Updated 1/29/2024

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

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp24)

https://northeastern-datalab.github.io/cs7240/sp24/

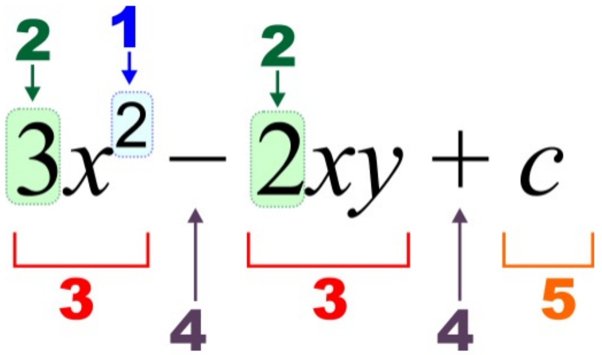
1/29/2024

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)
  - RA $\rightarrow$ RC
  - RC $\rightarrow$ RA

What is “Algebra”?

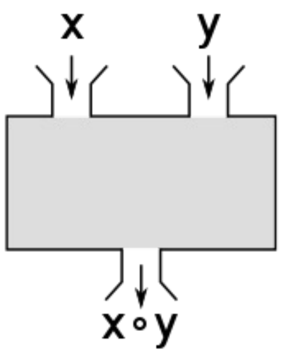
- Algebra is the study of mathematical symbols and the rules for manipulating these symbols
  - e.g., Linear Algebra
  - e.g., Relational Algebra
  - e.g., Boolean Algebra
  - e.g., Elementary algebra
  - e.g., Abstract algebra (groups, rings, fields, ...)



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 (acting on operands)
    - 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 $op(x,y) \equiv op(y,x)$

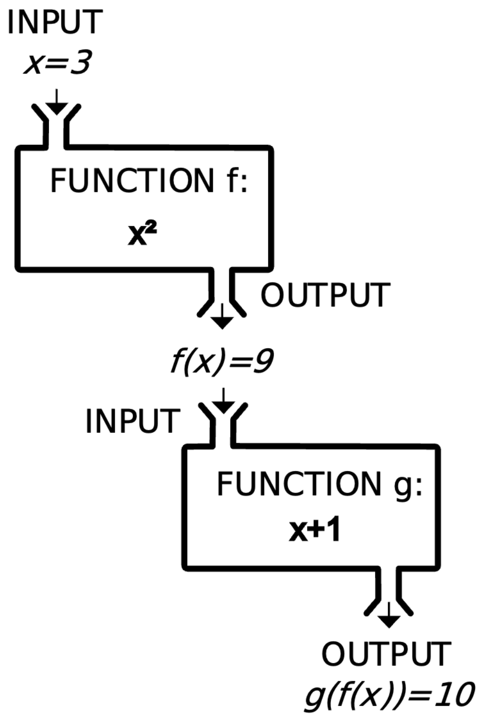
- Examples:
  - Boolean algebra: $(\{true,false\},\{\wedge,\vee,\neg\})$
  - Ring of integers: $(\mathbb{Z},\{+,\cdot\})$
  - Relational algebra

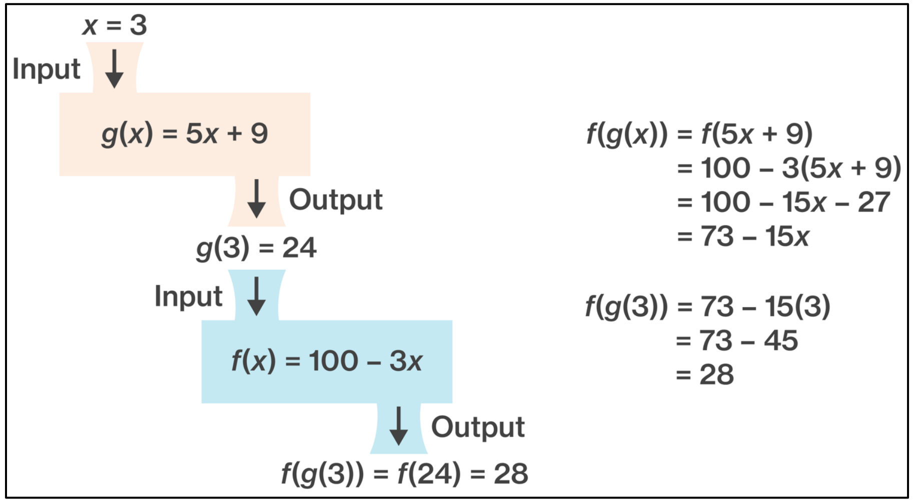
ring: set equipped with two binary operations with certain properties like distributivity of multiplication over addition

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

Based on material by Benny Kimelfeld and Oded Shmueli for 236363 Database Management Systems, Technion, 2018.
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Function composition





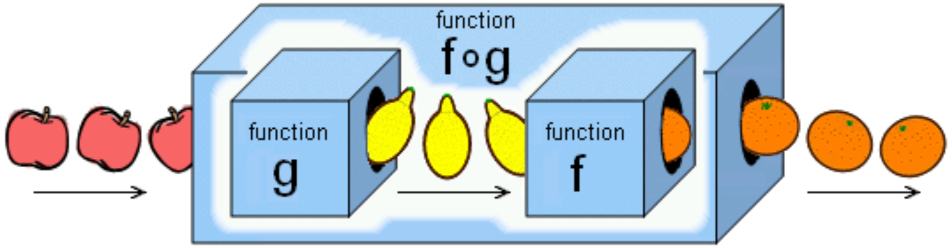
$$[f \circ g](x) = f[g(x)]$$

Let's find FoG(x) of two example equations:

$$f(x) = x + 2 \qquad g(x) = x^2 + 1$$

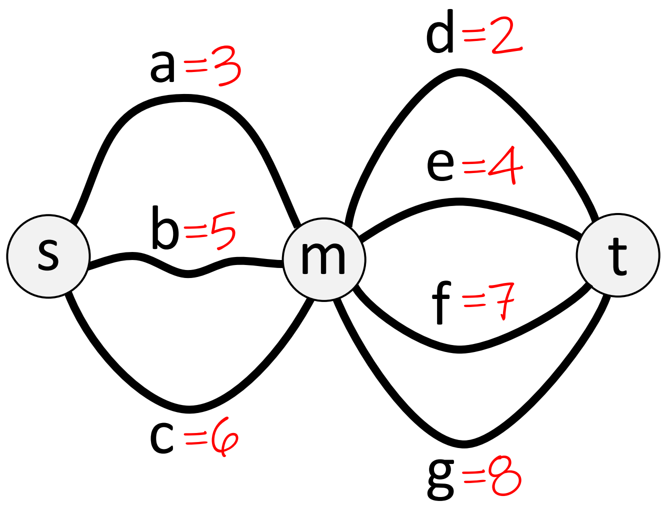
What is $[f \circ g](x)$?

$$[f \circ g](x) = f[g(x)]$$
$$[f \circ g](x) = f[x^2 + 1]$$
$$[f \circ g](x) = (x^2 + 1) + 2$$
$$[f \circ g](x) = x^2 + 3$$



Sources: https://www.coursehero.com/sg/college-algebra/composition-of-functions/, https://upload.wikimedia.org/wikipedia/commons/2/21/Function_machine5.svg , https://en.wikibooks.org/wiki/Algebra/Functions , http://www.statisticslectures.com/topics/compositionoffunctions/

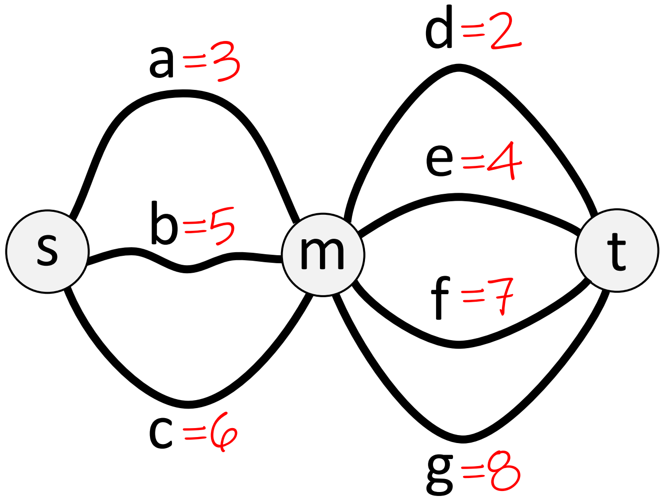
Distributivity = efficient factorization



What is the shortest path from s to t?

?

Distributivity = efficient factorization



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

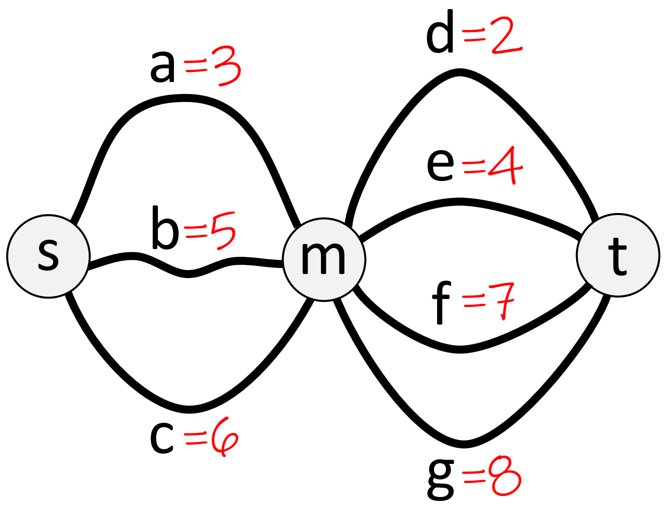
min[3+2, 3+4, 3+7, 3+8, ..., 6+8]

?

What is the shortest path from s to t?

Answer: 5 = 3 + 2

Distributivity = efficient factorization



What is the shortest path from s to t?

Answer: 5 = 3 + 2

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]

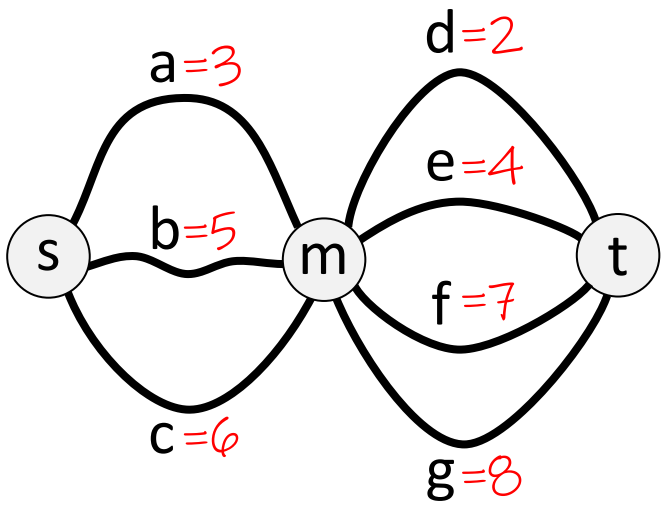
min[x,y]+z = min[(x+z), (y+z)]
(+ distributes over min)

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]

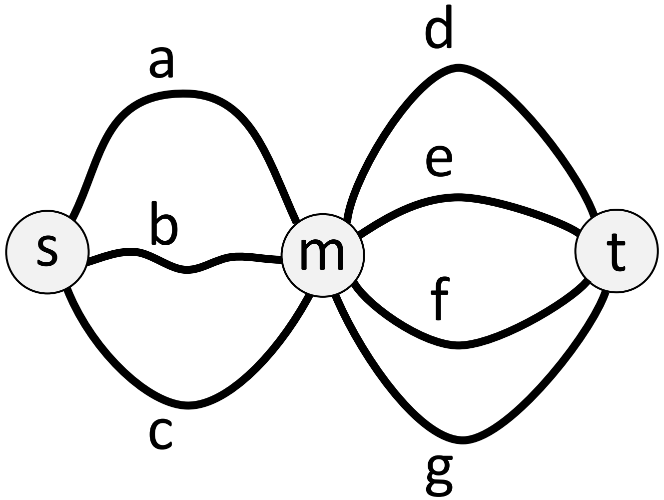
What is the shortest path from s to t?

Answer: 5 = 3 + 2

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

(+ distributes over min)

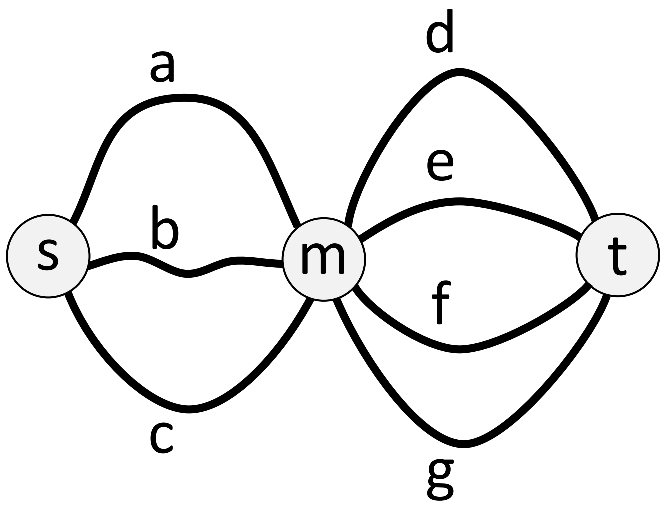
Distributivity = efficient factorization



How many paths are there from s to t?

?

Distributivity = efficient factorization



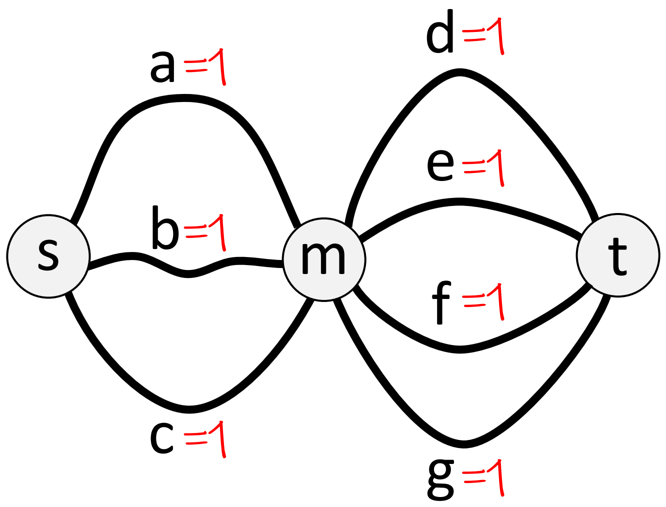
How many paths are there from s to t?

Answer: $12 = 3 \cdot 4$

Distributivity = efficient factorization

(Ring of real numbers)

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



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

$\text{count}\underbrace{[1\cdot1, 1\cdot1, 1\cdot1, 1\cdot1, ..., 1\cdot1]}_{12}$

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

count[1,1,1] · count[1,1,1,1]

How many paths are there from s to t?

Answer: 12 = 3 · 4

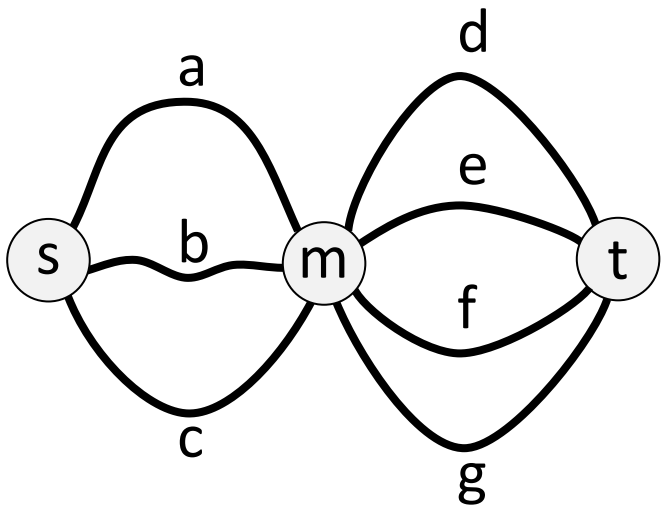
$+[x,y] \cdot z = +[x\cdot z, y\cdot z]$

(· distributes over +)

Distributivity = efficient factorization

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

Semirings generalize this idea



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

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

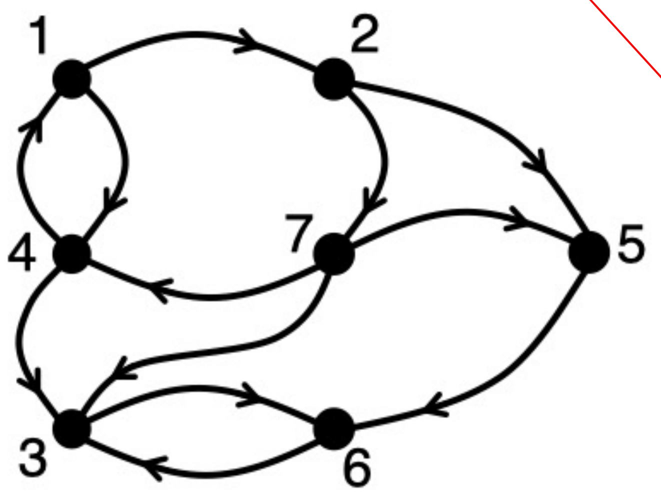
$\oplus[x,y] \otimes z = \oplus[x \otimes z, y \otimes z]$

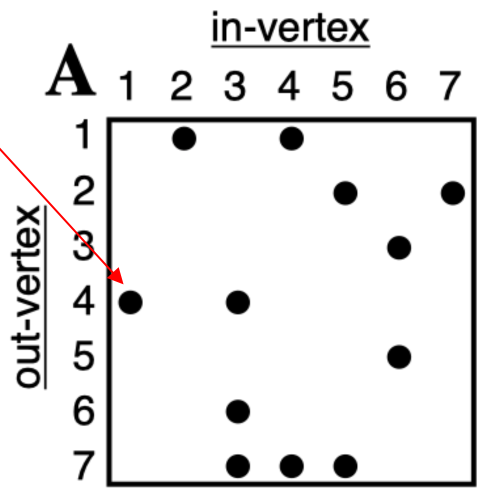
($\otimes$ distributes over $\oplus$)

Matrix multiplication

A... Adjacency matrix, or Arcs

think of dots as "1"s



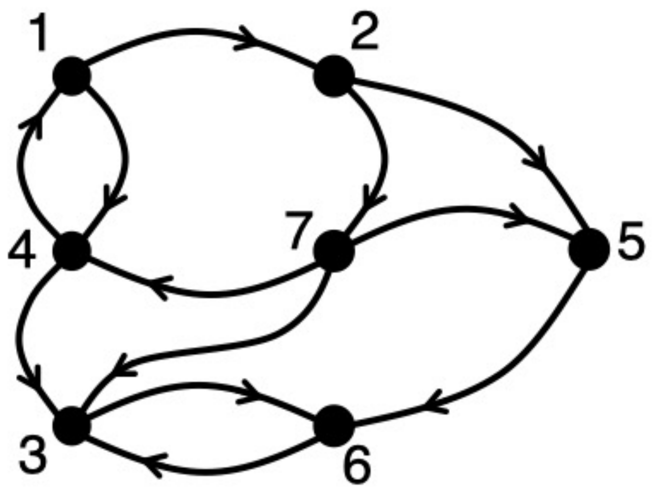


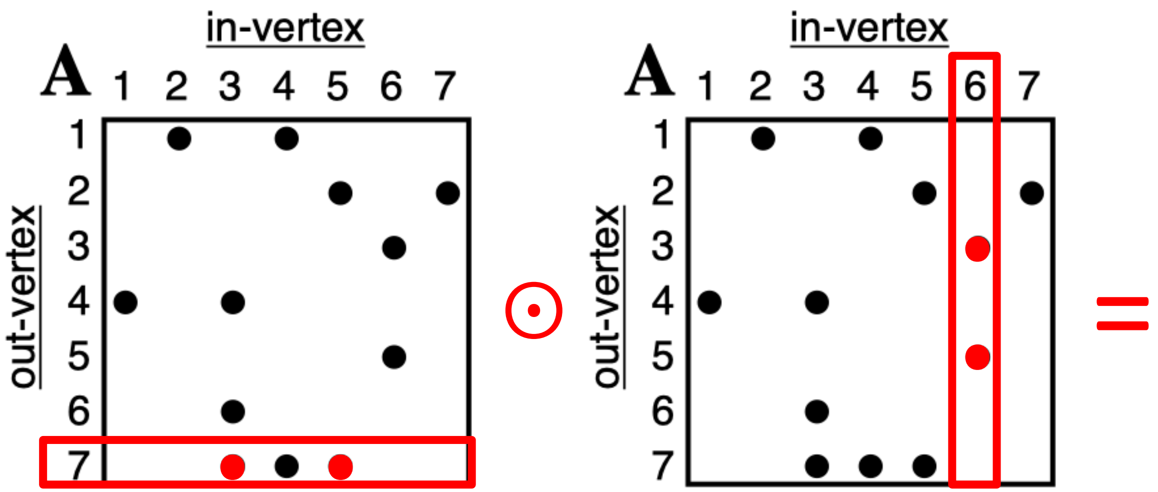
How many paths of
length 2 are there
from 7 to 6?

