- but more efficient- RA expression

Execute each operator of the optimized plan!

Algebra and the connection to logic and queries

- Algebra
- Relational Algebra
  - Operators
  - Independence
  - Power of algebra: optimizations
- Equivalence RA and safe RC (Codd's theorem)

Relational Algebra (RA) operators

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

R(A, B) R.A R(A, _)

- Five basic operators:
  1. Selection: σ ("sigma")
  2. Projection: Π
  3. Cartesian Product: ×
  4. Union: ∪
  5. Difference: –
- Auxiliary operators (sometimes counted as basic):
  6. Renaming: ρ ("rho")
- Derived
  7. Joins ⋈ (natural, equi-join, theta join, semi-join)
  8. Intersection / complement
  9. Division

Two perspectives: we focus on the named perspective, where every attribute must have a unique name, thus attribute order does not matter (contrast with vectors)

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

RDBMSs use multisets (bags), however in RA we will consider sets

Relational difference – can also be written as \ like set difference. "–" is used e.g. by [Silberschatz+'20], [Ramakrishnan+'03], [Garcia-Molina+2014], and [Elmasri+'15]

Relational Algebra (RA) operators

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

1. Selection (σ)

- Returns all tuples which satisfy a condition
- Notation: σ_c (R)
- Examples
  - Employee(<u>ssn</u>, name, salary)
  - $\sigma_{Salary > 40000}$ (Employee)
  - $\sigma_{name = \text{"Smith"}}$ (Employee)
- The condition c can be comparison predicates =, <, ≤, >, ≥, <> combined with AND, OR, NOT

Student(<u>sid</u>, sname, gpa)

SQL:

```
SELECT   *
FROM     Student
WHERE    gpa > 3.5
```

RA:

?

1. Selection (σ)

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

Student(sid, sname, gpa)

SQL:

```
SELECT   *
FROM     Student
WHERE    gpa > 3.5
```

RA:

$$\sigma_{\text{gpa} > 3.5}(\text{Student})$$

1. Selection example

Employee

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

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

?

1. Selection example

Employee

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

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

SSN	Name	Salary
5423341	Smith	60000
4352342	Fred	50000

2. Projection (Π)

- Eliminates columns, then removes duplicates (set perspective!)
- Notation: $\Pi_{A1,\ldots,An}(R)$
- Alternative: $\Pi_{-B1,\ldots,Bn}(R)$

"project away" operator (not standard)

- Example: project on social-security number and names:
  - Employee(ssn, name, salary)
  - $\Pi_{SSN,\ Name}$ (Employee)
  - Output schema: Answer(SSN, Name)

Student(sid,sname,gpa)

SQL:

```
SELECT DISTINCT sname, gpa
FROM Student
```

RA:

?

2. Projection (Π)

- Eliminates columns, then removes duplicates (set perspective!)
- Notation: $\Pi_{A1,\dots,An}(R)$
- Alternative: $\Pi_{-B1,\dots,Bn}(R)$

  "project away" operator (not standard)

- Example: project on social-security number and names:
  - Employee(ssn, name, salary)
  - $\Pi_{SSN,\ Name}$ (Employee)
  - Output schema: Answer(SSN, Name)

Student(sid,sname,gpa)

SQL:

```
SELECT DISTINCT sname, gpa
FROM Student
```

RA:

$$\Pi_{\text{sname,gpa}}(\text{Student})$$

2. Projection example

Employee

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

Π_{name} (Employee)

?

2. Projection example

Employee

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

Π_{name} (Employee)

Name
Ciara

2. Projection example

Employee

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

$\Pi_{\text{name, salary}}$ (Employee)

?

2. Projection example

Employee

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

$\Pi_{\text{name, salary}}$ (Employee)

Bag semantics

?

Name	Salary
Ciara	20000
Ciara	60000

2. Projection example

Employee

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

$$\Pi_{\text{name, salary}}(\text{Employee})$$

Which semantics is more efficient?

?

Bag semantics

Name	Salary
Ciara	20000
Ciara	60000
Ciara	20000

Name	Salary
Ciara	20000
Ciara	60000

Composing RA Operators

Patient

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

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

zip	disease
98125	flu
98125	heart
98120	lung
98120	heart

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

zip	disease
98125	heart
98120	heart

Composing RA Operators

How do we call what we see on this page / the property of these two operators ?

