Relational Algebra

Algebra

A mathematical system consisting of
Operands: variables or values from which new values are constructed
Operators: symbols denoting procedures that construct new values given existing values

Relational Algebra

Operands are relations
Operators take one or two relation instances as arguments and return one relation instance as result

Properties

Closed: Relations in; Relations Out
Typed: the schema of the input relations determines the output schema; statically check if certain operations are legal

Relational Algebra

Unary Operations
- Projection π
- Selection σ
- Renaming ρ

Binary Operations
- Union $\cup$
- Difference $-$
- Cartesian Product $\times$

Derived Operations
- Join $\bowtie$
- Intersection $\cap$
- Division $\div$

Extensions
- Duplicate elimination δ
- Group By γ
- Aggregation (min, max, count, sum ...)
- Sort τ

Relational Operators

Unary Operations

Projection π

$\pi_{\{A_1,\ldots,A_n\}}(R)$

Selects Columns

aid	name	species	age	feedtime
325	Happy	monkey	1	8:30
327	Grumpy	monkey	1	9:00
678	Squeaky	dolphin	6	10:30
874	Angry	hen	4	5:30
921	Moma	orangutan	10	8:40

$\pi_{\{\text{species,age}\}}(\text{Animals})$

species	age
monkey	1
dolphin	6
hen	4
orangutan	10

Generalized Projection π

$\pi_{\{E_1,\ldots,\}}(R)$

Extends the projection operation with arithmetic expressions

aid	name	species	age	feedtime
325	Happy	monkey	1	8:30
327	Grumpy	monkey	1	9:00
678	Squeaky	dolphin	6	10:30
874	Angry	hen	4	5:30
921	Moma	orangutan	10	8:40

$\pi_{\{\text{species},\text{age}*12\}}(\text{Animals})$

species	age
monkey	12
dolphin	72
hen	48
orangutan	120

Selection σ

$$\sigma_{\{c\}}(R)$$

Returns all tuples/rows that satisfy a condition, *c*

c: Boolean combination (∧,∨) of terms
Term: attribute op constant or
attribute op attribute
op: $<, \leq, =, \neq, \geq, >$

aid	name	species	age	feedtime
325	Happy	monkey	1	8:30
327	Grumpy	monkey	1	9:00
678	Squeaky	dolphin	6	10:30
874	Angry	hen	4	5:30
921	Moma	orangutan	10	8:40

$$\sigma_{\{age>2\}}(\text{Animals})$$

aid	name	species	age	feedtime
678	Squeaky	dolphin	6	10:30
874	Angry	hen	4	5:30
921	Moma	orangutan	10	8:40

Selection σ

$\sigma_{\{c\}}(R)$

Returns all tuples/rows that satisfy a condition, *c*

c: Boolean combination (∧,∨) of terms
Term: attribute op constant or
attribute op attribute
op: $<, \leq, =, \neq, \geq, >$

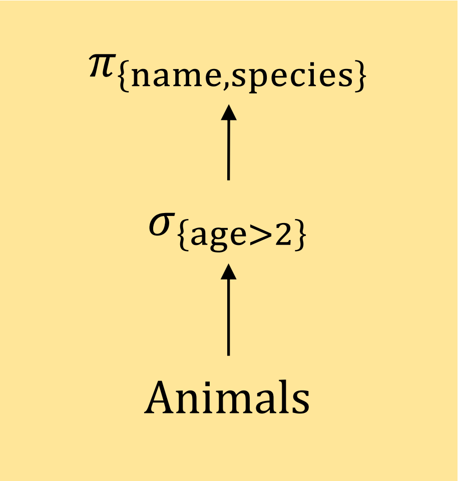
kid	name	address
007	Azza	A2-177
123	Batu	UnixLab
555	Miro	A1-1102G
562	Hazem	A2-186