?

Matrix multiplication

A… Adjacency matrix, or Arcs





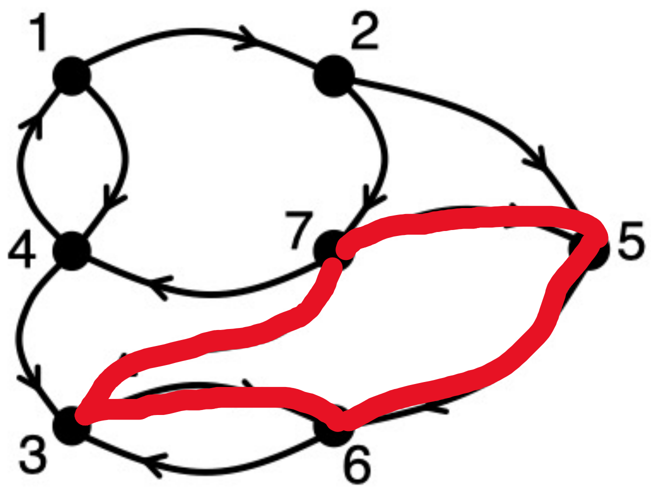
matrix
multiplication

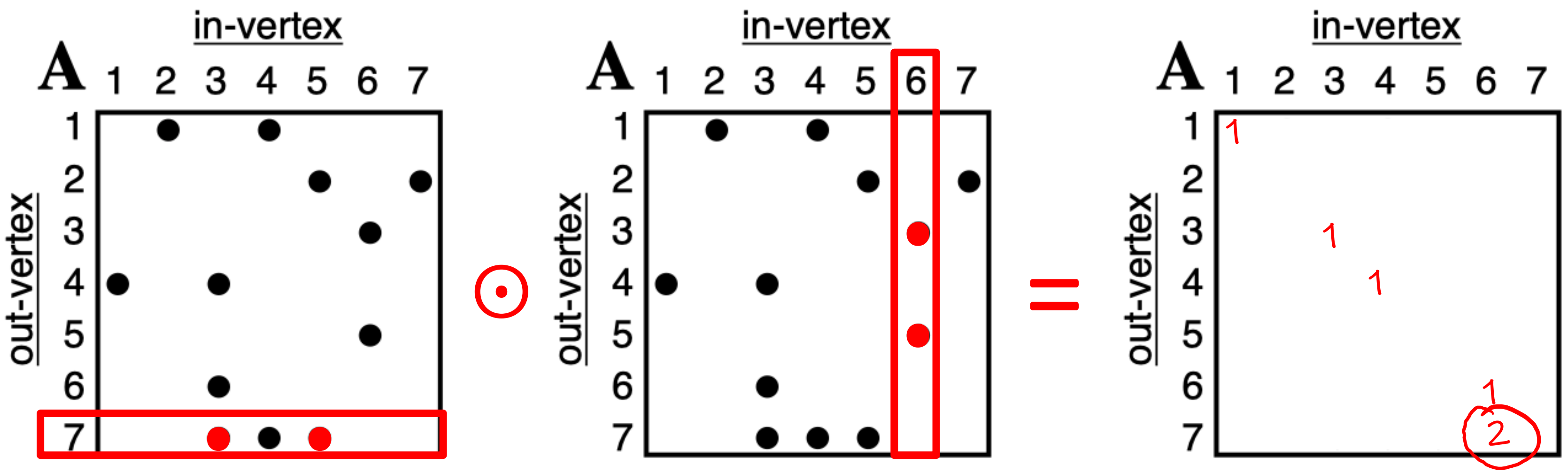
How many paths of
length 2 are there
from 7 to 6?

Matrix multiplication

A... Adjacency matrix, or Arcs

only diagonals and $7 \to 6$ are shown





matrix multiplication

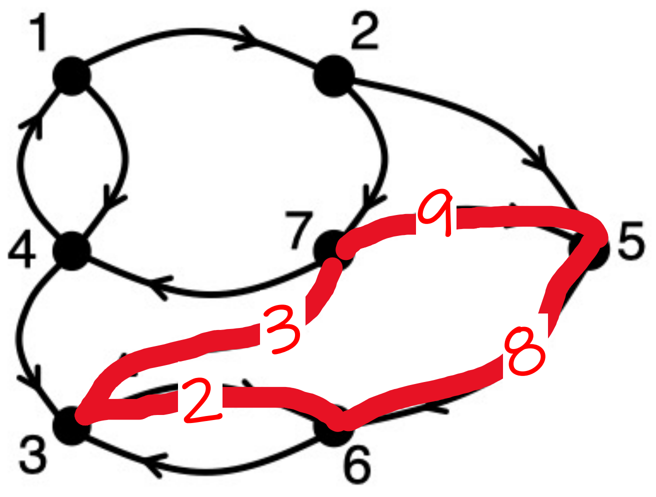
$= 0 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 + 1 \cdot 0 + 1 \cdot 1 + \ldots$

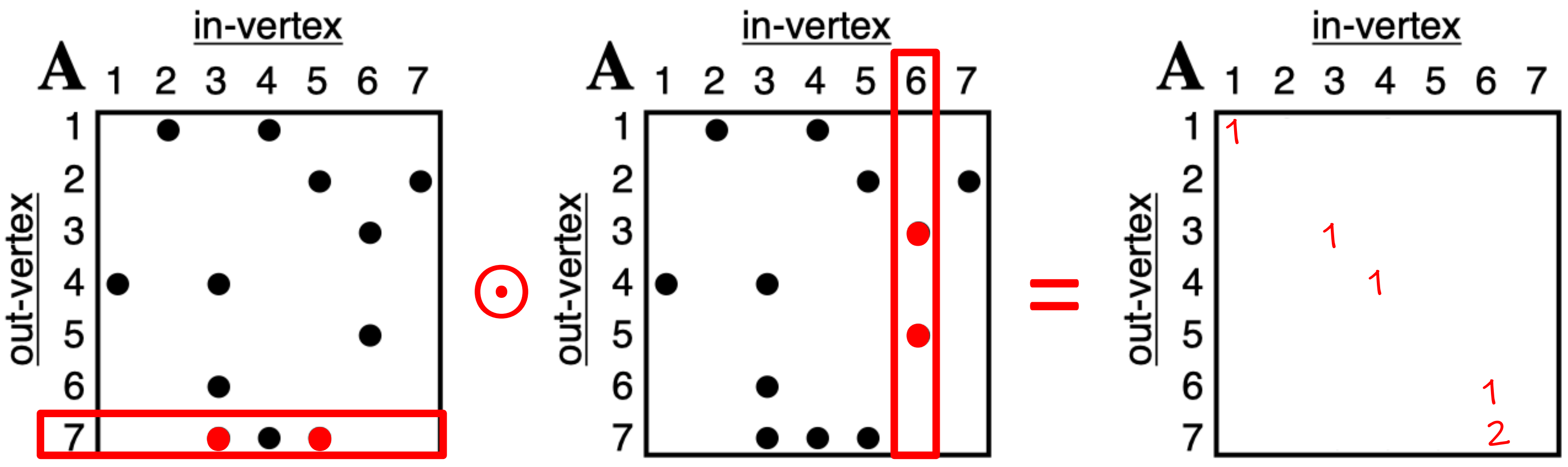
How many paths of length 2 are there from 7 to 6?

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? ?

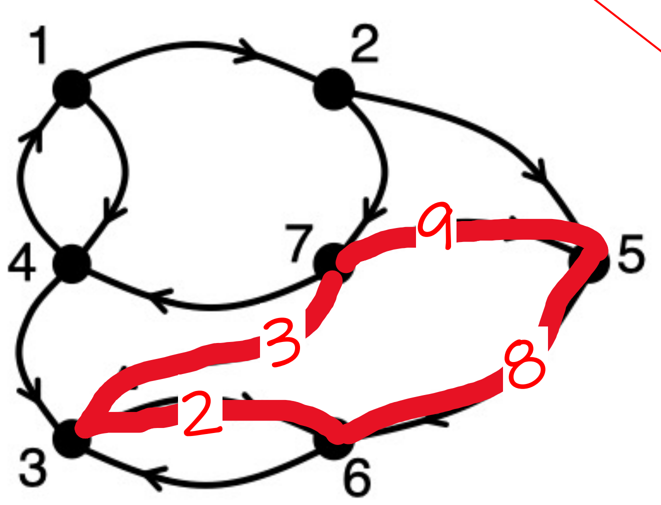
Example graph taken from "Kepner, Gilbert. Graph algorithms in the language of linear algebra, 2011" https://doi.org/10.1137/1.9780898719918

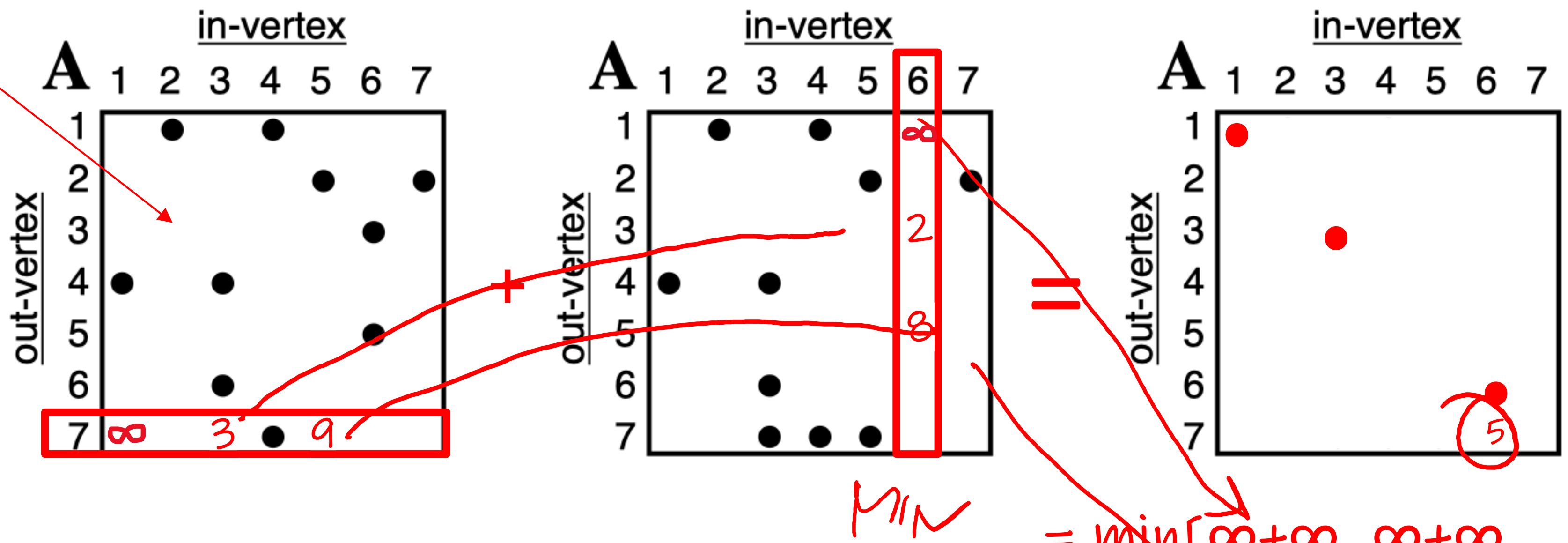
Matrix multiplication

Neutral element ∞ instead of 0

A... Adjacency matrix, or Arcs

only diagonals and 7→6 are shown





$= \min[\infty+\infty,\ \infty+\infty,\ 3+2,\ 3+\infty,\ 9+8,\ \ldots]$

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

The Relational Algebra

Company

<u>cid</u>	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)?

    ?

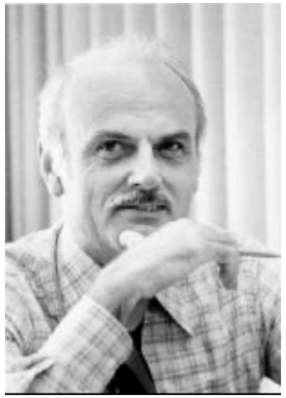
Based on material by Benny Kimelfeld and Oded Shmueli for 236363 Database Management Systems, Technion, 2018.
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

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/logic independence, declarative language
- Simple, elegant clean: Everything is a relation
- Why did it take multiple years to make it happen?
  - Big doubts it could be done efficiently.



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.

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.

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!

SQL Query	→	Relational Algebra (RA) Plan	→	Optimized RA Plan	→	Execution
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)
  - RA $\rightarrow$ RC
  - RC $\rightarrow$ RA

Relational Algebra (RA) operators

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

- 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:

- mainly named perspective, where every attribute must have a unique name, thus attribute order does not matter. E.g. "R.A=4" same for R(A,B) or R(B,A)
- contrast with vectors: E.g. R(x,y), x=4

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

Relational Algebra (RA) operators extending classical Set Theory

	Traditional **set** operators	Specific **relational** operators
Basic operators	$R \cup S$ (union) binary $R - S$ (difference) $R \times S$ (Cartesian prod.)	$\sigma_\theta(R)$ (selection) unary $\pi_{\boldsymbol{A}}(R)$ (projection)
Derived operators	Notice that the Cartesian product in set theory is <u>non-commutative</u> (cp. unnamed with named perspective) https://en.wikipedia.org/wiki/Cartesian_product $R \cap S$ (intersection)	$R \bowtie S$ (join) $R \ltimes S$ (semi-join) $R \rhd S$ (anti-join) $R \div S$ (division)
Extended operators		$\pi_{\boldsymbol{f(A)}}(R)$ (extended projection) $\delta(R)$ (duplicate elimination) $\gamma_{\boldsymbol{A},\mathbf{agg}(B)}(R)$ (grouping and aggregates) $\tau_A(R)$ (sorting) $R ⟗ S$ (outerjoin)

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") for named perspective
- Derived (or implied) operators
  7. Joins $\bowtie$ (natural, theta join, equi-join, [semi-join: moved to T3-U1])
  8. Intersection / complement
  9. Division: $\div$

1. Selection (σ)

Employee(<u>ssn</u>, name, salary)

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

SQL:

```
SELECT  *
FROM    Employee
WHERE   salary > 40000
```

RA:

?

1. Selection (σ)

Employee(<u>ssn</u>, name, salary)

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

SQL:

```
SELECT  *
FROM    Employee
WHERE   salary > 40000
```

RA:

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

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)

Employee(ssn, name, salary)

- 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)

SQL:

```
SELECT DISTINCT name, salary
FROM Employee
```

RA:

?

2. Projection (π)

Employee(ssn, name, salary)

- 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)

SQL:

```
SELECT DISTINCT name, salary
FROM Employee
```

RA:

$$\pi_{name,\ salary}(Employee)$$

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

$\pi_{\text{name, salary}}$ (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

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

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

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

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

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

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

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

zip	disease
98125	heart
98120	heart

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

RA Operators are compositional, in general

Patient

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

Both RA expressions are logically equivalent ☺

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

```
SELECT DISTINCT zip, disease
FROM Patient
WHERE disease = 'heart'
```

⇒

zip	disease
98125	heart
98120	heart

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

Logical Equivalece of RA Plans

R(A,B)

$$\pi_A\left(\sigma_{A=5}(R)\right) \overset{?}{\Leftrightarrow} \sigma_{A=5}\left(\pi_A(R)\right)$$

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

?

Logical Equivalece of RA Plans

R(A,B)

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

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

Yes ☺

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

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)

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

What about here? No ☹

Updated 2/2/2024

Topic 1: Data models and query languages
Unit 3: Relational Algebra (RA)
Lecture 7

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp24)

https://northeastern-datalab.github.io/cs7240/sp24/

2/2/2024

Pre-class conversations

- Last class summary
- Please keep on pointing out any errors on the slides
- It is time to start to hand in your first scribes (some ideas today)
- Project discussions (in 2 weeks: Fri 2/16: project ideas)

- today:
  - we continue with relational algebra (RA)
  - next week: equivalence of RA and *safe* RC (Codd's theorem)
- next time:
  - Recursion (Datalog)

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

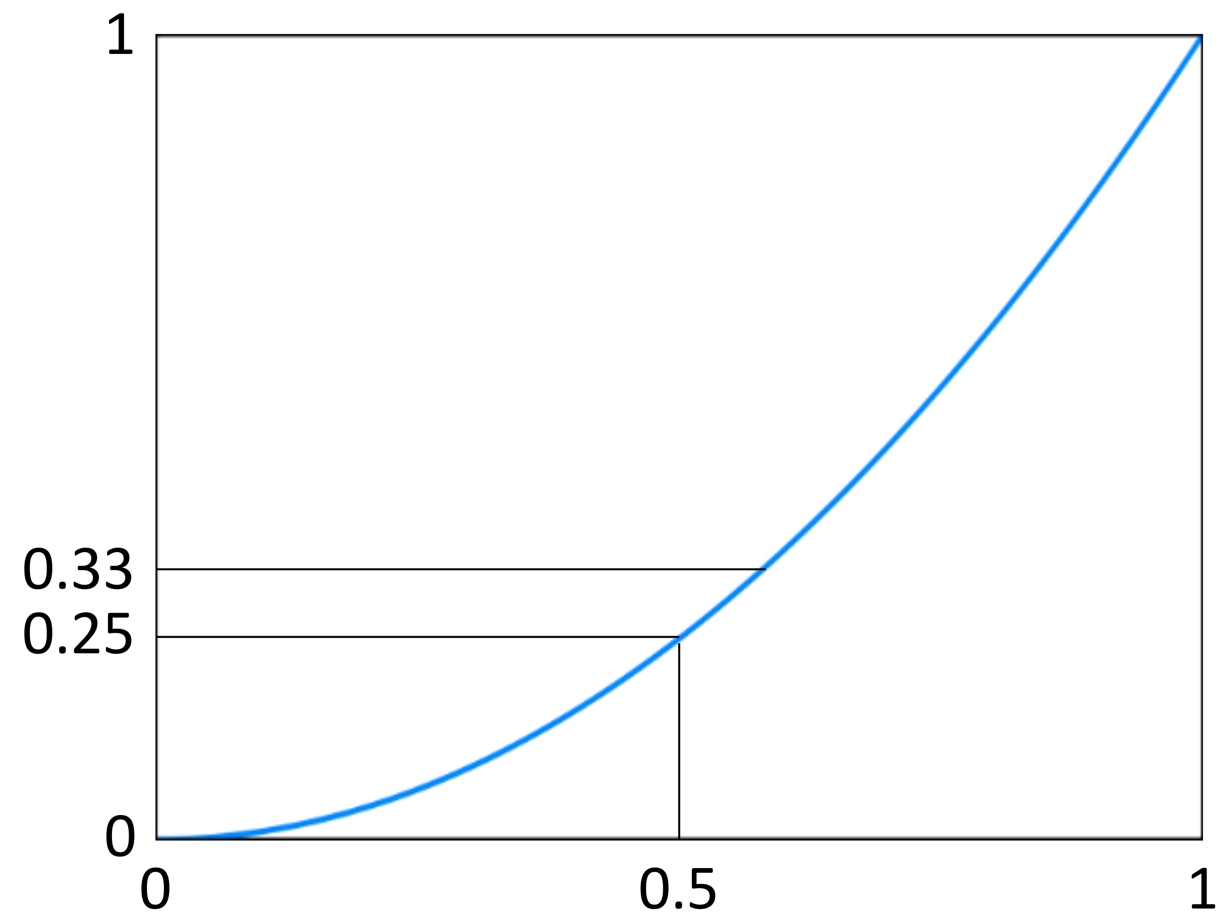
- Do functions commute with taking the expectation?
  - $\mathbb{E}[f(x)] = f(\mathbb{E}[x])$?
- Only for linear functions
  - Thus $f(x)=ax + b$
  - $\mathbb{E}[ax+b] = a\,\mathbb{E}[x] + b$
- Jensen's inequality for convex f

?