Patient

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

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

zip	disease
98125	flu
98125	heart
98120	lung
98120	heart

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

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

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

zip	disease
98125	heart
98120	heart

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

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

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

zip	disease
98125	flu
98125	heart
98120	lung
98120	heart

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

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

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

zip	disease
98125	heart
98120	heart

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

Logical Equivalece of RA Plans

R(A,B)

$$\Pi_A(\sigma_{A=5}(R)) \overset{?}{\Leftrightarrow} \sigma_{A=5}(\Pi_A(R))$$

Do projection & selection <u>commute</u> in this example?

?

Logical Equivalece of RA Plans

R(A,B)

$$\Pi_A(\sigma_{A=5}(R)) \overset{?}{\Leftrightarrow} \sigma_{A=5}(\Pi_A(R))$$

Do projection & selection commute in this example?

Yes ☺

$$\Pi_B(\sigma_{A=5}(R)) \overset{?}{\Leftrightarrow} \sigma_{A=5}(\Pi_B(R))$$

What about here?

?

Logical Equivalece of RA Plans

R(A,B)

$$\Pi_A(\sigma_{A=5}(R)) \overset{?}{\Leftrightarrow} \sigma_{A=5}(\Pi_A(R))$$

Do projection & selection commute in this example?

Yes ☺

R'(B)

$$\Pi_B(\sigma_{A=5}(R)) \overset{?}{\Leftrightarrow} \sigma_{A=5}(\Pi_B(R))$$

What about here?

No ☺

Commuting functions: a digression

Side-topic

- Do functions commute with taking the expectation?
  - $\mathbb{E}[f(x)] = f(\mathbb{E}[x])$?

?

Commuting functions: a digression

Side-topic

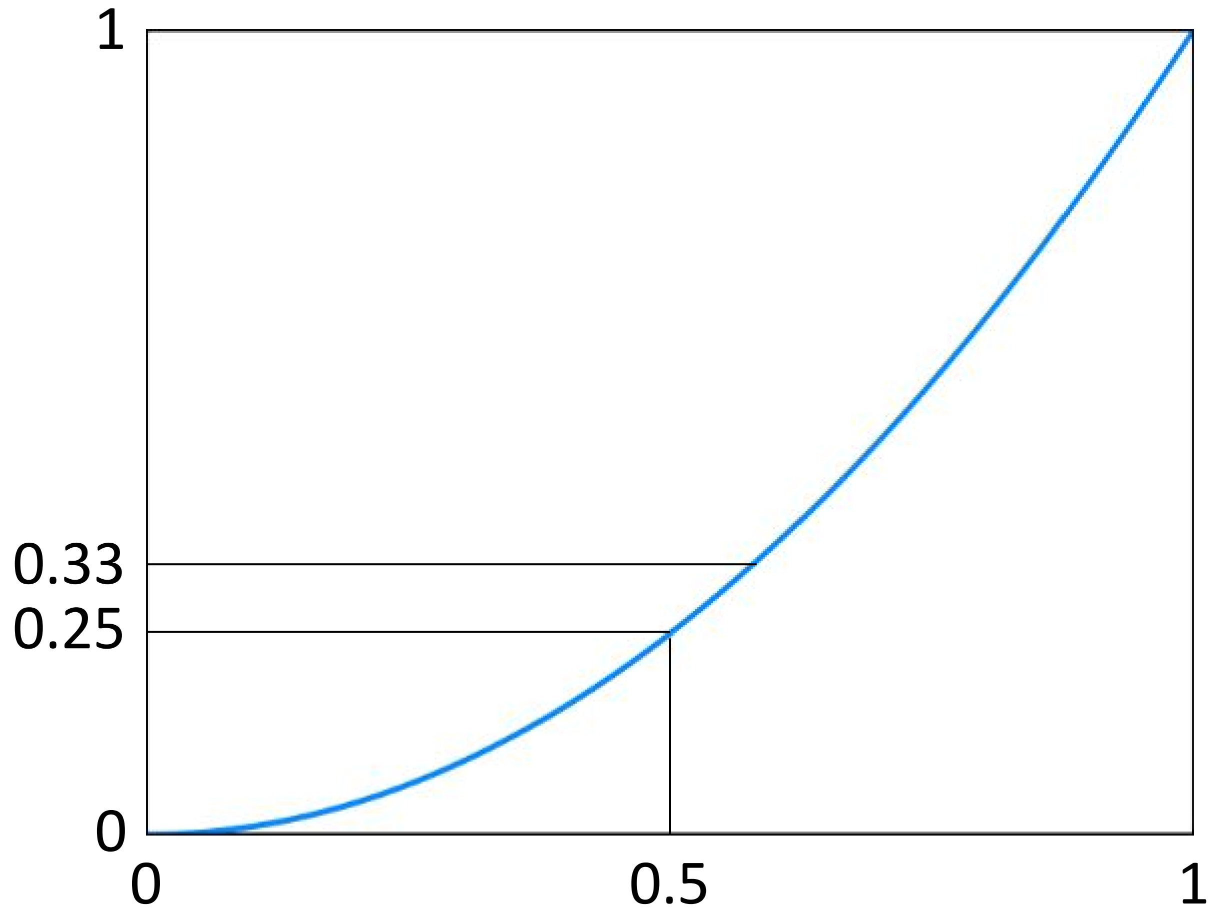
- Do functions commute with taking the expectation?
  - $\mathbb{E}[f(x)] = f(\mathbb{E}[x])$?
- Only for linear functions
  - Thus $f(x)=ax + b$
  - $\mathbb{E}[ax+b] = a\,\mathbb{E}[x] + b$
- Jensen's inequality for convex f