$\sigma_{\{\text{name}=\text{“Azza”}\}}(\text{Keepers})$

kid	name	address
007	Azza	A2-177

Composition

$$\pi(\sigma(R))$$



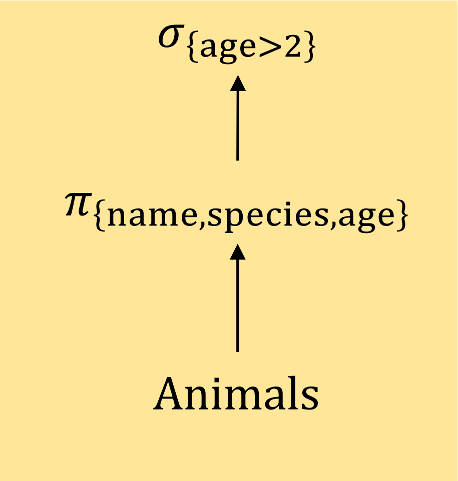
aid	name	species	age	feedtime
325	Happy	monkey	1	8:30
327	Grumpy	monkey	1	9:00
678	Squeaky	dolphin	6	10:30
874	Angry	hen	4	5:30
921	Moma	orangutan	10	8:40

$$\pi_{\{\text{name,species}\}}\left(\sigma_{\{\text{age}>2\}}(\text{Animals})\right)$$

name	species
Squeaky	dolphin
Angry	hen
Moma	orangutan

Composition

$$\sigma(\pi(R))$$



aid	name	species	age	feedtime
325	Happy	monkey	1	8:30
327	Grumpy	monkey	1	9:00
678	Squeaky	dolphin	6	10:30
874	Angry	hen	4	5:30
921	Moma	orangutan	10	8:40

$$\sigma_{\{age>2\}}(\pi_{\{name,species,age\}}(\text{Animals}))$$

name	species	age
Squeaky	dolphin	6
Angry	hen	4
Moma	orangutan	10

Renaming ρ

$$\rho_{\{R'(\ldots, i \to A'_i, \ldots)\}}(R)$$

Renames relation R to R' and a rename list of columns from position to new attribute name

aid	name	species	age	feedtime
325	Happy	monkey	1	8:30
327	Grumpy	monkey	1	9:00
678	Squeaky	dolphin	6	10:30
874	Angry	hen	4	5:30
921	Moma	orangutan	10	8:40

$$\rho_{\{\text{Infants }(2 \to \text{months})\}}(\pi_{\{\text{name}, \text{age}*12\}}(\sigma_{\{\text{age} \leq 2\}}(\text{Animals})))$$

name	months
Happy	12
Grumpy	12

Binary Operations

Adults

aid	name	species	age	feedtime
678	Squeaky	dolphin	6	10:30
874	Angry	hen	4	5:30
921	Moma	orangutan	10	8:40

Apes

aid	name	species	age	feedtime
325	Happy	monkey	1	8:30
327	Grumpy	monkey	1	9:00
921	Moma	orangutan	10	8:40

Union ∪

$R \cup S$

Adult ∪ Apes

aid	name	species	age	feedtime
325	Happy	monkey	1	8:30
327	Grumpy	monkey	1	9:00
678	Squeaky	dolphin	6	10:30
874	Angry	hen	4	5:30
921	Moma	orangutan	10	8:40

Combines two relations that are *compatible*:

- Same number of fields
- Same types

**UnionAll is a special union for bags that keeps duplicates.*

Difference –

$R - S$

Subtracts one relation from another *compatible* relation:

- Same number of fields
- Same types

Adults

aid	name	species	age	feedtime
678	Squeaky	dolphin	6	10:30
874	Angry	hen	4	5:30
921	Moma	orangutan	10	8:40

Apes

aid	name	species	age	feedtime
325	Happy	monkey	1	8:30
327	Grumpy	monkey	1	9:00
921	Moma	orangutan	10	8:40

Adults – Apes

aid	name	species	age	feedtime
678	Squeaky	dolphin	6	10:30
874	Angry	hen	4	5:30

Apes – Adults

aid	name	species	age	feedtime
325	Happy	monkey	1	8:30
327	Grumpy	monkey	1	9:00

**ExceptAll is special difference operation in SQL for bags where a row appears in difference A – B as many times as it appears in A, minus the number of times it appears in B, but never less than 0 times*

Cartesian Product ×

$$R \times S$$

Each row in R is paired with each row in S to produce |R|*|S| rows.