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" for creating "output" (the higher the better) and want to experimentally compare it against a baseline variant "2".
- Your Professor suggests to compare the methods on two data points (Alice and Bob) 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" for creating "output" (the higher the better) and want to experimentally compare it against a baseline variant "2".
- Your Professor suggests to compare the methods on two data points (Alice and Bob) and report the AVG of their relative ratios. How does this sound?

DATA (higher ↑ is better):

	Variant 1	Variant 2	Ratio $\frac{\text{Variant 1}}{\text{Variant 2}}$
Alice	20	10	20/10 = ?
Bob	10	20	10/20 = ?

AVG = ?

Ratio of averages != average of ratios

Side-topic

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

DATA (higher ↑ is better):

	Variant 1	Variant 2	Ratio $\frac{\text{Variant 1}}{\text{Variant 2}}$
Alice	20	10	20/10 = 2
Bob	10	20	10/20 = 0.5
			AVG = ?

Ratio of averages != average of ratios

Side-topic

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

DATA (higher ↑ is better):

	Variant 1	Variant 2	Ratio $\frac{\text{Variant 1}}{\text{Variant 2}}$
Alice	20	10	20/10 = 2
Bob	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" for creating "output" (the higher the better) and want to experimentally compare it against a baseline variant "2".
- Your Professor suggests to compare the methods on two data points (Alice and Bob) and report the AVG of their relative ratios. How does this sound?

DATA (higher ↑ is better):

	Variant 1	Variant 0	Ratio $\frac{\text{Variant 1}}{\text{Variant 0}}$
Alice	20	10	20/10 = 2
Bob	10	20	10/20 = 0.5
			AVG = 1.25

+ 25%

CONCLUSION

Variant 1 is on average 25% better

? ☹

See https://arxiv.org/pdf/2401.04758 Appendix O.1 for a more detailed discussion and suggestion to use the median. I only just learned that this exact problem is widely known in the computer benchmarking literature which suggests the geometric mean: "Fleming, Wallace. How not to lie with statistics: the correct way to summarize benchmark results. CACM 1986. https://dl.acm.org/doi/abs/10.1145/5666.5673

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") for named perspective
- Derived (or implied) operators
  7. Joins $\bowtie$ (natural, theta join, equi-join, [semi-join: moved to T3-U1])
  8. Intersection / complement
  9. Division: $\div$

3. Cartesian Product (×), or Cross-product

- Each tuple in R with each tuple in S
- Notation: $R \times S$
- $R \times S := \{(r, s) \mid r \in R, s \in S\}$
- 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. Cartesian Product (×), or Cross-product

- Each tuple in R with each tuple in S
- Notation: R×S
- R×S := {(r, s) | r ∈ R, s ∈ 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:

People × Student

3. Cross join example

People

ssn	pname	address
1234545	John	216 Rosse
5423341	Bob	217 Rosse

$\times$

Student

sid	sname	gpa
001	John	3.4
002	Bob	1.3

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

?

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 × 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

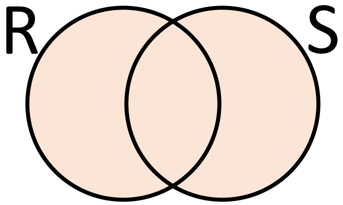
Relational Algebra (RA) operators

- Five basic operators:
  1. Selection: σ ("sigma")
  2. Projection: Π
  3. Cartesian Product: $\times$
  4. Union: $\cup$
  5. Difference: $-$
- Auxiliary (or special) operator
  6. Renaming: ρ ("rho") for named perspective
- Derived (or implied) operators
  7. Joins $\bowtie$ (natural, theta join, equi-join, [semi-join: moved to T3-U1])
  8. Intersection / complement
  9. Division: $\div$

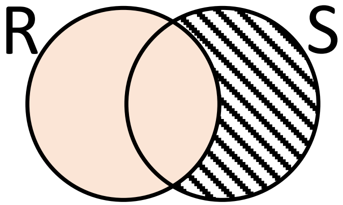
4. Union (∪) and 5. Difference (–)

R ∪ S

R – S



$R \cup S := \{x \mid x \in R \lor x \in S\}$



$R - S := \{x \mid x \in R \land x \notin S\}$

- 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?

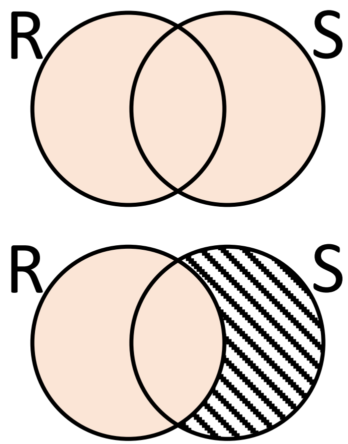
?

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

4. Union (∪) and 5. Difference (–)

Actor (<u>aid</u>, fname, lname)
Play (aid, mid, role)

$R \cup S$
$R - S$



$R \cup S := \{x \mid x \in R \lor x \in S\}$

$R - S := \{x \mid x \in R \land x \notin S\}$

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

?

- Examples:
  - Students ∪ Faculty
  - AllNEUEmployees – RetiredFaculty

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

$\pi_{\text{-college}}$(Faculty (<u>neuid</u>, fname, lname, college))

What about the union of Student and Faculty?

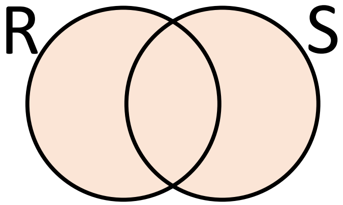
No! Only makes sense if R and S are "<u>union compatible</u>", thus have the same schema!

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

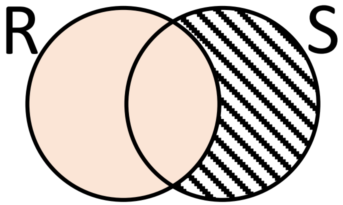
4. Union (∪) and 5. Difference (–)

Actor (<u>aid</u>, fname, lname)
Play (aid, mid, role)

$R \cup S$
$R - S$



$R \cup S := \{x \mid x \in R \lor x \in S\}$



$R - S := \{x \mid x \in R \land x \notin S\}$

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

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

- Examples:
  - Students ∪ Faculty
  - AllNEUEmployees – RetiredFaculty

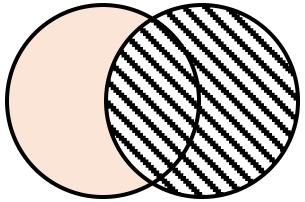
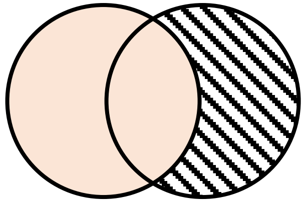
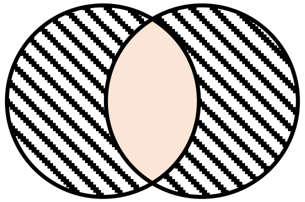
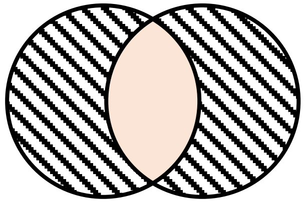
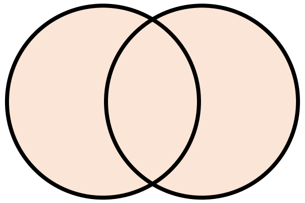
Student (<u>neuid</u>, fname, lname)
$\pi_{\text{-college}}$(Faculty (<u>neuid</u>, fname, lname, college))

What about the union of Student and Faculty?

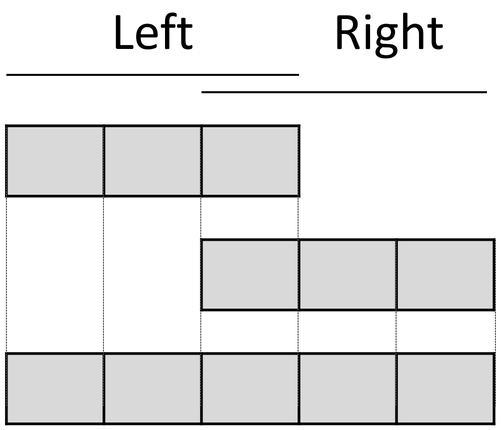
No! Only makes sense if R and S are "<u>union compatible</u>", thus have the same schema!

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

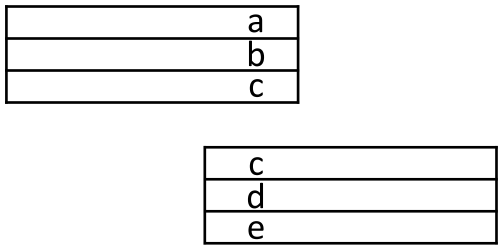
VENN diagrams for join types: rows vs column

Which rows are returned	*Which columns are returned:* Left	Union
Left	 (Left) Anti-Join	 Left (Outer) Join
Intersection	 (Left) Semi-Join	 Inner
Union		 Full (Outer) Join

columns (schema):



rows (data):



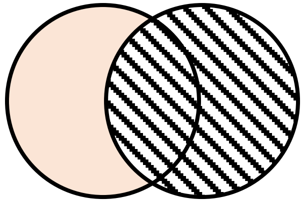
VENN diagrams for join types: rows vs column

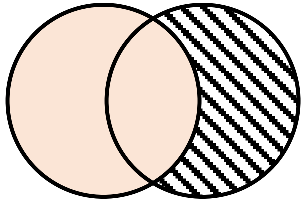
Join keys from which rows are returned:

Which columns are returned:

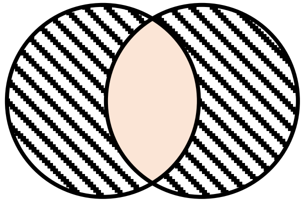
Left | Union | Intersection

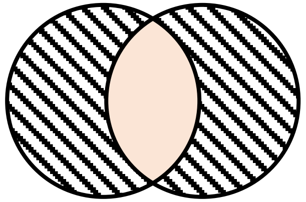
Left


(Left) Anti-Join


Left (Outer) Join

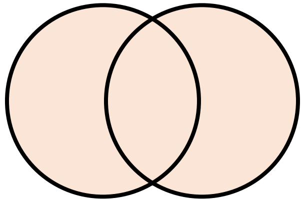
Intersection

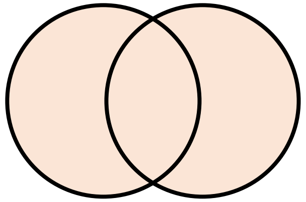

(Left) Semi-Join


Inner

Union

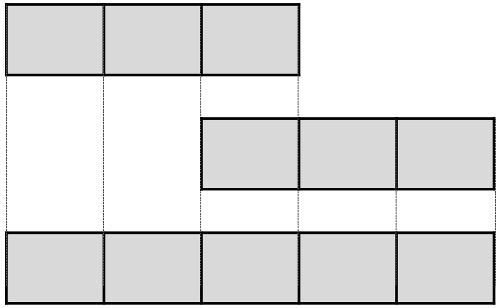
union = full join (for identical schemas)


Full (Outer) Join

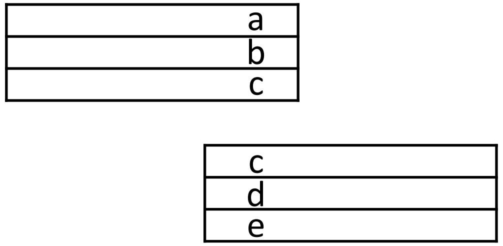

Union

Left | Right

columns (schema):

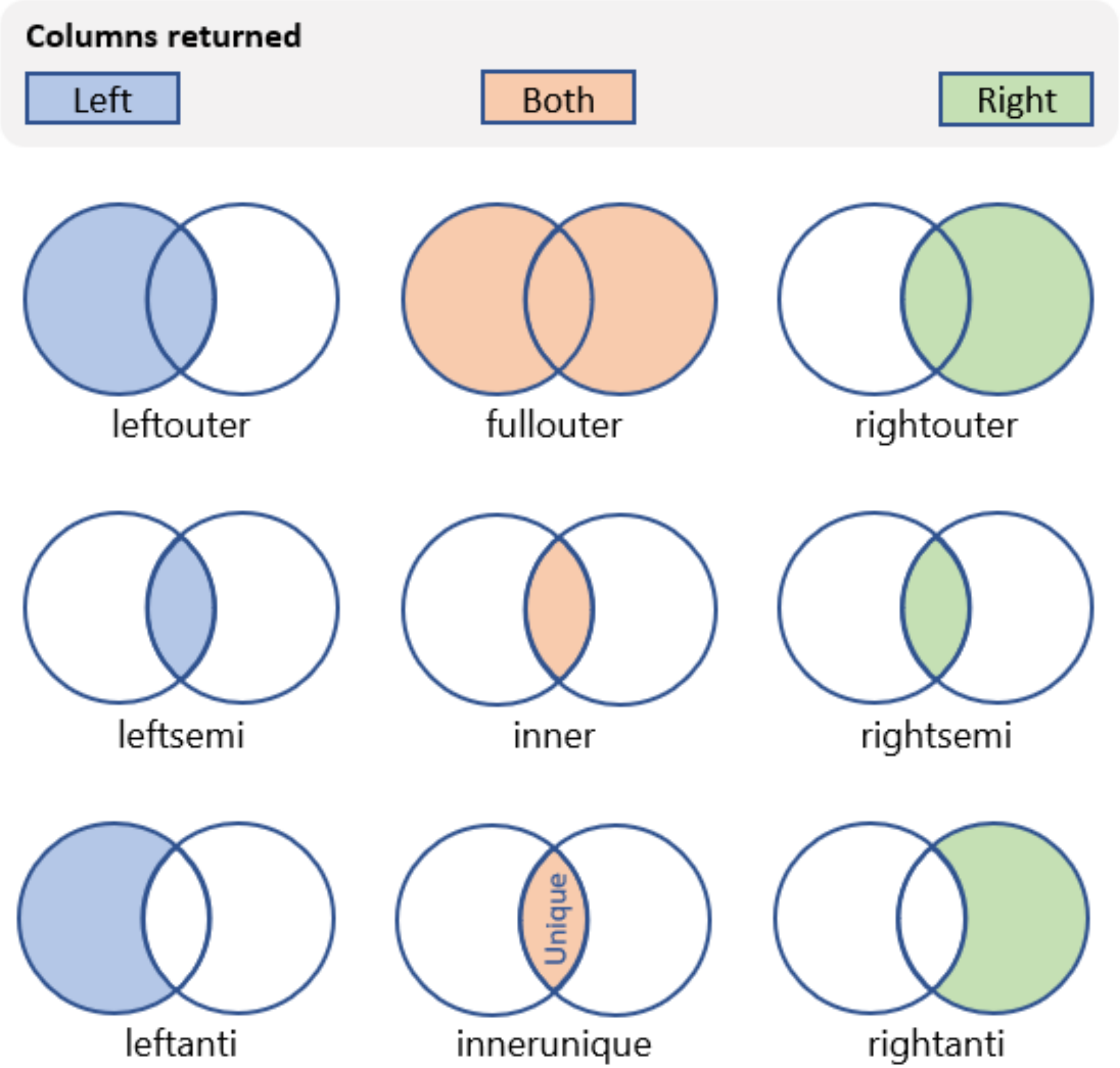


rows (data):



This insight led to a 7240 class project and subsequent paper: Khatiwada, Shraga, Gatterbauer, Miller. Integrating Data Lake Tables. PVLDB 2022. https://doi.org/10.14778/3574245.3574274

VENN diagrams for join types: rows vs column

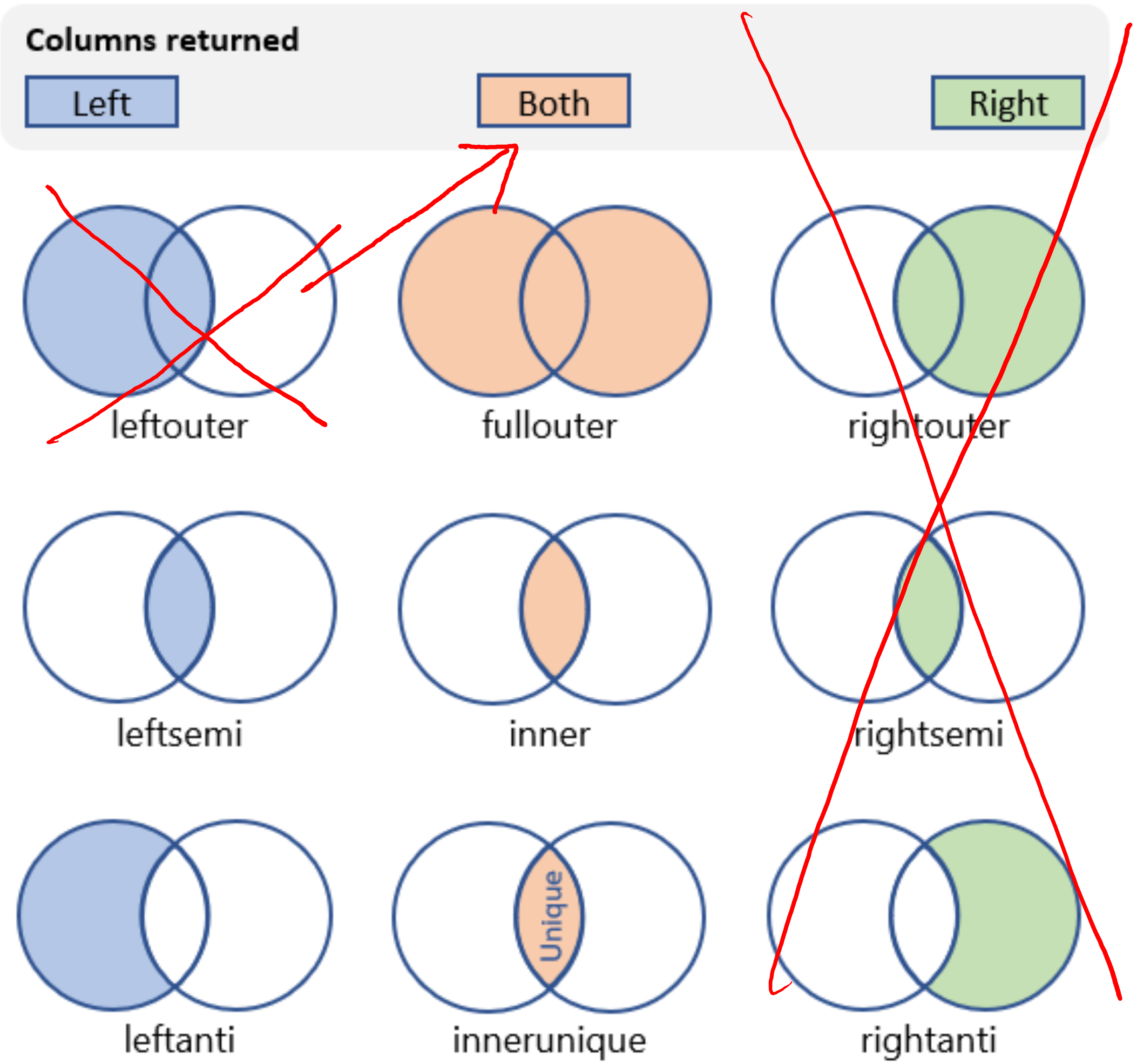


?