?

Commuting functions: a digression

Side-topic

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



Ratio of averages != average of ratios

Side-topic

- Assume you developed a new variant "1" and want to experimentally compare it against a baseline variant "2".
- Your Professor suggests to compare the methods on two data points (North and South) and report the AVG of their relative ratios. How does this sound?

?

Ratio of averages != average of ratios

Side-topic

- Assume you developed a new variant "1" and want to experimentally compare it against a baseline variant "2".
- Your Professor suggests to compare the methods on two data points (North and South) and report the AVG of their relative ratios. How does this sound?

DATA (higher ↑ is better):

	Variant 1	Variant 2	Ratio $\frac{\text{Variant 1}}{\text{Variant 2}}$
North	20	10	20/10 = ?
South	10	20	10/20 = ?
			AVG = ?

Ratio of averages != average of ratios

Side-topic

- Assume you developed a new variant "1" and want to experimentally compare it against a baseline variant "2".
- Your Professor suggests to compare the methods on two data points (North and South) and report the AVG of their relative ratios. How does this sound?

DATA (higher ↑ is better):

	Variant 1	Variant 2	Ratio $\frac{\text{Variant 1}}{\text{Variant 2}}$
North	20	10	20/10 = 2
South	10	20	10/20 = 0.5

AVG = ?

Ratio of averages != average of ratios

Side-topic

- Assume you developed a new variant "1" and want to experimentally compare it against a baseline variant "2".
- Your Professor suggests to compare the methods on two data points (North and South) and report the AVG of their relative ratios. How does this sound?

DATA (higher ↑ is better):

	Variant 1	Variant 2	Ratio $\frac{\text{Variant 1}}{\text{Variant 2}}$
North	20	10	20/10 = 2
South	10	20	10/20 = 0.5

AVG = 1.25 + 25%

Ratio of averages != average of ratios

Side-topic

- Assume you developed a new variant "1" and want to experimentally compare it against a baseline variant "2".
- Your Professor suggests to compare the methods on two data points (North and South) and report the AVG of their relative ratios. How does this sound?

DATA (higher ↑ is better):

	Variant 1	Variant 2	Ratio $\frac{\text{Variant 1}}{\text{Variant 2}}$
North	20	10	20/10 = 2
South	10	20	10/20 = 0.5
			AVG = 1.25

+ 25%

CONCLUSION

Variant 1 is on average 25% better

? ☹

RA Operators are Compositional, in general

Student(sid,sname,gpa)

How do we represent this query in RA?

```
SELECT DISTINCT sname, gpa
FROM Student
WHERE gpa > 3.5
```

⇒ ?

RA Operators are Compositional, in general

Student(sid,sname,gpa)

```
SELECT DISTINCT sname, gpa
FROM Student
WHERE gpa > 3.5
```

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

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

Which of those two variants is correct?