Rarely used in practice; mainly used to express joins

Enclosures

eid	room	building
72	Gym	C2
89	Pool	C2

Adults

aid	name	species	age	feedtime
678	Squeaky	dolphin	6	10:30
874	Angry	hen	4	5:30
921	Moma	orangutan	10	8:40

$$Enclosures \times Adults$$

eid	room	building	aid	name	species	age	feedtime
72	Gym	C2	325	Happy	monkey	1	8:30
72	Gym	C2	327	Grumpy	monkey	1	9:00
72	Gym	C2	678	Squeaky	dolphin	6	10:30
72	Gym	C2	874	Angry	hen	4	5:30
72	Gym	C2	921	Moma	orangutan	10	8:40
89	Pool	C2	325	Happy	monkey	1	8:30
89	Pool	C2	327	Grumpy	monkey	1	9:00
89	Pool	C2	678	Squeaky	dolphin	6	10:30
89	Pool	C2	874	Angry	hen	4	5:30
89	Pool	C2	921	Moma	orangutan	10	8:40

Derived Set Operations

Adults

aid	name	species	age	feedtime
678	Squeaky	dolphin	6	10:30
874	Angry	hen	4	5:30
921	Moma	orangutan	10	8:40

Apes

aid	eid	species	age	feedtime
325	Happy	monkey	1	8:30
327	Grumpy	monkey	1	9:00
921	Moma	orangutan	10	8:40

Intersection ∩

$$R \cap S$$

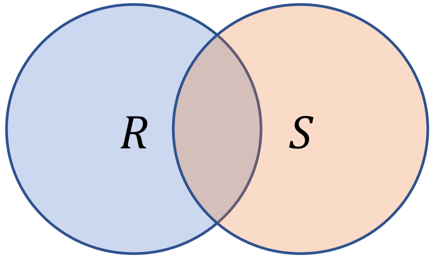
Intersects two relations that are *compatible*:

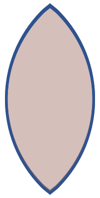
- Same number of fields
- Same types

Adult ∩ Apes

aid	name	species	age	feedtime
921	Moma	orangutan	10	8:40

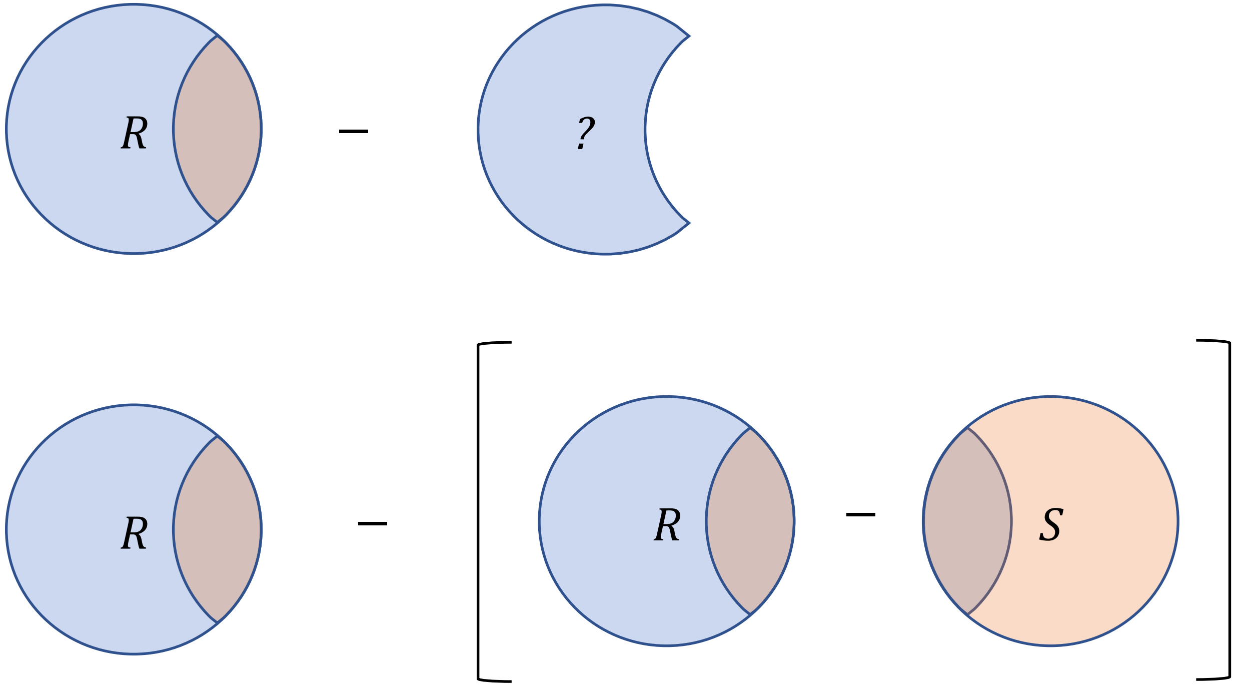
** An element appears in the intersection of two bags the minimum of the number of times it appears in either.*