Source: https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/join-operator?pivots=azuredataexplorer

VENN diagrams for join types: rows vs column

Leftouter returns both!



"Right" is redundant

?

Source: https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/join-operator?pivots=azuredataexplorer

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") for named perspective
- Derived (or implied) operators
  7. Joins ⋈ (natural, theta join, equi-join, [semi-join: moved to T3-U1])
  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(sid,sname,gpa)

SQL:

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

RA:

?

Alternative to "$A_1 \to B_1$" is the substitution symbol "B_1/A_1" (notice the difference in sequencing)

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

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

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

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

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 "$A_1 \to B_1$" is the substitution symbol "B_1/A_1" (notice the difference in sequencing)

6. Why we need renaming in the named perspective

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 in the named perspective

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 $\rightarrow$ E
1	2
3	4

S

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

R × S

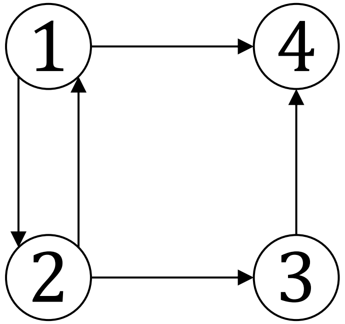
A	R.B	S.B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

$\rho_{B \to E}$(R) × 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 would *really* need renaming if we had R × R:
(Compare to table aliases)

6. Named vs Unnamed perspective (=positional notation)



A for arc or adjacency

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

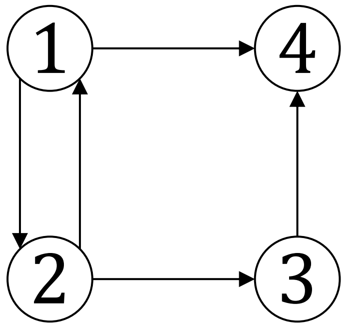
In DRC:

?

A:

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

6. Named vs Unnamed perspective (=positional notation)



A for arc or adjacency

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

In DRC:

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

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

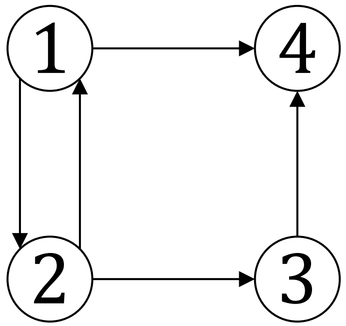
A:

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

In RA:

?

6. Named vs Unnamed perspective (=positional notation)



A for arc or adjacency

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

In DRC:

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

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

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:

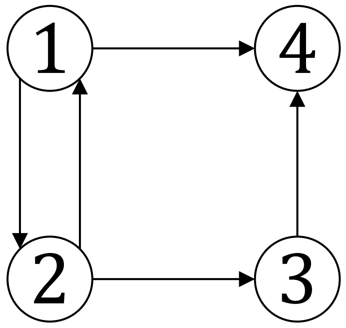
$$\pi_S(\sigma_{T=S2}(A \times \rho_{S\to S2, S\to S2}(A)))$$

?

named perspective

unnamed perspective

6. Named vs Unnamed perspective (=positional notation)



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

In DRC:

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

unnamed= positional

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

In TRC:

?

"named":

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

In RA:

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

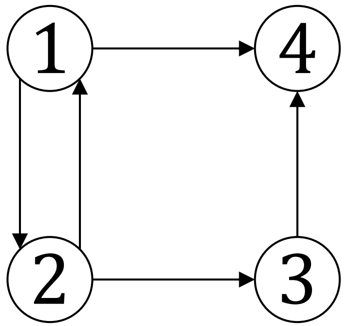
named perspective

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

unnamed perspective

I adopt the notation $2 from [Ullman'99] (also mentioned by [Silberschatz+'20]. It is often just written as "$\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"...

6. Named vs Unnamed perspective (=positional notation)



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

In DRC:

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

unnamed= positional

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

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

A:

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

In TRC:

∃a1∈A, ∃a2∈A

"named":

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

? unnamed for TRC (not typical!)

In RA:

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

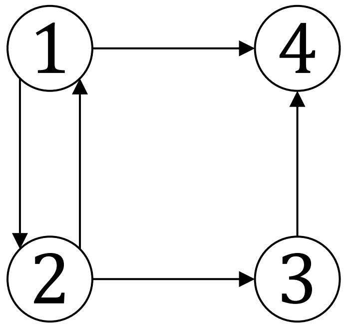
named perspective

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

unnamed perspective

I adopt the notation $2 from [Ullman'99] (also mentioned by [Silberschatz+'20]. It is often just written as "$\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"...

6. Named vs Unnamed perspective (=positional notation)



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

In DRC:

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

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

In TRC: ∃a1∈A, ∃a2∈A

"named":

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

$\{\ q \mid \exists a1, a2 \in A[a1.\$2=a2.\$1 \wedge a1.\$1=q.\$1]\}$

unnamed for TRC (not typical!)

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'99] (also mentioned by [Silberschatz+'20]. It is 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"...

Write in RA

Employee(id, name, salary)

Q: *Find the ID and name of those employees who earn more than the employee whose ID is 123?*

?

Write in RA

Employee(id, name, salary)

Q: *Find the ID and name of those employees who earn more than the employee whose ID is 123?*

$$\pi_{\text{e.id,e.name}}\left(\sigma_{\text{e.salary>o.salary}}(\rho_{\text{e}}(\text{employee})\times\sigma_{\text{id=123}}(\rho_{\text{o}}(\text{employee})))\right)$$

$$\pi_{\text{id,name}}\left(\sigma_{\text{salary>s}}\left(\text{employee}\times(\rho_{\text{salary}\rightarrow\text{s}}(\pi_{\text{salary}}(\sigma_{\text{id=123}}(\text{employee}))))\right)\right)$$

$$\pi_{\$1,\$2}\left(\sigma_{\$4=123\,\wedge\,\$3>\$6}(\text{employee}\times\text{employee})\right)$$

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") for named perspective
- Derived (or implied) operators
  7. Joins $\bowtie$ (natural, theta join, equi-join, [semi-join: moved to T3-U1])
  8. Intersection / complement
  9. Division: $\div$

Derived relational operators:
- can be expressed in basic RA; thus not needed

But enhancing the basic operator set with derived operators is a good idea:
- Queries become easier to write/understand/maintain
- Easier for DBMS to apply specialized optimizations (recall the conceptual evaluation strategy)

we discuss later in class in detail
(SJs are at the heart of efficient algorithms)

most important

7a. Natural Join ($\bowtie$)

Product(<u>pname</u>, price, category, cid)
Company(<u>cid</u>, cname, stockprice, country)

316

- Notation: R $\bowtie$ S
- Joins R and S on equality of all shared attributes
  - Only makes sense in named perspective!
  - If R has attribute set A, and S has attribute set B, and they share attributes A⋂B = C, can also be written as R $\bowtie_C$ S

- Natural join in basic RA:
  - Meaning: R $\bowtie$ S = $\pi_{A \cup B}(\sigma_{R.C=S.C}(R \times S))$
  - Meaning: R $\bowtie$ S = $\pi_{A \cup B}(\sigma_{C=D}(\rho_{C \to D}(R) \times S))$
    - 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

SQL

```sql
SELECT pname, price, category,
P.cid, cname, stockprice, country
FROM Product P, Company C
WHERE P.cid= C.cid
```

SQL (alternative syntax)

?

7a. Natural Join (⋈)

Product(<u>pname</u>, price, category, cid)
Company(<u>cid</u>, cname, stockprice, country)

- Notation: R ⋈ S
- Joins R and S on equality of all shared attributes
  - Only makes sense in named perspective!
  - If R has attribute set A, and S has attribute set B, and they share attributes A⋂B = C, can also be written as R $\bowtie_C$ S

- Natural join in basic RA:
  - Meaning: $R \bowtie S = \pi_{A \cup B}(\sigma_{R.C=S.C}(R \times S))$
  - Meaning: $R \bowtie S = \pi_{A \cup B}(\sigma_{C=D}(\rho_{C \to D}(R) \times S))$
    - 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

SQL

```
SELECT pname, price, category,
P.cid, cname, stockprice, country
FROM Product P, Company C
WHERE P.cid= C.cid
```

SQL (alternative syntax)

```
SELECT *
FROM Product
NATURAL JOIN Company
```

RA:

?

7a. Natural Join (⋈)

Product(<u>pname</u>, price, category, cid)
Company(<u>cid</u>, cname, stockprice, country)

316

- Notation: R ⋈ S
- Joins R and S on equality of all shared attributes
  - Only makes sense in named perspective!
  - If R has attribute set A, and S has attribute set B, and they share attributes A⋂B = C, can also be written as R $\bowtie_C$ S
- Natural join in basic RA:
  - Meaning: $R \bowtie S = \pi_{A \cup B}(\sigma_{R.C=S.C}(R \times S))$
  - Meaning: $R \bowtie S = \pi_{A \cup B}(\sigma_{C=D}(\rho_{C \to D}(R) \times S))$
    - 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

SQL

```
SELECT pname, price, category,
P.cid, cname, stockprice, country
FROM Product P, Company C
WHERE P.cid= C.cid
```

SQL (alternative syntax)

```
SELECT *
FROM Product
NATURAL JOIN Company
```

RA:

$$\text{Product} \bowtie \text{Company}$$

7a. Natural Join (⋈): an alternative perspective

We only want to pair those tuples that match in some way.

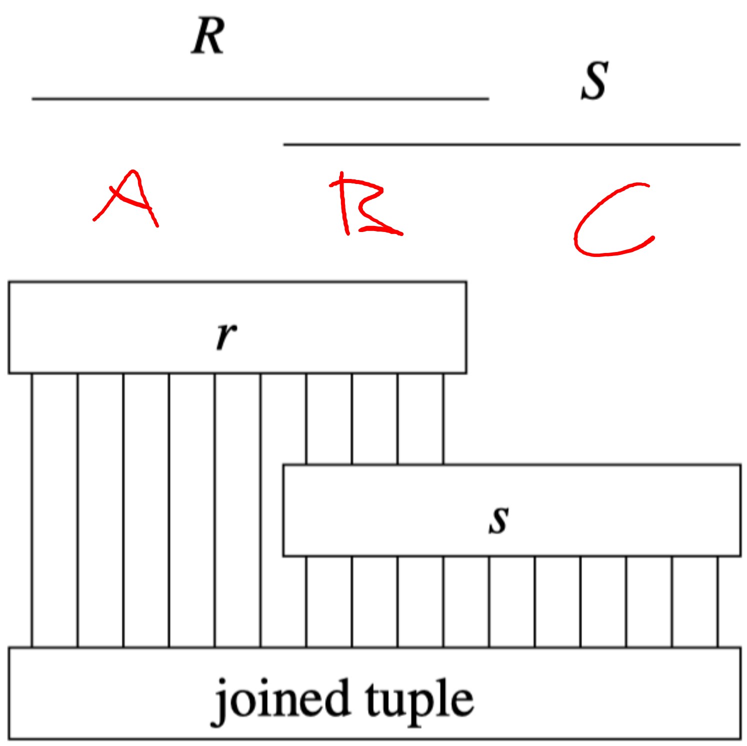


Figure 15: Joining tuples

More formally the semantics of the natural join are defined as follows:

$$R \bowtie S = \{ r \cup s \mid r \in R \ \wedge \ s \in S \ \wedge \ Fun(r \cup s) \} \quad \textbf{(1)}$$

where *Fun(t)* is a predicate that is true for a relation *t* (in the mathematical sense) iff *t* is a function. It is usually required that *R* and *S* must have at least one common attribute, but if this constraint is omitted, and *R* and *S* have no common attributes, then the natural join becomes exactly the Cartesian product.

7a. Natural Join (⋈): An example

R

A	B
1	2
3	4

S

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

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

?

7a. Natural Join (⋈): An example

R

A	**B**
1	2
3	4

S

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

R ⋈ S

?

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

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

7a. Natural Join (⋈): An example

R

A	B
1	2
3	4

S

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

R ⋈ S

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

R ⋈ S = in basic RA

?

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

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~~

7a. Natural Join (⋈): 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

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

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~~

$R \bowtie S =$

$\Pi_{A,R.B,C,D}(\sigma_{R.B=S.B}(R \times S)) =$

$\Pi_{A,B,C,D}(\sigma_{B=E}(\rho_{B \to E}(R) \times S)) =$

$\Pi_{\$1,\$2,\$4,\$5}(\sigma_{\$2=\$3}(R \times S)) =$

7a. Natural Join (⋈): practice

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

?

7a. Natural Join (⋈): practice

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

Answer(A, B, C, D,E)

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

?

7a. Natural Join (⋈): practice

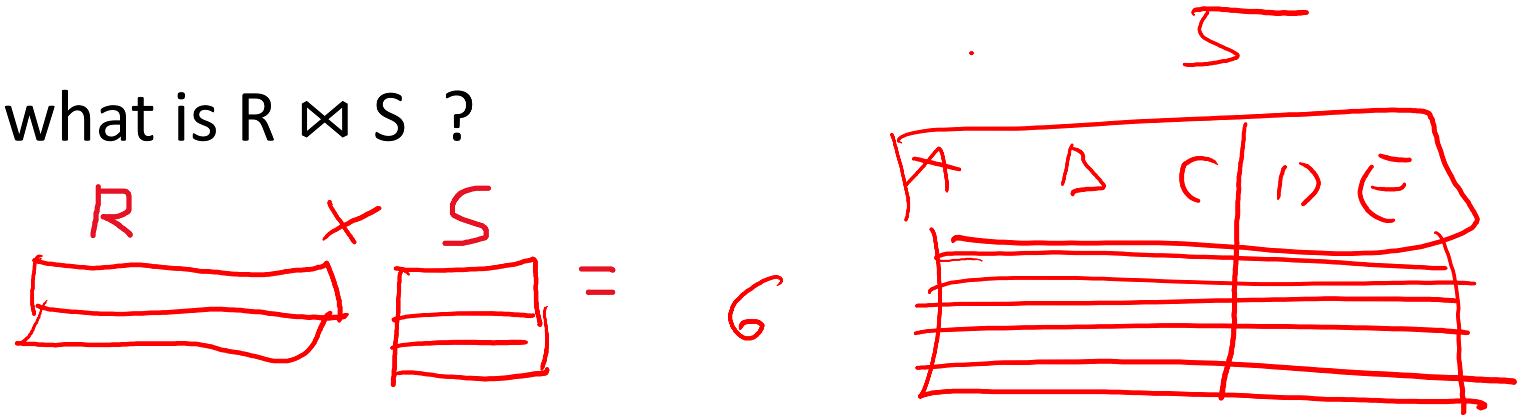
- Given schemas R(A, B, C, D), S(A, C, E), what is the schema of R ⋈ S ?

Answer(A, B, C, D,E)

no condition in the selection
that could be violated:

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

R × S



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

?

7a. Natural Join (⋈): practice

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

Answer(A, B, C, D,E)

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

$R \times S$

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

$R \cap S$

R ⋈ S =

A	B
1	2
1	3
2	2

A	B
1	2
3	1
2	2

?

7a. Natural Join (⋈): practice

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

  Answer(A, B, C, D,E)

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

  $R \times S$

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

  $R \cap S$

R ⋈ S =

A	B
1	2
1	3
2	2

A	B
1	2
3	1
2	2

Ans

A	B
1	2
2	2

7b. Theta Join ($\bowtie_\theta$)

- A join that involves a predicate

$$\boxed{R_1 \bowtie_\theta R_2 = \sigma_\theta(R_1 \times R_2)}$$

- θ ("theta") can be any condition
- No projection: #attributes in output = sum #attributes in input

Note that natural join is a theta join + a selection

- Example: band-joins for approx. matchings across tables

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

Assume relatively fresh data (within 1 year)

?

7b. Theta Join ($\bowtie_\theta$)

- A join that involves a predicate

$$R_1 \bowtie_\theta R_2 = \sigma_\theta(R_1 \times R_2)$$

- θ ("theta") can be any condition
- No projection: #attributes in output = sum #attributes in input

Note that natural join is a theta join + a selection

- Example: band-joins for approx. matchings across tables

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

Assume relatively fresh data (within 1 year)

$$A \bowtie_{P.zip=V.zip \wedge (P.age>=V.age -1 \wedge P.age<=V.age +1)} V$$

Student(sid,name,gpa)
People(ssn,name,address)

SQL:

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

RA:

?

7b. Theta Join ($\bowtie_\theta$)

- A join that involves a predicate

$$R_1 \bowtie_\theta R_2 = \sigma_\theta(R_1 \times R_2)$$

- θ ("theta") can be any condition
- No projection: #attributes in output = sum #attributes in input

Note that natural join is a theta join + a selection

- Example: band-joins for approx. matchings across tables

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

Assume relatively fresh data (within 1 year)

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

Student(sid,name,gpa)
People(ssn,name,address)

SQL:

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

RA:

$$\text{Students} \bowtie_\theta \text{People}$$

7c. Equi-join ($\bowtie_{A=B}$)

- A theta join where q is an equality

$$\boxed{R_1 \bowtie_{A=B} R_2 = \sigma_{A=B}(R_1 \times R_2)}$$

- Example over Gizmo DB:
  - Product $\bowtie_{\text{manufacturer=cname}}$ Company
- Most common join in practice!

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

SQL:

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

RA:

?

7c. Equi-join ($\bowtie_{A=B}$)

- A theta join where q is an equality

$$\boxed{R_1 \bowtie_{A=B} R_2 = \sigma_{A=B}(R_1 \times R_2)}$$

- Example over Gizmo DB:
  - Product $\bowtie_{\text{manufacturer=cname}}$ Company
- Most common join in practice!