?

RA Operators are Compositional, in general

Student(<u>sid</u>,sname,gpa)

```
SELECT DISTINCT sname, gpa
FROM Student
WHERE gpa > 3.5
```

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

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

Both are correct: logically equivalent ☺

Relational Algebra (RA) operators

- Five basic operators:
  1. Selection: σ ("sigma")
  2. Projection: Π
  3. Cartesian Product: ×
  4. Union: ∪
  5. Difference: −
- Auxiliary (or special) operator
  6. Renaming: ρ ("rho")
- Derived (or implied) operators
  7. Joins ⋈ (natural, theta join, equi-join, semi-join)
  8. Intersection / complement
  9. Division

3. Cross-Product (×)

- Each tuple in R with each tuple in S
- Notation: R × S
- Example:
  - Students × Advisors
- Rare in practice; mainly used to express joins

Student(sid,sname,gpa)
People(ssn,pname,address)

SQL:

```
SELECT *
FROM People, Student
```

RA:

?

3. Cross-Product (×)

- Each tuple in R with each tuple in S
- Notation: R × S
- Example:
  - Students × Advisors
- Rare in practice; mainly used to express joins

Student(sid,sname,gpa)
People(ssn,pname,address)

SQL:

```
SELECT *
FROM People, Student
```

RA:

$$\text{People} \times \text{Student}$$

3. Cross join example

People

ssn	pname	address
1234545	John	216 Rosse
5423341	Bob	217 Rosse

×

Student

sid	sname	gpa
001	John	3.4
002	Bob	1.3

$$People \times Student$$

?

3. Cross join example

People

ssn	pname	address
1234545	John	216 Rosse
5423341	Bob	217 Rosse

×

Student

sid	sname	gpa
001	John	3.4
002	Bob	1.3

People × Student

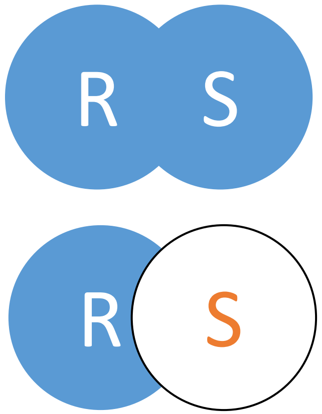
ssn	pname	address	sid	sname	gpa
1234545	John	216 Rosse	001	John	3.4
~~5423341~~	~~Bob~~	~~217 Rosse~~	~~001~~	~~John~~	~~3.4~~
~~1234545~~	~~John~~	~~216 Rosse~~	~~002~~	~~Bob~~	~~1.3~~
5423341	Bob	216 Rosse	002	Bob	1.3

Relational Algebra (RA) operators

- Five basic operators:
  1. Selection: σ ("sigma")
  2. Projection: Π
  3. Cartesian Product: ×
  4. Union: ∪
  5. Difference: –
- Auxiliary (or special) operator
  6. Renaming: ρ ("rho")
- Derived (or implied) operators
  7. Joins ⋈ (natural, theta join, equi-join, semi-join)
  8. Intersection / complement
  9. Division

4. Union (∪) and 5. Difference (–)

R ∪ S
R – S



- Examples:
  - Students ∪ Faculty
  - AllNEUEmployees – RetiredFaculty

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

What about the union of
Student and Faculty?

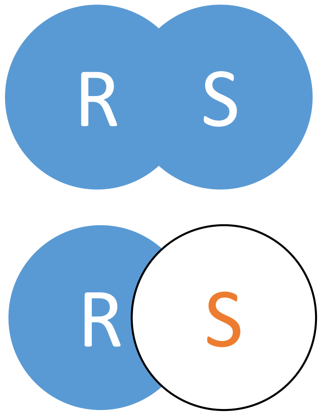
?

Difference also sometimes written with "\", thus as R\S

4. Union (∪) and 5. Difference (–)

Actor (aid, fname, lname)
Play (aid, mid, role)

R ∪ S
R – S



- Examples:
  - Students ∪ Faculty
  - AllNEUEmployees – RetiredFaculty

Other example: find actor ids who don't play in any movie:

?

Student (neuid, fname, lname)
$\pi_{-college}$(Faculty (neuid, fname, lname, college))

What about the union of Student and Faculty?