$R \cap S$

Deriving Intersection ∩



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

Division ÷

$R \div S$

Select rows in R and project attribute names unique to R, such that all the rows in S are present in R

Eats

species	food
chimp	grain
chimp	insects
chimp	fruit
chimp	meat
hen	grain
hen	insects
hyena	meat
hyena	insects

A

food
insects

Eats ÷ A

species
chimp
hen
hyena

B

food
grain
insects

Eats ÷ B

species
chimp
hen

C

food
grain
meat
insects

Eats ÷ C

species
chimp

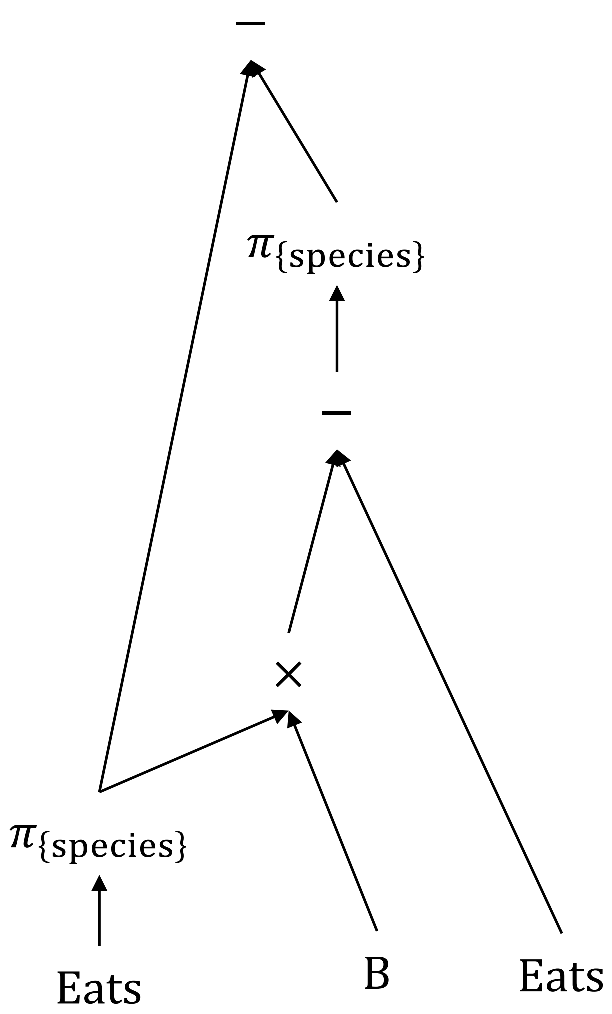
Evaluating Division ÷

$\mathbb{A}_R, \mathbb{A}_S$ are the set of attribute names in R and S respectively, then

$$R \div S$$
$$= \pi_{\{\mathbb{A}_R - \mathbb{A}_S\}} R$$
$$- \pi_{\{\mathbb{A}_R - \mathbb{A}_S\}}[\pi_{\{\mathbb{A}_R - \mathbb{A}_S\}} R \times S \ - R]$$