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

SQL:

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

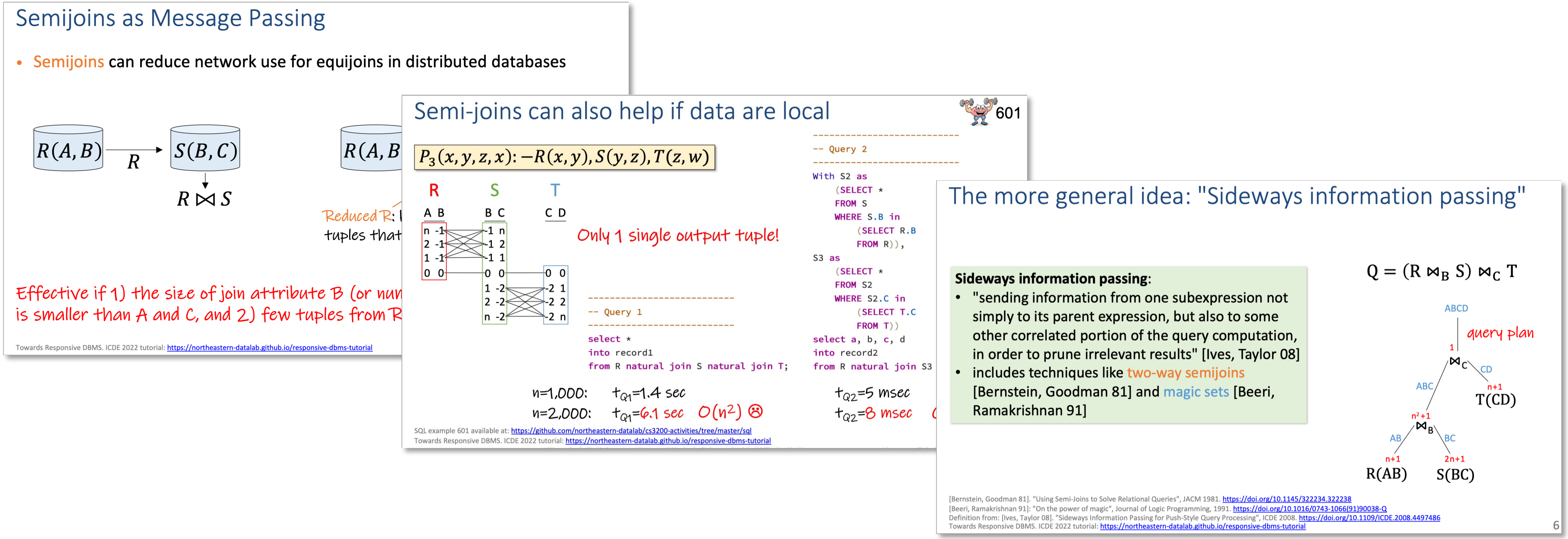
RA:

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

What is the connection with a natural join? ?

7d. Semi-join (⋉) [moved to T3-U1]

- $R \ltimes S$: Return tuples from R for which there is a matching tuple in S that is equal on their common attribute names.



See "Part 3: Acyclic queries & Enumeration": https://northeastern-datalab.github.io/responsive-dbms-tutorial/slides/Responsive-DBMS-tutorial-part-3-AcyclicQueries-Enumeration.pdf, https://www.youtube.com/watch?list=PL_72ERGKF6DTInW_P3a9zTYPSNLwbqOAx&v=toi7ysuyRkw from ICDE'22 tutorial "Toward Responsive DBMS: Optimal Join Algorithms, Enumeration, Factorization, Ranking, and Dynamic Programming" by Tziavelis et al. https://doi.org/10.1109/ICDE53745.2022.00299, https://northeastern-datalab.github.io/responsive-dbms-tutorial/

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

Example: Converting SFW Query to RA

Student(sid,name,gpa)
People(ssn,name,address)

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

How do we represent this query in RA?

⇨ ?

Example: Converting SFW Query to RA

Student(sid,name,gpa)
People(ssn,name,address)

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

How do we represent this query in RA?

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

$$\Pi_{\text{gpa,address}}(\sigma_{\text{gpa}>3.5\, \wedge\, \text{S.name}=\text{P.name}}(S \times P))$$

$$\Pi_{\text{gpa,address}}(\sigma_{\text{gpa}>3.5\, \wedge\, \text{name}=\text{name2}}(S \times \rho_{\text{name} \rightarrow \text{name2}} P))$$

Some Examples

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

Find names of suppliers of parts with size greater than 10

?

Find names of suppliers of red parts or parts with size greater than 10

?

Some Examples

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

Find names of suppliers of parts with size greater than 10

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

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

Find names of suppliers of red parts or parts with size greater than 10

?

Some Examples

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

Find names of suppliers of parts with size greater than 10

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

$\pi_{sname}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{psize>10}(\text{Part}))$ → Representation of RA as tree?

?

Find names of suppliers of red parts or parts with size greater than 10

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

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

Some Examples

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

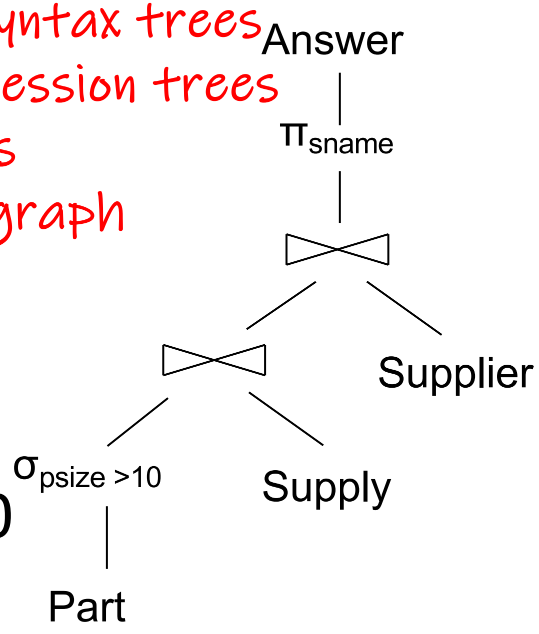
Usually unary or binary. Think of:
- abstract syntax trees
- binary expression trees
- parse trees
- data flow graph

Find names of suppliers of parts with size greater than 10

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

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

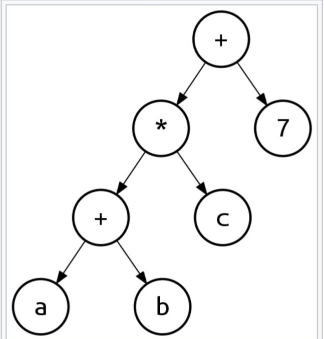
Representation of RA as tree?



Find names of suppliers of red parts or parts with size greater than 10

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

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



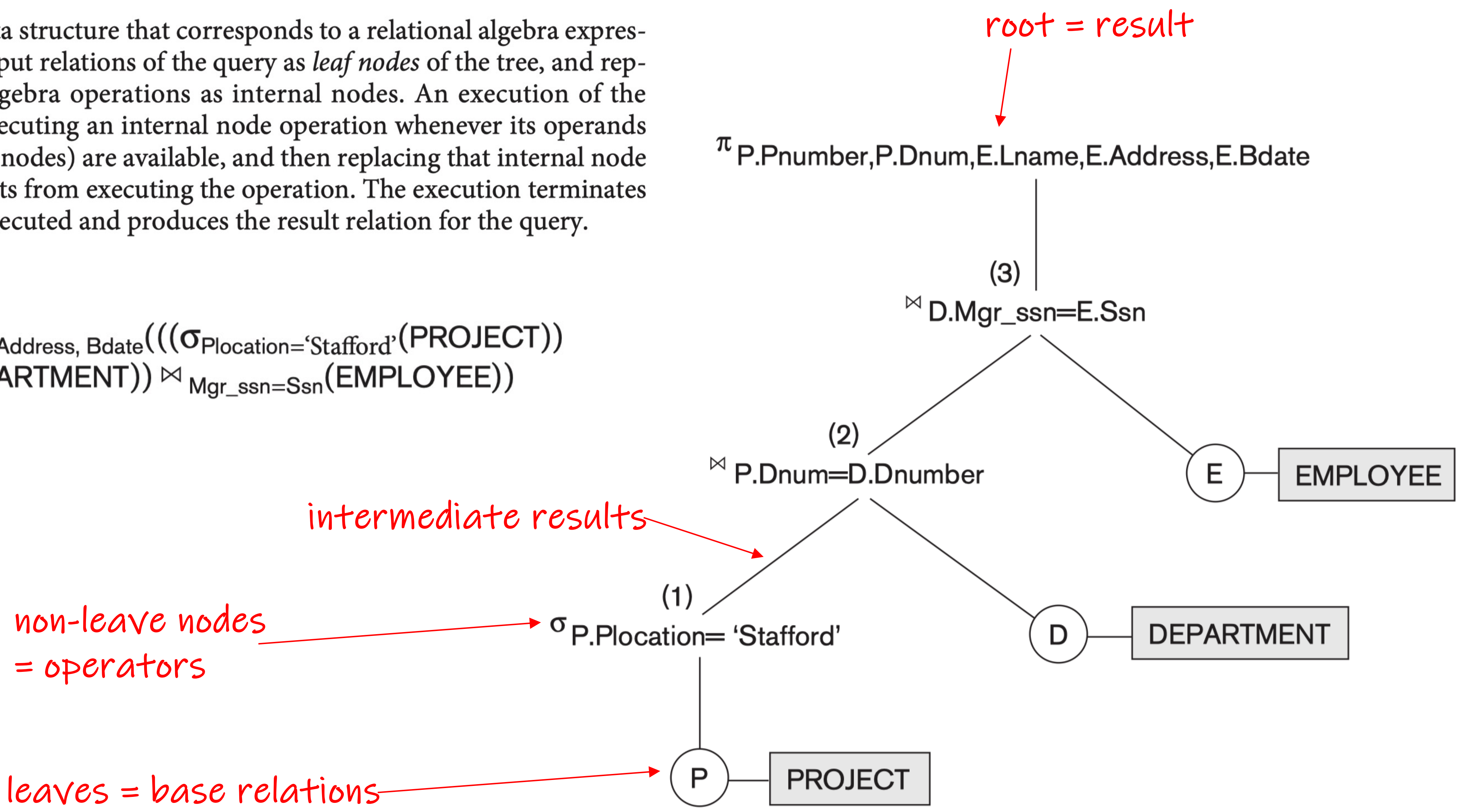
Expression tree of the expression (a+b)*c+7

https://en.wikipedia.org/wiki/Binary_expression_tree

Query (Evaluation / Execution) Tree, Data flow graph

A **query tree** is a tree data structure that corresponds to a relational algebra expression. It represents the input relations of the query as *leaf nodes* of the tree, and represents the relational algebra operations as internal nodes. An execution of the query tree consists of executing an internal node operation whenever its operands (represented by its child nodes) are available, and then replacing that internal node by the relation that results from executing the operation. The execution terminates when the root node is executed and produces the result relation for the query.

$$\pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}(((\sigma_{\text{Plocation='Stafford'}}(\text{PROJECT})) \bowtie_{\text{Dnum=Dnumber}}(\text{DEPARTMENT})) \bowtie_{\text{Mgr\_ssn=Ssn}}(\text{EMPLOYEE}))$$



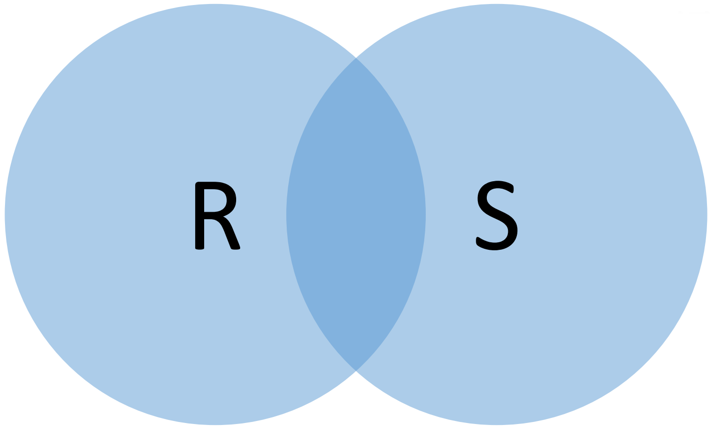
Relational Algebra (RA) operators

- 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") for named perspective
- Derived (or implied) operators
  7. Joins $\bowtie$ (natural, theta join, equi-join, [semi-join: moved to T3-U1])
  8. Intersection / complement
  9. Division: $\div$

8. What about Intersection ∩?

- As derived operator using union and minus

?

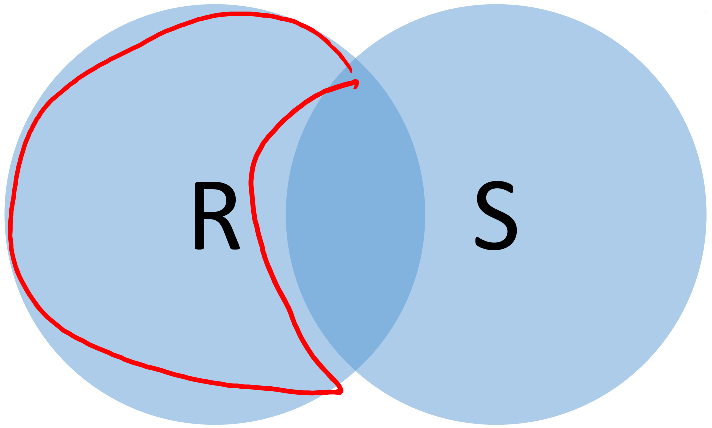


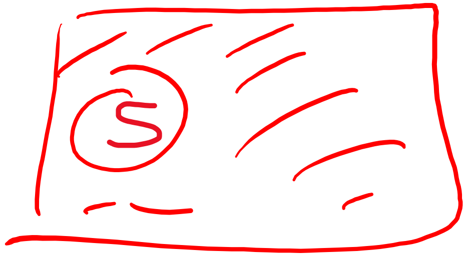
8. What about Intersection ∩?

- As derived operator using union and minus

(R-S)

?





$S^c = D - S$

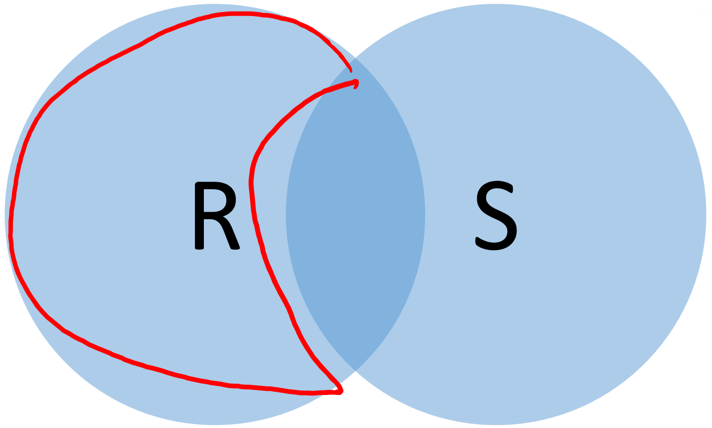
R

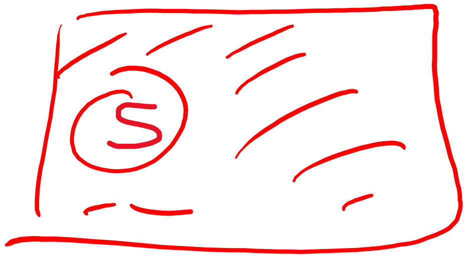
8. What about Intersection $\cap$?

- As derived operator using union and minus

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

?





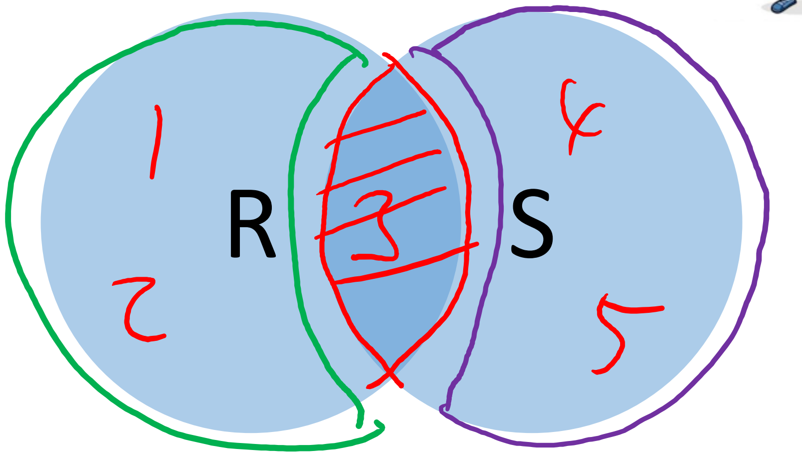
$S^c = \Delta - S$

R

8. What about Intersection ∩?

- As derived operator using union and minus

$(R \cup S) - ((\underline{R - S}) \cup (\underline{S - R}))$



$\{1, 2, 3\} \cap \{3, 4, 5\} = 3$

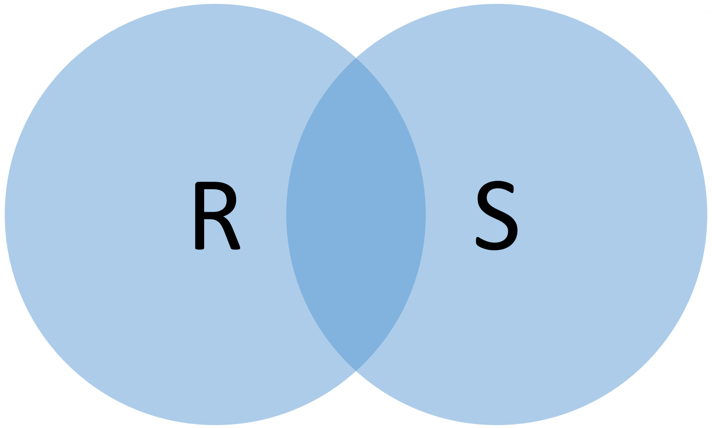
8. What about Intersection ∩?

- As derived operator using union and minus

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

- Derived operator using minus only!

?



8. What about Intersection ∩?

- As derived operator using union and minus

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

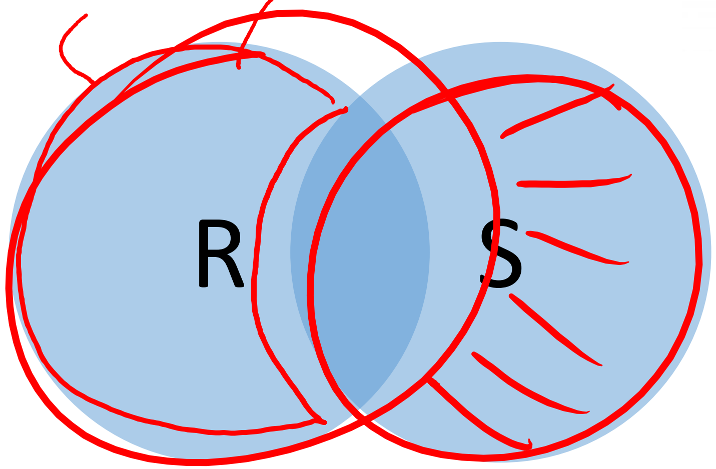
- Derived operator using minus only!

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

- Derived using join

?