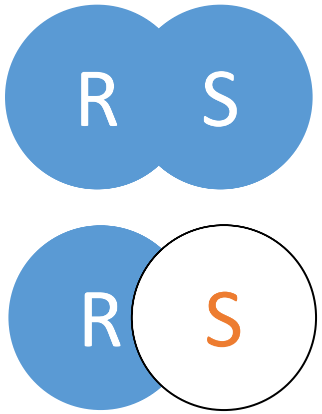
No! Only makes sense if R and S are "compatible", thus have the same schema!

Difference also sometimes written with "\", thus as R\S

4. Union (∪) and 5. Difference (–)

Actor (aid, fname, lname)
Play (aid, mid, role)

R ∪ S
R – S



Other example: find actor ids who don't play in any movie:

$$\pi_{aid}(Actor) - \pi_{aid}(Play)$$

- Examples:
  - Students ∪ Faculty
  - AllNEUEmployees – RetiredFaculty

Student (neuid, fname, lname)
$\pi_{\text{-college}}$(Faculty (neuid, fname, lname, college))

What about the union of Student and Faculty?

No! Only makes sense if R and S are "compatible", thus have the same schema!

Difference also sometimes written with "\", thus as R\S

Relational Algebra (RA) operators

- Five basic operators:
  1. Selection: σ ("sigma")
  2. Projection: Π
  3. Cartesian Product: ×
  4. Union: ∪
  5. Difference: −
- Auxiliary (or special) operator
  6. Renaming: ρ ("rho")
- Derived (or implied) operators
  7. Joins ⋈ (natural, theta join, equi-join, semi-join)
  8. Intersection / complement
  9. Division

6. Renaming (ρ rho)

- Does not change the instance, only the schema (table or attribute names)
- Only needed in named perspective, thus a 'special' operator (neither basic nor derived)
- Several existing conventions:

$\rho_S(R)$ S new table name

$\rho_{S(B_1,\ldots,B_n)}(R)$ if positions can be used

$\rho_{S(A_1 \to B_1,\ldots,A_n \to B_n)}(R)$

$\rho_{A_1 \to B_1,\ldots,A_n \to B_n}(R)$ if attribute names, not order matters

$\rho_{B_1,\ldots,B_n}(R)$

Student(<u>sid</u>,sname,gpa)

SQL:

```
SELECT
    sid    AS studId,
    sname  AS name,
    gpa    AS gradePtAvg
FROM Student
```

RA:

?

Alternative to "→ " is the substitution symbol "/ ", e.g. B_1/A_1 same as $A_1 \to B_1$

6. Renaming (ρ rho)

- Does not change the instance, only the schema (table or attribute names)
- Only needed in named perspective, thus a 'special' operator (neither basic nor derived)
- Several existing conventions:

$\rho_S(R)$ S new table name

$\rho_{S(B_1,\ldots,B_n)}(R)$ if positions can be used

$\rho_{S(A_1 \to B_1,\ldots,A_n \to B_n)}(R)$

$\rho_{A_1 \to B_1,\ldots,A_n \to B_n}(R)$ if attribute names, not order matters

$\rho_{B_1,\ldots,B_n}(R)$

Student(<u>sid</u>,sname,gpa)

SQL:

```
SELECT
   sid    AS studId,
   sname  AS name,
   gpa    AS gradePtAvg
FROM Student
```

RA:

$\rho_{\text{studId,name,gradePtAvg}}(\text{Student})$

Alternative to "$\to$" is the substitution symbol "/", e.g. B_1/A_1 same as $A_1 \to B_1$

6. Why we need renaming

R

A	B
1	2
3	4

S

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

$R \times S$

?

6. Why we need renaming

R

A	B
1	2
3	4

S

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

$R \times S$

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

What if we use renaming ?

6. Why we need renaming

R

A	B→E
1	2
3	4

S

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

R × S

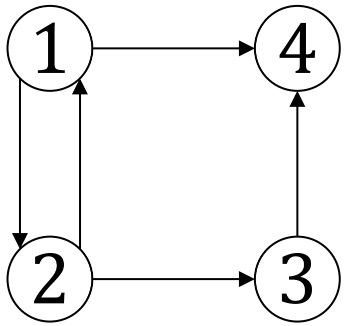
A	R.B	S.B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

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

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

We need renaming if we had R × R

6. Named vs Unnamed perspective



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

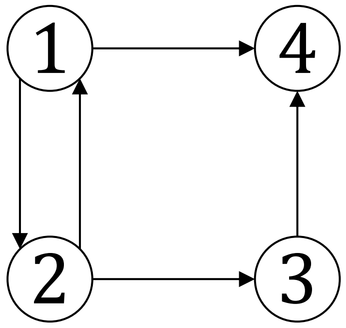
In DRC:

?