R − (R − S)

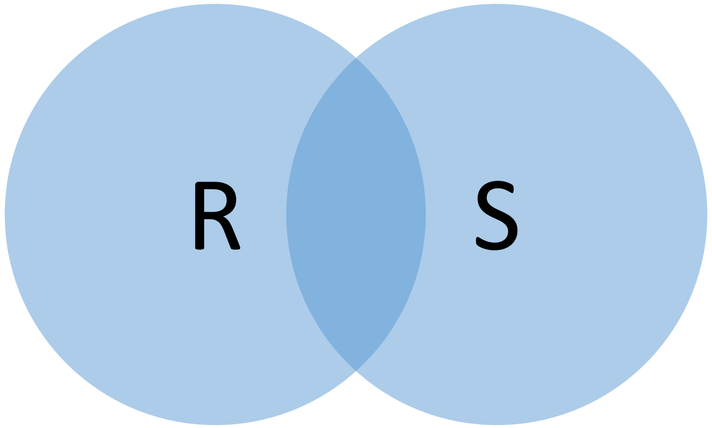


S − (S − R)

8. What about Intersection ∩?

- As derived operator using union and minus

$$\boxed{\begin{array}{l} R \cap S = ((R \cup S) - (R - S)) - (S - R) \\ R \cap S = (R \cup S) - ((R - S) \cup (S - R)) \end{array}}$$



- Derived operator using minus only!

$$\boxed{R \cap S = \qquad S \qquad - (S - R)}$$

- Derived using join

$$\boxed{R \cap S = R \bowtie S}$$

Legal input: schemas need to be union compatible (same schema). E.g. not:
R(A,B,C)
S(A,B)

If R and S have the same schema, then $R \bowtie S$ and $R \ltimes S$ equal to $R \cap S$

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") for named perspective
- Derived (or implied) operators
  7. Joins $\bowtie$ (natural, theta join, equi-join, [semi-join: moved to T3-U1])
  8. Intersection / complement
  9. Division: $\div$

9. Division (R ÷ S)

- Consider two relations R(X,Y) and S(Y)
- Then R ÷ S is ...

X, Y are sets of attributes
Legal input: att(R) ⊃ att(S)

What could be a meaningful definition of division ?

Compare to Integer division: 7/2=3

3 is the biggest integer that multiplied with 2 is smaller or equal to 7

9. Division ($R \div S$)

- Consider two relations R(X,Y) and S(Y)
- Then $R \div S$ is ...
  - ... the largest relation T(X) s.t. $S \times T \subseteq R$

X, Y are sets of attributes
Legal input: att(R) $\supset$ att(S)

(safety: $T \subseteq \pi_X R$)

9. Division (R ÷ S)

- Consider two relations R(X,Y) and S(Y)
- Then R ÷ S is ...
  - ... the largest relation T(X) s.t. S × T ⊆ R, or
  - ... the relation T(X) that contains the X's that occur with all Y's in S, or
  - ... {t(X) | ∀s(Y)∈S.[∃r(X,Y)∈R]} (+ safety)

X, Y are sets of attributes
Legal input: att(R) ⊃ att(S)

(safety: $T \subseteq \pi_X R$)

R Dividend

X	Y
Alice	1
Alice	2
Bob	1
Bob	2
Bob	3

S Divisor

Y
1
2
3

T

?

9. Division ($R \div S$)

- Consider two relations R(X,Y) and S(Y)
- Then $R \div S$ is ...
  - ... the largest relation T(X) s.t. $S \times T \subseteq R$, or
  - ... the relation T(X) that contains the X's that occur with all Y's in S, or
  - ... $\{t(X) \mid \forall s(Y) \in S.[\exists r(X,Y) \in R]\}$ (+ safety)

X, Y are sets of attributes
Legal input: att(R) ⊃ att(S)

(safety: $T \subseteq \pi_X R$)

R Dividend

X	Y
Alice	1
Alice	2
Bob	1
Bob	2
Bob	3
~~A~~	~~3~~

S Divisor

Y
1
2
3

T

X
Bob

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

$\div$

Course

course
ML

$=$?

$\div$

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

$\div$

Course

course
ML

$=$

sid	student
2	Bob
3	Charly

Studies $\div$

course
AI
DB
ML

$=$

sid	student
3	Charly

Assume R,S have disjoint attribute sets (possibly by renaming)

$(R\text{x}S)\div S =$?

$(R\text{x}S)\div R =$?

Based on material by Benny Kimelfeld and Oded Shmueli for 236363 Database Management Systems, Technion, 2018.
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

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

Assume R,S have disjoint attribute sets (possibly by renaming)

$$(R \times S) \div S = R$$

$$(R \times S) \div R = S$$

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

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

Assume R,S have disjoint attribute sets (possibly by renaming)

$(R \times S) \div S = R$

$(R \times S) \div R = S$

Q: If R has 1000 tuples and S has 100 tuples, how many tuples can be in R÷S? 0-10

Q: If R has 1000 tuples and S has 1001 tuples, how many tuples can be in R÷S? 0

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 Type

course	type
AI	elective
DB	core
ML	core

Who took all core courses in RA with relational division?

?

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 Type

course	type
AI	elective
DB	core
ML	core

Who took all core courses in RA with relational division?

$$\text{Studies} \div \pi_{\text{course}}\left(\sigma_{\text{type}='\text{core}'}\text{CourseType}\right)$$

Based on material by Benny Kimelfeld and Oded Shmueli for 236363 Database Management Systems, Technion, 2018.
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

How to write R÷S in Primitive RA? (×,−, π)

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

R ÷ S = Q

X	Y
a	0
a	1
a	2
b	1

Y
1
2

?

How to write R÷S in Primitive RA? (×,−, π)

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

? in primite RA

R ÷ S = Q

X	Y
a	0
a	1
a	2
b	1
b	2

Y
1
2

X
a

1

2

3

4: {a} = {a,b} – {b}

How to write R÷S in Primitive RA? (×,–, π)

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

$$\boxed{\pi_X R \times S}$$

Each X of R w/ each Y of S

R ÷ S = Q

R:

X	Y
a	0
a	1
a	2
b	1
b	2

S:

Y
1
2

Q:

X
a

1

2

3

4: {a} = {a,b} – {b}

How to write R÷S in Primitive RA? (×,−, π)

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

$$\boxed{\left(\boxed{\pi_X R \times S}\right) - R}$$

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:

X	Y
a	0
a	1
a	2
b	1
b	2

S:

Y
1
2

Q:

X
a

1

2

3

4: {a} = {a,b} – {b}

How to write R÷S in Primitive RA? (×,−, π)

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

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

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 = Q

R:

X	Y
a	0
a	1
a	2
b	1
b	2

S:

Y
1
2

Q:

X
a

1

2

3

4: {a} = {a,b} – {b}

How to write R÷S in Primitive RA? (×,−, π)

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

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

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 ÷ S = Q

X	Y
a	0
a	1
a	2
b	1
b	2

Y
1
2

X
a

1

2

3

4: {a} = {a,b} – {b}

What if S=Ø?

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

$R \quad \div \quad S = Q$

X	Y
a	0
a	1
a	2
b	1

Y

?

What if $S=\emptyset$?

DOM = {a, b, c}

$R \div S = Q$

X	Y
a	0
a	1
a	2
b	1

Y

X
a
b

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

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

Recall: $\{(x) \mid \forall s(y) \in S.[\exists r(x,y) \in R]\}$ (+ safety)

Now you see why we needed the safety condition "$T \subseteq \pi_X R$" when defining "$R \div S$ as the largest relation $T(X)$ s.t. $S \times T \subseteq R$"

R÷S in Primitive RA vs. RC

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

In RA:

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

In DRC: ?

R ÷ S = Q

X	Y
a	0
a	1
a	2
b	1
b	2

Y
1
2

X
a

R÷S in Primitive RA vs. RC

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

In RA:

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

In DRC:

$$\{\, x \mid \exists z.[R(x,z)] \wedge \quad ? \quad \}$$

x is "guarded": safe and thus domain independent

$R \div S = Q$

R:

X	Y
a	0
a	1
a	2
b	1
b	2

S:

Y
1
2

Q:

X
a

R÷S in Primitive RA vs. RC

DOM = {a.b.c}

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

In RA:

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

In DRC: what if S(Y)=∅ ?

?

$$\{\, x \mid \exists z.[R(x,z)] \wedge \forall y.[S(y) \rightarrow R(x,y)] \,\}$$

R ÷ S = Q

R:

X	Y
a	0
a	1
a	2
b	1
b	2

S:

Y
1
2

Q:

X
a

R÷S in Primitive RA vs. RC

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

In RA:

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

In DRC:

$$\{\, x \mid \exists z.[R(x,z)] \land \forall y.[S(y) \rightarrow R(x,y)] \,\}$$

? without universal quantification

R ÷ S = Q

X	Y
a	0
a	1
a	2
b	1
b	2

Y
1
2

X
a

R÷S in Primitive RA vs. RC

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

R ÷ S = Q

X	Y
a	0
a	1
a	2
b	1
b	2

Y
1
2

X
a

In RA:

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

In DRC:

$$\{\, x \mid \exists z.[R(x,z)] \wedge \forall y.[S(y) \rightarrow R(x,y)] \,\}$$

$$\{\, x \mid \exists z.[R(x,z)] \wedge \nexists y.[S(y) \wedge \neg R(x,y)] \,\}$$

In TRC:

?

R÷S in Primitive RA vs. RC

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

R		÷	S	=	Q
X	Y		Y		X
a	0		1		a
a	1		2		
a	2				
b	1				
b	2				

In RA:

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

In DRC:

$$\{ x \mid \exists z.[R(x,z)] \wedge \forall y.[S(y) \rightarrow R(x,y)] \}$$

$$\{ x \mid \exists z.[R(x,z)] \wedge \nexists y.[S(y) \wedge \neg R(x,y)] \}$$

In TRC:

$$\{ r.X \mid r \in R.[\ \nexists s \in S.[\quad ? \quad]]\}$$

R÷S in Primitive RA vs. RC

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

R ÷ S = Q

X	Y
a	0
a	1
a	2
b	1
b	2

Y
1
2

X
a

In RA:

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

In DRC:

$$\{\, x \mid \exists z.[R(x,z)] \land \forall y.[S(y) \rightarrow R(x,y)] \,\}$$

$$\{\, x \mid \exists z.[R(x,z)] \land \nexists y.[S(y) \land \neg R(x,y)] \,\}$$

? in SQL

In TRC:

$$\{\, r.X \mid r \in R.[\, \nexists s \in S.[\nexists r_2 \in R.[r_2.Y = s.Y \land r_2.X = r.X]]] \}$$

R÷S in Primitive RA vs. RC

In SQL

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

$R \div S = Q$

X	Y
a	0
a	1
a	2
b	1
b	2

Y
1
2

X
a

In TRC:

$$\{\, r.X \mid r \in R.[\, \nexists s \in S.[\nexists r_2 \in R.[r_2.Y = s.Y \wedge r_2.X = r.X]]]\}$$

RA vs. RC

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

There are logical expressions that cannot be expressed in basic RA with the same number of table references

In RA:

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

3 references to R in RA, but only 2 references in RC

In DRC:

$$\{\, x \mid \exists z.[R(x,z)] \wedge \forall y.[S(y) \rightarrow R(x,y)] \,\}$$

In TRC:

$$\{\, r.X \mid r{\in}R.[\, \nexists s{\in}S.[\nexists r_2{\in}R.[r_2.Y{=}s.Y \wedge r_2.X{=}r.X]]]\}$$

On the Reasonable Effectiveness of Relational Diagrams: Explaining Relational Query Patterns and the Pattern Expressiveness of Relational Languages, SIGMOD 2024. https://arxiv.org/pdf/2401.04758

R÷S as set-containment join (not part of standard RA)

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

R ÷ S = Q

X	Y
a	0
a	1
a	2
b	1

Y
1
2

X
a

As set containment join

$$R \bowtie_{R.Y \supseteq S.Y} S$$

In DRC (extended with set containment):

$$\{\, x \mid \{y \mid R(x,y)\} \supseteq \{\, y \mid S(y)\} \,\}$$

Set-containment joins (not part of standard RA)

$R \bowtie_{B \supseteq C} S = Q$

A	B
a	0
a	1
a	2
b	1

C	D
1	a
2	a
1	b

A	D
a	a
a	b
b	b

More general set containment join

$R \bowtie_{B \supseteq C} S$

In DRC (extended with set containment):

$$\{(A,D) \mid \{B \mid R(A,B)\} \supseteq \{ C \mid S(C,D)\} \}$$

Set-containment joins generalize equi-joins

$$\pi_{-B,C}(R \bowtie_{B=C} S) = Q$$

R:

A	B
a	0
a	1
a	2
b	1

S:

C	D
1	a
2	a
1	b

Q:

A	D
a	a
a	b
b	b
b	a

Equi-join as instance of set intersection join

?

Set-containment joins generalize equi-joins

$$\pi_{-B,C}(R \bowtie_{B=C} S) = Q$$

A	B
a	0
a	1
a	2
b	1

C	D
1	a
2	a
1	b

A	D
a	a
a	b
b	b
b	a

"non-empty set intersection" join

$$R \bowtie_{B \cap C \neq \emptyset} S \equiv \pi_{-B,C}(R \bowtie_{B=C} S)$$

In DRC (extended with set containment):

$$\{(A,D) \mid \{B \mid R(A,B)\} \cap \{ C \mid S(C,D)\} \neq \emptyset \}$$

$$\{(A,D) \mid \exists B\, [\, R(A,B) \wedge S(B,D)\,]\, \}$$

Parentheses Convention

- We have defined 3 unary operators (w/ renaming) and 3 binary operators
- It is acceptable to omit the parentheses from o(R) when o is unary
  - Then, unary operators take precedence over binary ones
- Example:

$$(\sigma_{course='DB'}(\text{Course})) \times (\rho_{cid \rightarrow cid1}(\text{Studies}))$$

becomes

$$\sigma_{course='DB'}\text{Course} \times \rho_{cid \rightarrow cid1}\text{Studies}$$

Updated 2/6/2024

Topic 1: Data models and query languages
Unit 3: Relational Algebra (RA)
Lecture 8

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp24)

https://northeastern-datalab.github.io/cs7240/sp24/

2/6/2024

Pre-class conversations

- Last class summary
- Please keep on pointing out any errors on the slides
- It is time to start to hand in your first scribes (some ideas today)
- Project discussions (in 2 weeks: Fri 2/16: project ideas)

- today:
  - ~~we continue with relational algebra (RA)~~
  - next week: equivalence of RA and *safe* RC (Codd's theorem)
- ~~next time~~: TODAY
  - Recursion (Datalog)

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)
  - RA $\rightarrow$ RC
  - RC $\rightarrow$ RA

5 Primitive Operators

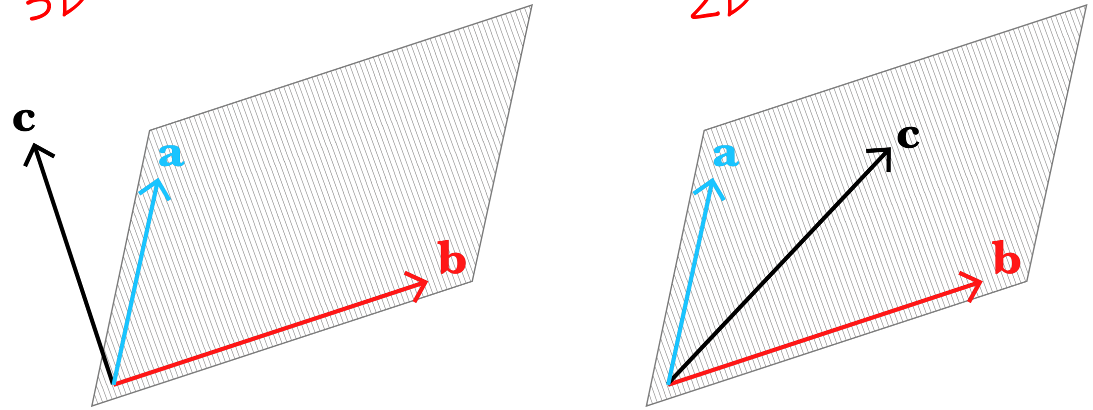
1. Projection (π)
2. Selection (σ)
3. Union ($\cup$)
4. Set Difference ($-$)
5. Cross Product ($\times$)

Is this a well chosen set of primitives? ?

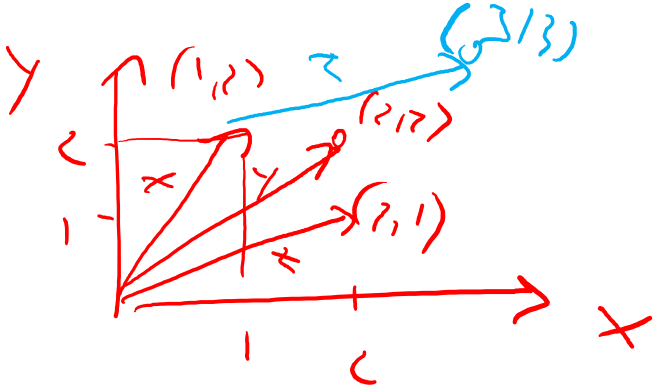
5 Primitive Operators

1. Projection (π)
2. Selection (σ)
3. Union ($\cup$)
4. Set Difference ($-$)
5. Cross Product ($\times$)

3D 2D



independent not independent



$y = (x + z) \cdot \frac{2}{3}$

$2x + 3y + 2z = 0$

$\frac{2}{3} \quad -1 \quad \frac{2}{3}$

Is this a well chosen set of primitives?

Could we drop an operator "without losing anything"?

Independence among Primitives

- Let o be an RA operator, and let A be a set of RA operators
- We say that o is independent of A if o cannot be expressed in A; that is, no expression in A is equivalent to o

THEOREM: Each of the five primitives is independent of the other four

$\{\pi, \sigma, \times, \cup, -\}$

Proof:

- Separate argument for each of the 5 (For each operator, we need to discover a property that is uniquely possessed by that operator, and thus not by any RA expression that involves only the other 4 operations)
- Arguments follow a common pattern (union as example next slides)

Recipe for Proving Independence of an operator o

1. Fix a schema S and an instance D over S

2. Find some property P over relations

3. Prove: for every expression φ that does not use o, the relation φ(D) satisfies P

   Such proofs are typically by induction on the size of the expression, since operators compose

4. Find an expression ψ such that ψ uses o and ψ(D) violates P

Concrete Example: Proving Independence of Union ∪

1. Fix a schema *S* and an instance D over *S*

   S: R(A), S(A)   D: {R(0), S(1)}

R

A
0

S

A
1

2. Find some property P over relations

   #tuples < 2

3. Prove: for every expression φ that does not use ∘, the relation $\varphi(D)$ satisfies P

   Induction base: R and S have #tuples<2

   Induction step: If $\varphi_1(D)$ and $\varphi_2(D)$ have #tuples<2, then so do:

   $\sigma_c(\varphi_1(D))$, $\pi_A(\varphi_1(D))$, $\varphi_1(D) \times \varphi_2(D)$, $\varphi_1(D) - \varphi_2(D)$, $\rho_{A \to B}(\varphi_1(D))$

4. Find an expression ψ such that ψ uses ∘ and $\psi(D)$ violates P

   $\psi = R \cup S$

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)
  - RA $\rightarrow$ RC
  - RC $\rightarrow$ RA

Commutativity and distributivity of RA operators

- The basic commutators:
  - Push <u>projection</u> through selection, join, union
  - Push <u>selection</u> through projection, join, union
  - Also: Joins can be re-ordered!

$\pi_{\mathbf{A}}(R \cup S) = \pi_{\mathbf{A}}(R) \cup \pi_{A}(S)$

$\sigma_{\theta}(R \cup S) = \sigma_{\theta}(R) \cup \sigma_{\theta}(S)$

$(R \cup S) \times T = (R \times T) \cup (S \times T)$

- Note that this is not an exhaustive set of operations

What about sorting and joins?

This simple set of tools allows us to greatly improve the execution time of queries by optimizing RA plans!

We next illustrate with an SFW (Select-From-Where) query

An example: SQL to RA to Optimized RA

```
R(A,B) S(B,C) T(C,D)
```

```sql
SELECT R.A,T.D
FROM   R,S,T
WHERE  R.B = S.B
   and S.C = T.C
   and R.A < 10;
```

in RA

?

An example: SQL to RA to Optimized RA

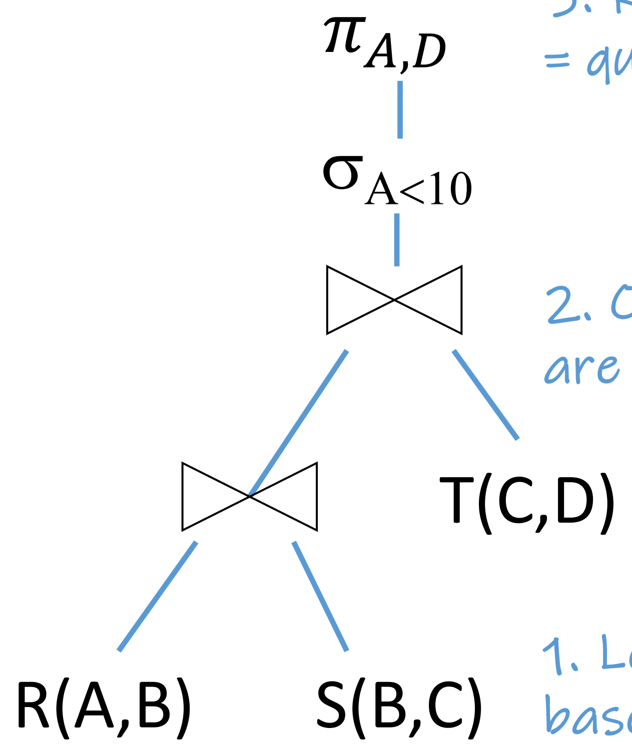
Heuristic: have selection and projection earlier to have fewer (or smaller) "intermediate" tuples

```
R(A,B) S(B,C) T(C,D)
```

```sql
SELECT R.A,T.D
FROM   R,S,T
WHERE  R.B = S.B
   and S.C = T.C
   and R.A < 10;
```

in RA

$$\pi_{A,D}\left(\sigma_{A<10}\left(T \bowtie (R \bowtie S)\right)\right)$$



3. Root node = query results

2. Other nodes are operators

1. Leaves are base relations

Query tree / expression tree / computation tree / data flow graph

An example: SQL to RA to Optimized RA

Heuristic: have selection and projection earlier to have fewer (or smaller) "intermediate" tuples

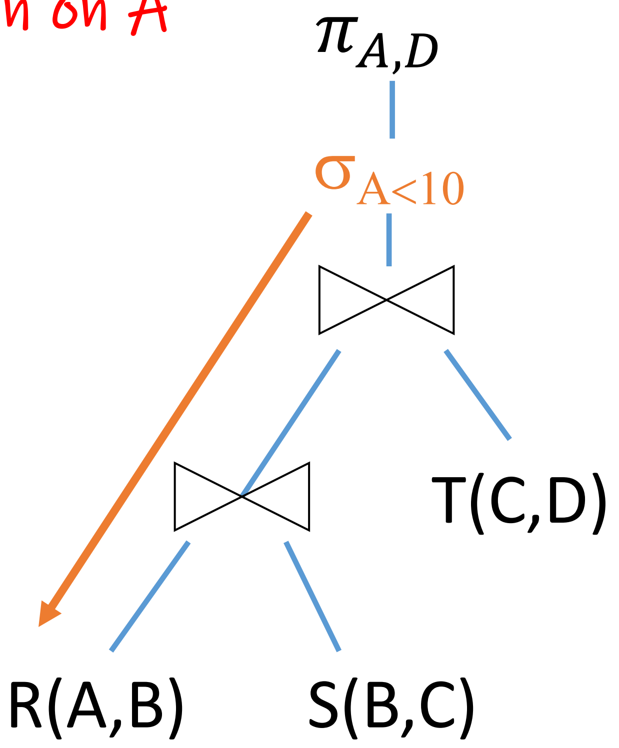
R(A,B) S(B,C) T(C,D)

```
SELECT R.A,T.D
FROM   R,S,T
WHERE  R.B = S.B
  and  S.C = T.C
  and  R.A < 10;
```

in RA

$$\pi_{A,D}\left(\sigma_{A<10}\left(T \bowtie (R \bowtie S)\right)\right)$$

1. Push down selection on A



Pushing down may be suboptimal if selection condition is very expensive (e.g. running some image processing algorithm). Projection could be unnecessary effort (but more rarely).

An example: SQL to RA to Optimized RA

Heuristic: have selection and projection earlier to have fewer (or smaller) "intermediate" tuples

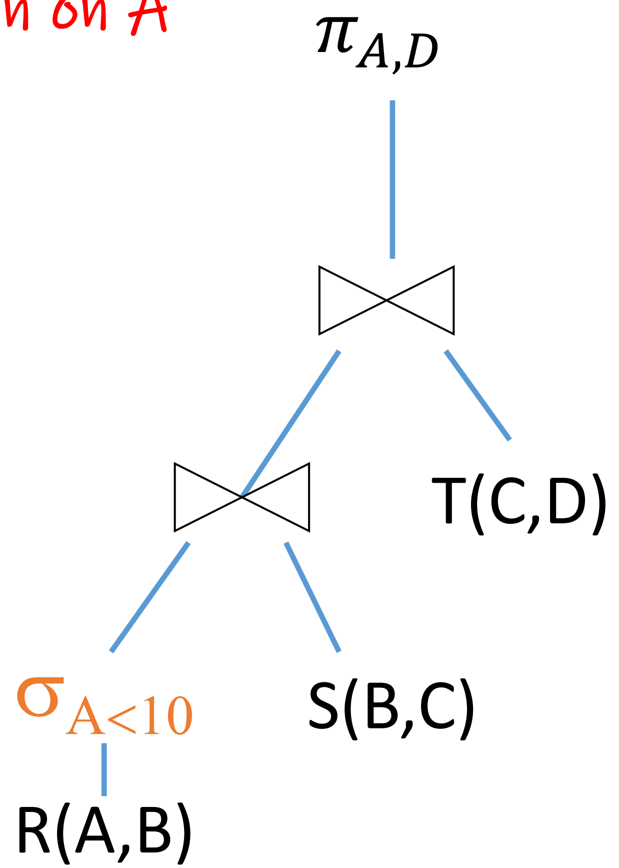
```
R(A,B) S(B,C) T(C,D)
```

```sql
SELECT R.A,T.D
FROM   R,S,T
WHERE  R.B = S.B
   and S.C = T.C
   and R.A < 10;
```

in RA

$$\pi_{A,D}\left(T \bowtie \left(\sigma_{A<10} R \bowtie S\right)\right)$$

1. Push down selection on A



An example: SQL to RA to Optimized RA

Heuristic: have selection and projection earlier to have fewer (or smaller) "intermediate" tuples

`R(A,B) S(B,C) T(C,D)`

```sql
SELECT R.A,T.D
FROM   R,S,T
WHERE  R.B = S.B
   and S.C = T.C
   and R.A < 10;
```

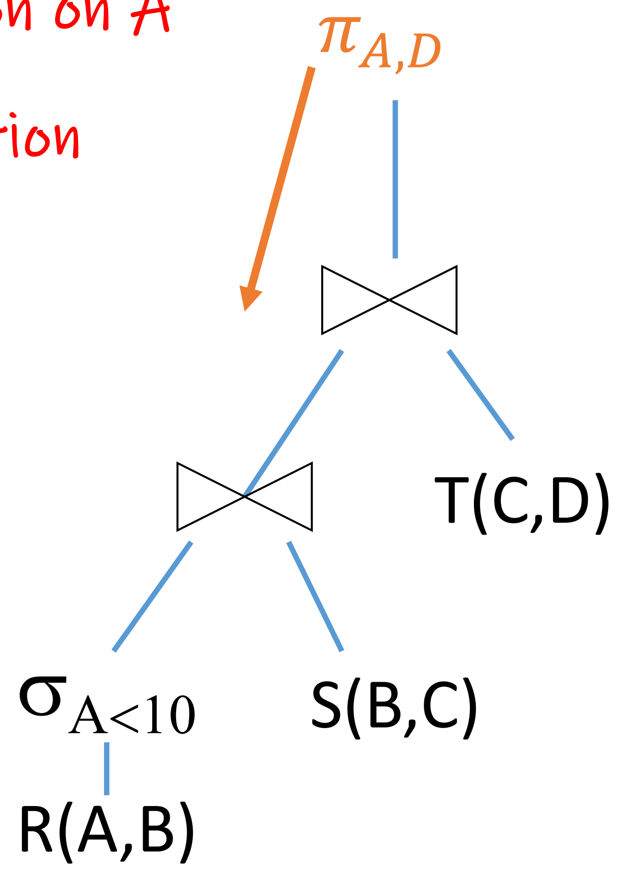
in RA

$$\pi_{A,D}\left(T \bowtie (\sigma_{A<10} R \bowtie S)\right)$$

$\pi_{-B,C}$

1. Push down selection on A
2. Push down projection

$\pi_{-B,C}$



An example: SQL to RA to Optimized RA

Variable Elimination!

```
R(A,B) S(B,C) T(C,D)
```

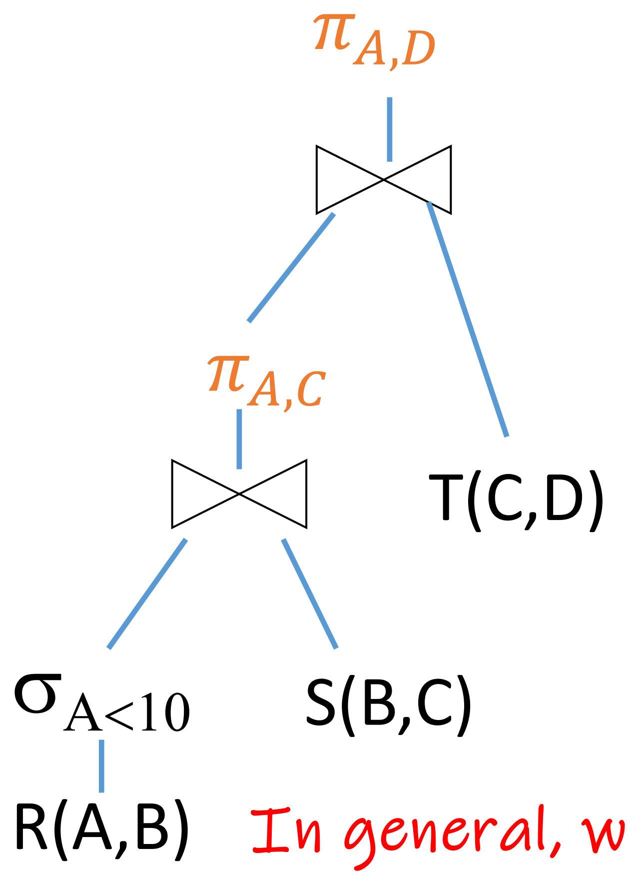
```
SELECT R.A,T.D
FROM   R,S,T
WHERE  R.B = S.B
   and S.C = T.C
   and R.A < 10;
```

in RA

$$\pi_{A,D}\left(T \bowtie \pi_{A,C}(\sigma_{A<10}R \bowtie S)\right)$$

π_{-C} π_{-B}

We now eliminate B earlier



π_{-C}

π_{-B}

In general, when is an attribute not needed?

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)
  - RA $\rightarrow$ RC
  - RC $\rightarrow$ RA

"Clear" variables*

Formula with clear variables : each quantifier "has its own variables" & each variable has only free or only bound occurrences

$$\forall x.\, \exists y.\, R(x, y, z) \wedge \neg \exists x.\, S(y, x)$$

? which variables are free or bound?

* "Clear variable" is a non-standard term used in excellent slides on logic by Marie Duzi: http://www.cs.vsb.cz/duzi/

"Clear" variables*

Formula with clear variables : each quantifier "has its own variables" & each variable has only free or only bound occurrences

$$\forall x.\, \exists y.\, [R(x, y, z)] \land \neg\exists x.\, S(y, x)$$

bound free bound

notice operator precedence: ∃ before ∧:
$\forall x. \exists y. [R(x,y,z)] \land \neg \exists x. [S(y,x)]$

Not "clear": Two x's and y's are different variables.

? how to make it "clear"

* "Clear variable" is a non-standard term used in excellent slides on logic by Marie Duzi: http://www.cs.vsb.cz/duzi/

"Clear" variables*

Formula with clear variables : each quantifier "has its own variables" & each variable has only free or only bound occurrences

$$\forall x.\,\exists y.\,R(x, y, z) \land \neg\exists x.\,S(y, x)$$

bound free bound

notice operator precedence: $\exists$ before $\land$:
$\forall x. \exists y. [R(x,y,z)] \land \neg \exists x. [S(y,x)]$

Not "clear": Two x's and y's are different variables.

$$\forall x.\,\exists y.\,R(x, y, z) \land \neg\exists u.\,S(v, u)$$

now "clear"

$$\{(z, v) \mid \forall x.\,\exists y.\,R(x, y, z) \land \neg\exists u.\,S(v, u)\}$$

Now a query. But how to make it domain-independent ?

* "Clear variable" is a non-standard term used in excellent slides on logic by Marie Duzi: http://www.cs.vsb.cz/duzi/

"Clear" variables*

Formula with clear variables : each quantifier "has its own variables" & each variable has only free or only bound occurrences

$$\forall x.\, \exists y.\, R(x, y, z) \land \neg \exists x.\, S(y, x)$$

bound free bound

notice operator precedence: ∃ before ∧:
$\forall x. \exists y. [R(x,y,z)] \land \neg \exists x. [S(y,x)]$

Not "clear": Two x's and y's are different variables.

$$\forall x.\, \exists y.\, R(x, y, z) \land \neg \exists u.\, S(v, u)$$

now "clear"

$\exists s,t. R(s,t,z) \land$ $\exists p. S(p,v) \land$

$$\{(z, v) \mid \forall x.\, \exists y.\, R(x, y, z) \land \neg \exists u.\, S(v, u)\}$$

Now a query. But how to make it domain-independent

* "Clear variable" is a non-standard term used in excellent slides on logic by Marie Duzi: http://www.cs.vsb.cz/duzi/

"Clear" variables*

Formula with clear variables : each quantifier "has its own variables" & each variable has only free or only bound occurrences

$$\forall x.\, \exists y.\, R(x, y, z) \land \neg\exists x.\, S(y, x)$$

bound free bound

notice operator precedence: ∃ before ∧:
$\forall x.\exists y.[R(x,y,z)] \land \neg\exists x.[S(y,x)]$

Not "clear": Two x's and y's are different variables.

$$\forall x.\, \exists y.\, R(x, y, z) \land \neg\exists u.\, S(v, u)$$

now "clear"

$\exists s,t.R(s,t,z) \land$ $\exists p.S(p,v) \land$

$$\{(z, v) \mid \cancel{\forall x.\, \exists y.\, R(x, y, z) \land} \neg\exists u.\, S(v, u)\}$$

$\forall x.[\exists w,t.R(x,w,t) \rightarrow \exists y.R(x,y,z)]$

Now a query. But how to make it domain-independent

* "Clear variable" is a non-standard term used in excellent slides on logic by Marie Duzi: http://www.cs.vsb.cz/duzi/

Repeated variable names

In sentences with multiple quantifiers, distinct variables do not need to range over distinct objects! (cp. homomorphism vs. isomorphism)

? Which of the following formulas imply each other?

$\forall x.\forall y.\, E(x,y)$ $\qquad$ $\forall x.\, E(x,x)$

$\exists x.\exists y.\, E(x,y)$ $\qquad$ $\exists x.\, E(x,x)$

Repeated variable names

In sentences with multiple quantifiers, distinct variables do not need to range over distinct objects! (cp. homomorphism vs. isomorphism)

Assume DOM = {1, 2}:

E

s	t
1	1
1	2
2	1
2	2

$\forall x. \forall y.\ E(x,y) \quad \Rightarrow \quad \forall x.\ E(x,x)$

$\exists x. \exists y.\ E(x,y) \quad \Leftarrow \quad \exists x.\ E(x,x)$

E

s	t
1	2

Repeated variable names

In sentences with multiple quantifiers, distinct variables do not need to range over distinct objects! (cp. homomorphism vs. isomorphism)

Assume DOM = Ø :

$\forall x. \forall y.\, E(x,y) \quad \Rightarrow \quad \forall x.\, E(x,x)$

$\Downarrow$ Only if domain is not empty! Dom ≠ Ø $\Downarrow$

$\exists x. \exists y.\, E(x,y) \quad \Leftarrow \quad \exists x.\, E(x,x)$

E

s	t

Example RC→RA

Person(id, name, country)
Spouse(id1, id2)

Q: ?

In DRC:

$$\{\, x \mid \exists z,w.\ \mathrm{Person}(x,z,w) \land \forall y.[\neg \mathrm{Spouse}(x,y)] \,\}$$

Example RC→RA

Person(id, name, country)
Spouse(id1, id2)

Q: "Find persons without a spouse"

In DRC:

$$\{\, x \mid \exists z,w.\ \mathrm{Person}(x,z,w) \land \forall y.[\neg \mathrm{Spouse}(x,y)] \,\}$$

In RA:

?

Example RC→RA

Person(id, name, country)
Spouse(id1, id2)

In DRC:

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

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

In RA:

$$\pi_{id}\text{Person} - \pi_{id1}\text{Spouse}$$

$$\pi_{id}\text{Person} - \rho_{id1\rightarrow id}\,(\pi_{id1}\text{Spouse})$$

Recall: named vs ordered perspective

Example RA→RC for $R(X,Y) \div S(Y)$

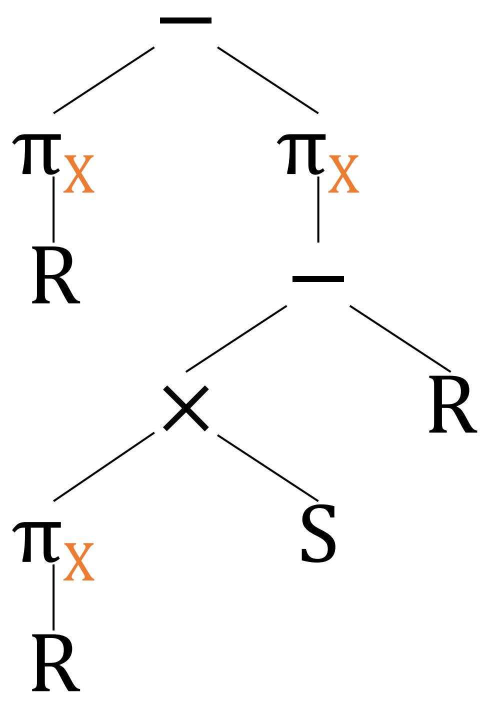
In DRC:

$$\{\ x \mid \exists z.[R(x,z)] \land \quad \forall y.[S(y) \ \rightarrow \ R(x,y)]\ \}$$

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

In RA:

?