A:

S	T
1	2
2	1
2	3
1	4
3	4

6. Named vs Unnamed perspective



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

A:

S	T
1	2
2	1
2	3
1	4
3	4

In DRC:

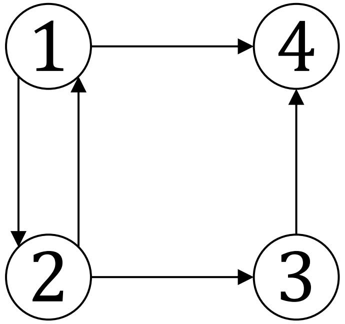
$$\{ x \mid \exists y,z.[A(x,y) \land A(y,z)]\}$$

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

In RA:

?

6. Named vs Unnamed perspective



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

In DRC:

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

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

A:

S	T
1	2
2	1
2	3
1	4
3	4

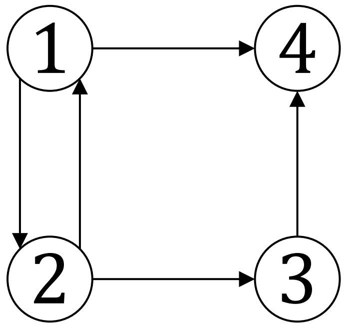
S2	T2
1	2
2	1
2	3
1	4
3	4

In RA:

$$\pi_S(\sigma_{T=S2}(A \times \rho_{S\rightarrow S2, S\rightarrow S2}(A)))$$ named perspective

? unnamed perspective

6. Named vs Unnamed perspective



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

In DRC:

$\{ x \mid \exists y,z.[A(x,y) \land A(y,z)]\}$ "unnamed"

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

A:

\$1 S	\$2 T
1	2
2	1
2	3
1	4
3	4

\$3 S2	\$4 T2
1	2
2	1
2	3
1	4
3	4

?

"named"

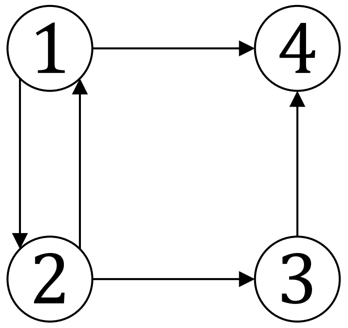
In RA:

$\pi_S(\sigma_{T=S2}(A \times \rho_{S\to S2, S\to S2}(A)))$ named perspective

$\pi_{\$1}(\sigma_{\$2=\$3}(A \times A))$ unnamed perspective

I adopt the notation \$2 from Ullman's old textbook. Often just written as "$\pi_1(\sigma_{2=3}(A \times A))$", which is ambiguous. A more recent database textbook uses "2 $\dot{=}$ 3" for "\$2=\$3" which gets confusing for "\$2=3"...

6. Named vs Unnamed perspective



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

In DRC:

$\{ x \mid \exists y,z.[A(x,y) \wedge A(y,z)]\}$ "unnamed"

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

In TRC:

$\{ q(S) \mid \exists a1, a2 \in A[a1.T=a2.S \wedge a1.S=q.S]\}$ "named"

In RA:

$\pi_S(\sigma_{T=S2}(A \times \rho_{S \to S2, S \to S2}(A)))$ named perspective

$\pi_{\$1}(\sigma_{\$2=\$3}(A \times A))$ unnamed perspective

A:

\$1 S	\$2 T	\$3 S2	\$4 T2
1	2	1	2
2	1	2	1
2	3	2	3
1	4	1	4
3	4	3	4

I adopt the notation \$2 from Ullman's old textbook. Often just written as "$\pi_1(\sigma_{2=3}(A \times A))$", which is ambiguous. A more recent database textbook uses "2 $\doteq$ 3" for "\$2=\$3" which gets confusing for "\$2=3"...