Example RA→RC for $R(X,Y) \div S(Y)$

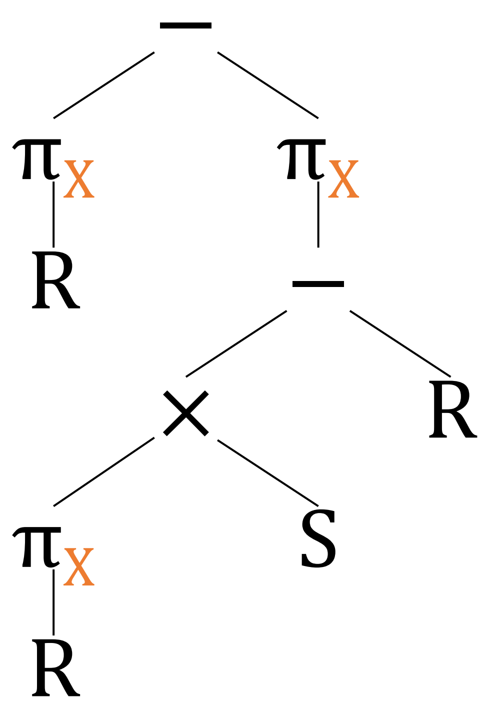
In DRC:

$$\{\, x \mid \exists z.[R(x,z)] \wedge \quad \forall y.[S(y) \;\rightarrow\; R(x,y)] \,\}$$

$$\{\, x \mid \exists z.[R(x,z)] \wedge \neg\exists y.[S(y) \wedge \neg R(x,y)] \,\}$$

In RA:

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



Translation back into DRC:

?

Example RA→RC for $R(X,Y) \div S(Y)$

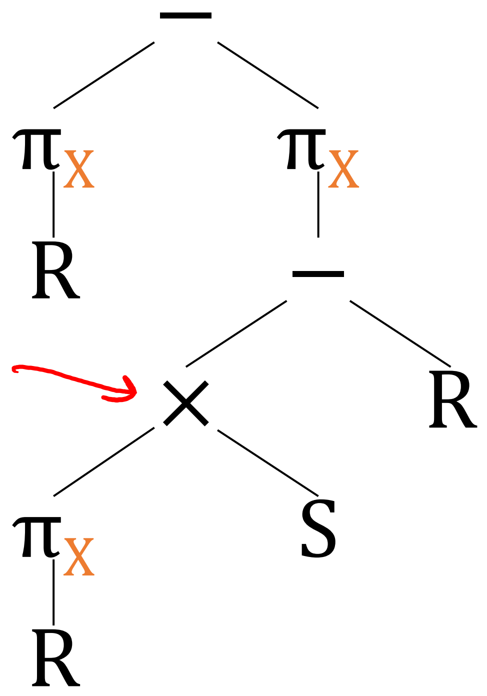
In DRC:

$$\{\, x \mid \exists z.[R(x,z)] \wedge \forall y.[S(y) \rightarrow R(x,y)] \,\}$$

$$\{\, x \mid \exists z.[R(x,z)] \wedge \neg\exists y.[S(y) \wedge \neg R(x,y)] \,\}$$

In RA:

$$\pi_X R - \pi_X((\underbrace{\pi_X R \times S}_{0}) - R)$$



Translation back into DRC:

$$\{\, x \mid \exists z.[R(x,z)] \wedge \neg\exists y.[\exists z.[R(x,z)] \wedge S(y) \wedge \neg R(x,y)] \,\}$$

Equivalence Between RA and Domain-Independent RC

> CODD'S THEOREM:
> RA and **domain-independent** RC
> have the same expressive power.

More formally, on every schema $\mathbf{S}$:

1. For every RA expression E, there is a domain-independent RC query Q s.t. $\mathrm{Q} \equiv \mathrm{E}$
2. For every domain-independent RC query Q, there is an RA expression E s.t. $\mathrm{Q} \equiv \mathrm{E}$

The proof has two directions:

RA→RC:
by induction on the size of the RA expression

RC→RA:
more involved

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)
  - RA $\rightarrow$ RC
  - RC $\rightarrow$ RA

RA → DRC: Intuition

Intuition: $\{x \mid \exists y.[R(x,y)] \wedge \exists y.[S(x,y)]\}$

contrast with: $\{x \mid \exists y.[R(x,y)] \wedge \exists z.[S(x,z)]\}$

- Construction by induction
- Key technical detail: need to maintain a mapping b/w attribute names and variables

$Q(1) \leftarrow \underset{y=2}{R(1,2)}, \underset{y=3}{S(1,3)}$

RA expression	DRC formula ϕ Here, ϕ_i is the formula constructed for expression E_i
R (n columns)	$R(X_1,\ldots,X_n)$
$E_1 \times E_2$	
$E_1 \cup E_2$	
$E_1 - E_2$	
$\pi_{A_1,\ldots,A_k}(E_1)$	
$\sigma_c(E_1)$	

RA → DRC: Intuition

Intuition: $\{x \mid \exists y.[R(x,y)] \wedge \exists y.[S(x,y)]\}$

contrast with: $\{x \mid \exists y.[R(x,y)] \wedge \exists z.[S(x,z)]\}$

- Construction by induction
- Key technical detail: need to maintain a mapping b/w attribute names and variables

RA expression	DRC formula ϕ Here, ϕ_i is the formula constructed for expression E_i
R (n columns)	$R(X_1,\ldots,X_n)$
$E_1 \times E_2$	$\phi_1 \wedge \phi_2$ disjoint variables (rename)
$E_1 \cup E_2$	$\phi_1 \vee \phi_2$ use identical variables (rename) UNION COMPATIBLE
$E_1 - E_2$	$\phi_1 \wedge \neg\phi_2$ use identical variables (rename)
$\pi_{A_1,\ldots,A_k}(E_1)$	
$\sigma_c(E_1)$	

RA → DRC: Intuition

- Construction by induction
- Key technical detail: need to maintain a mapping b/w attribute names and variables

Intuition: $\{x \mid \exists y.[R(x,y)] \wedge \exists y.[S(x,y)]\}$

contrast with: $\{x \mid \exists y.[R(x,y)] \wedge \exists z.[S(x,z)]\}$

RA expression	DRC formula ϕ Here, ϕ_i is the formula constructed for expression E_i
R (n columns)	$R(X_1,\ldots,X_n)$
$E_1 \times E_2$	$\phi_1 \wedge \phi_2$ disjoint variables (rename)
$E_1 \cup E_2$	$\phi_1 \vee \phi_2$ use identical variables (rename)
$E_1 - E_2$	$\phi_1 \wedge \neg\phi_2$ use identical variables (rename)
$\pi_{A_1,\ldots,A_k}(E_1)$	$\exists X_1 \ldots \exists X_m . \phi_1$ where $X_1, \ldots, X_m$ are the variables not among $A_1, \ldots, A_k$
$\sigma_c(E_1)$	$\phi_1 \wedge c$

Correspondence more natural with project-away operator: $\pi_{-A_1,\ldots,A_m}(E_1)$

RA → DRC: Example $R \div S$

$R(A,B) \quad S(B)$

RA	DRC	Mapping
R		
$\pi_A(R)$		
S		
$\pi_A(R) \times S$		
$(\pi_A(R) \times S) - R$		
$\pi_A\big((\pi_A(R) \times S) - R\big)$		
$\pi_A(R) - \pi_A\big((\pi_A(R) \times S) - R\big)$		

RA → DRC: Example R÷S

R(A,B) S(B)

RA	DRC	Mapping
R	$R(x, z)$	x:R.A, z:R.B
$\pi_A(R)$	$\exists z. R(x, z)$	x:R.A
S	$S(y)$	y:S.B
$\pi_A(R) \times S$		
$(\pi_A(R) \times S) - R$		
$\pi_A\big((\pi_A(R) \times S) - R\big)$		
$\pi_A(R) - \pi_A\big((\pi_A(R) \times S) - R\big)$		

RA → DRC: Example R÷S

R(A,B)   S(B)

RA	DRC	Mapping
R	$R(x, z)$	x:R.A, z:R.B
$\pi_A(R)$	$\exists z.\, R(x, z)$	x:R.A
S	$S(y)$	y:S.B
$\pi_A(R) \times S$	$\exists z.\, \boxed{R(x, z)} \wedge S(y)$ y needs to be different from z	x:R.A, y:S.B
$(\pi_A(R) \times S) - R$	$\exists z.\, R(x, z) \wedge S(y) \wedge \neg R(x, y)$	x:R.A, y:S.B
$\pi_A\big((\pi_A(R) \times S) - R\big)$	$\exists y.\, [\exists z.\, R(x, z) \wedge S(y) \wedge \neg R(x, y)\,]$	x:R.A
$\pi_A(R) -$ $\pi_A\big((\pi_A(R) \times S) - R\big)$	$\exists z.\, R(x, z) \wedge$ x's need to be same variable $\neg \exists y.\, [\exists z.\, R(x, z) \wedge S(y) \wedge \neg R(x, y)\,]$	x:R.A

y's don't need to be same variable

RA → DRC: Example R÷S

$R(A,B) \quad S(B)$

This is the DRC expression we got by translating from RA:

$$\{\, x \mid \exists z.[R(x,z)] \wedge \neg \exists y.[\exists z.[R(x,z)] \wedge S(y) \wedge \neg R(x,y)]\}$$

This is the DRC expression for relational division that we saw earlier.

$$\{\, x \mid \exists z.[R(x,z)] \wedge \neg \exists y.[\qquad S(y) \wedge \neg R(x,y)]\}$$

Claim: there is no logically equivalent RA expression that uses the table R only twice.
For details see: "*On the Reasonable Effectiveness of Relational Diagrams: Explaining Relational Query Patterns and the Pattern Expressiveness of Relational Languages*", SIGMOD'24. https://arxiv.org/pdf/2401.04758

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)
  - RA $\rightarrow$ RC
  - RC $\rightarrow$ RA

DRC → RA: Intuition

Proof (Sketch):

- Show first that for every relational database schema **S**, there is a relational algebra expression **E** such that for every database instance **D**, we have that ADom(**D**) = **E**(**D**).
  - Tip: just the union of all columns
- Use the above fact and induction on the construction of RC formulas to obtain a translation of RC under the active domain interpretation to RA.

DRC → RA: Intuition

$E(A,B)$

- In this translation, the most interesting part is the simulation of the universal quantifier ∀ in relational algebra

  uses the logical equivalence: $\forall y.\, \phi \equiv$?

Based on Phokion Kolaitis' "Logic and Databases" series at Simons Institute, 2016. https://simons.berkeley.edu/talks/logic-and-databases
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

DRC → RA: Intuition

$E(A,B)$

- In this translation, the most interesting part is the simulation of the universal quantifier ∀ in relational algebra

  uses the logical equivalence: $\forall y. \phi \equiv \neg \exists y. \neg \phi$

- As an illustration, consider: $\forall y. E(x, y) \equiv$?

DRC → RA: Intuition

$E(A,B)$

- In this translation, the most interesting part is the simulation of the universal quantifier ∀ in relational algebra

  uses the logical equivalence: $\forall y.\, \phi \equiv \quad \neg\exists y.\, \neg\phi$

- As an illustration, consider: $\forall y.\, E(x, y) \equiv \neg\exists y.\, \neg E(x, y)$

  and recall: $ADom(D) = \quad ?$

DRC → RA: Intuition

$E(A,B)$

- In this translation, the most interesting part is the simulation of the universal quantifier ∀ in relational algebra

  uses the logical equivalence: $\forall y. \phi \equiv \neg \exists y. \neg \phi$

- As an illustration, consider: $\forall y. E(x,y) \equiv \neg \exists y. \neg E(x,y)$

  and recall: $\boxed{ADom(D) = \pi_A(E) \cup \pi_B(E)}$

DRC formula ϕ	RA expression for ϕ^{adom}
$\neg E(x,y)$	
$\exists y. \neg E(x,y)$	?
$\neg \exists y. \neg E(x,y)$	

DRC → RA: Intuition

$E(A,B)$

- In this translation, the most interesting part is the simulation of the universal quantifier ∀ in relational algebra

  uses the logical equivalence: $\forall y.\, \phi \equiv \neg \exists y.\, \neg \phi$

- As an illustration, consider: $\forall y.\, E(x, y) \equiv \neg \exists y.\, \neg E(x, y)$

  and recall: $\boxed{\text{ADom}(D) = \pi_A(E) \cup \pi_B(E)}$

DRC formula ϕ	RA expression for ϕ^{adom}
$\neg E(x, y)$	$\rho_A(\text{ADom}(D)) \times \rho_B(\text{ADom}(D)) - E$
$\exists y.\, \neg E(x, y)$	π_{-B} $\pi_A[\rho_A(\text{ADom}(D)) \times \rho_B(\text{ADom}(D)) - E]$
$\neg \exists y.\, \neg E(x, y)$	$\rho_A(\text{ADom}(D)) - \pi_A[\rho_A(\text{ADom}(D)) \times \rho_B(\text{ADom}(D)) - E]$

Entire Story in One Slide (repeated slide)

1. RC = FOL over DB
2. RC can express "bad queries" that depend not only on the DB, but also on the domain from which values are taken (domain dependence)
3. We cannot test whether an RC query is "good," but we can use a "good" subset of RC that captures all "good" queries (safety)
4. "Good" RC and RA can express the same queries! (equivalence = Codd's theorem)

Discussion

- What is the monotone fragment of RA ?

?

- What are the safe queries in RA ?

?

- Where do we use RA (applications) ?

?

Discussion

- What is the monotone fragment of RA ?
  - Basic except difference (−): $\cup$, σ, π, $\bowtie$

- What are the safe queries in RA ?

?

- Where do we use RA (applications) ?

?

Source: Dan Suciu, CSE 554, 2011.

Discussion

- What is the monotone fragment of RA ?
  - Basic except difference (−): ∪, σ, π, ⋈
- What are the safe queries in RA ?
  - All RA queries are safe
- Where do we use RA (applications) ?

?

$A(x) - B(x)$

$\neg B(x)$

Source: Dan Suciu, CSE 554, 2011.

Discussion

- What is the monotone fragment of RA ?
  - Basic except difference ($-$): $\cup$, σ, π, $\bowtie$
- What are the safe queries in RA ?
  - All RA queries are safe
- Where do we use RA (applications) ?
  - Translating SQL (from WHAT to HOW)
  - Directly as query languages (e.g. Pig-Latin)

See next pages

Source: Dan Suciu, CSE 554, 2011.

Example 1. *Suppose we have a table* **urls: (url, category, pagerank)**. *The following is a simple SQL query that finds, for each sufficiently large category, the average pagerank of high-pagerank urls in that category.*

```
SELECT category, AVG(pagerank)
FROM urls WHERE pagerank > 0.2
GROUP BY category HAVING COUNT(*) > 10^6
```

An equivalent Pig Latin program is the following. (Pig Latin is described in detail in Section 3; a detailed understanding of the language is not required to follow this example.)

```
good_urls = FILTER urls BY pagerank > 0.2;
groups = GROUP good_urls BY category;
big_groups = FILTER groups BY COUNT(good_urls)>10^6;
output = FOREACH big_groups GENERATE
            category, AVG(good_urls.pagerank);
```

Example 1. *Suppose we have a table* **urls: (url, category, pagerank)**. *The following is a simple SQL query that finds, for each sufficiently large category, the average pagerank of high-pagerank urls in that category.*

```
SELECT category, AVG(pagerank)
FROM urls WHERE pagerank > 0.2
GROUP BY category HAVING COUNT(*) > 10^6
```

An equivalent Pig Latin program is the following. (Pig Latin is described in detail in Section 3; a detailed understanding of the language is not required to follow this example.)

```
good_urls = FILTER urls BY pagerank > 0.2;
groups = GROUP good_urls BY category;
big_groups = FILTER groups BY COUNT(good_urls)>10^6;
output = FOREACH big_groups GENERATE
            category, AVG(good_urls.pagerank);
```

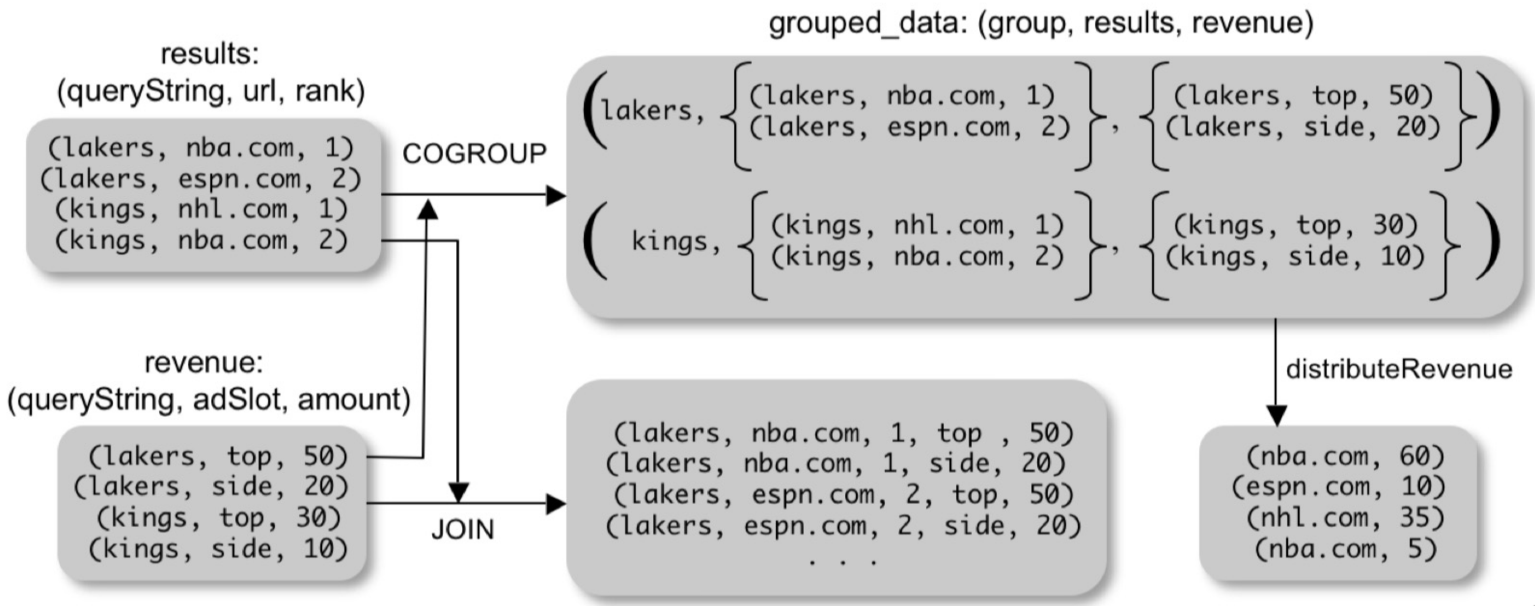


Figure 2: COGROUP versus JOIN.

```
grouped_data  =  COGROUP results BY queryString,
                         revenue BY queryString;
```

For more, see:
https://pig.apache.org/docs/r0.17.0/basic.html

3.5.2 JOIN *in Pig Latin*

Not all users need the flexibility offered by COGROUP. In many cases, all that is required is a regular equi-join. Thus, Pig Latin provides a JOIN keyword for equi-joins. For example,

```
join_result  =  JOIN results BY queryString,
                revenue BY queryString;
```

It is easy to verify that JOIN is only a syntactic shortcut for COGROUP followed by flattening. The above join command is equivalent to:

```
   temp_var  =  COGROUP results BY queryString,
                revenue BY queryString;
join_result  =  FOREACH temp_var GENERATE
                FLATTEN(results), FLATTEN(revenue);
```