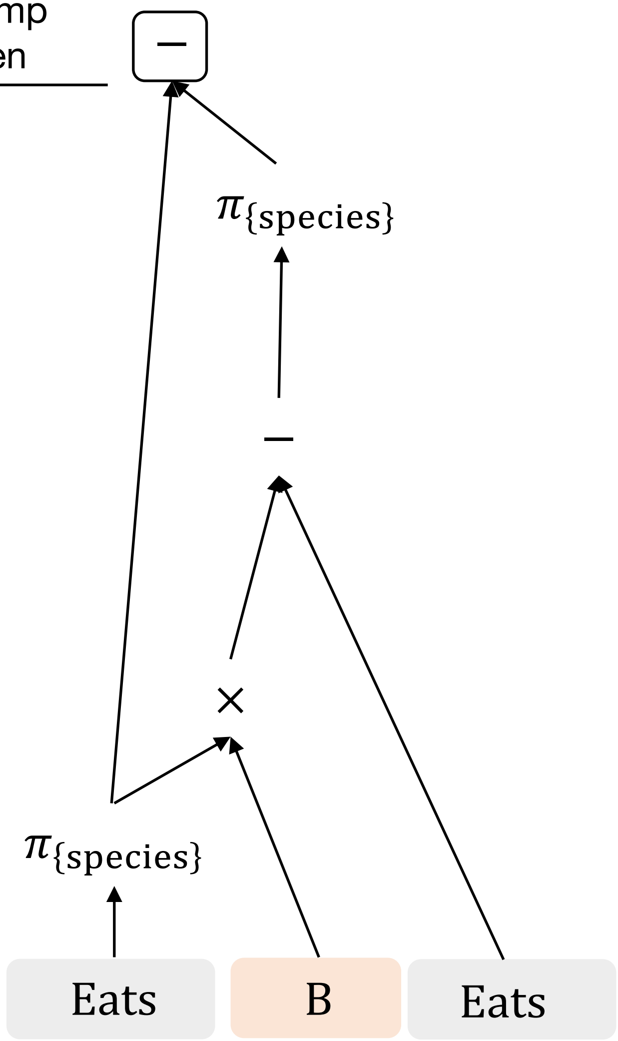
$$\mathbb{A}_{\text{Eats}} - \mathbb{A}_B = \{\text{species}, \text{food}\} - \{\text{food}\} = \{\text{species}\}$$

$$\text{Eats} \div \text{B}$$
$$= \pi_{\{\text{species}\}}\text{Eats}$$
$$- \pi_{\{\text{species}\}}[\pi_{\{\text{species}\}}\text{Eats} \times B \ - \text{Eats}]$$



Eats ÷ B

species
chimp
hen



Eats

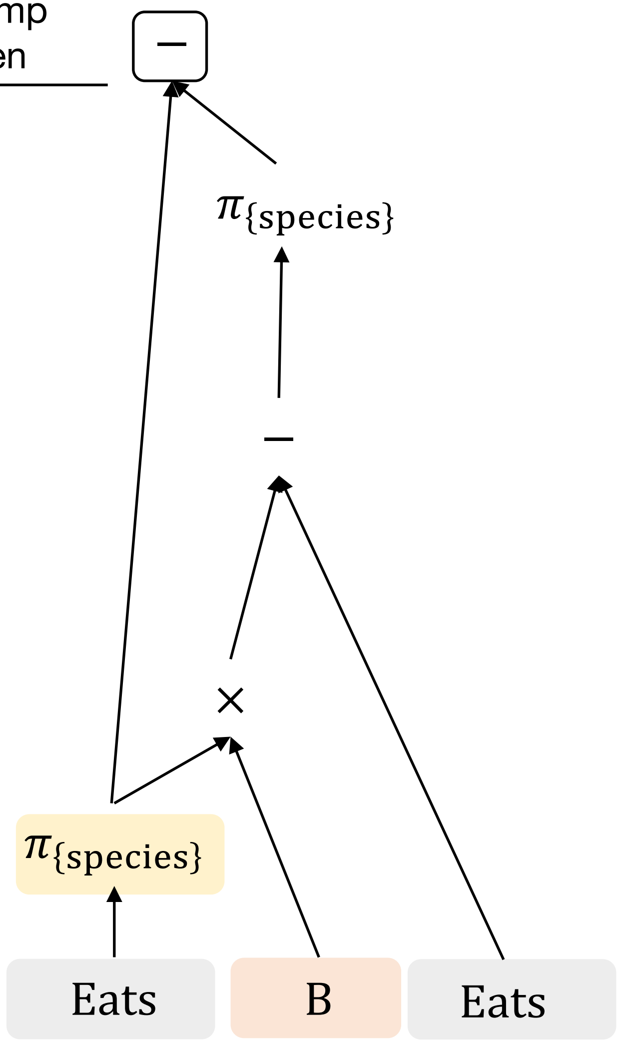
species	food
chimp	grain
chimp	insects
chimp	fruit
chimp	meat
hen	grain
hen	insects
hyena	meat
hyena	insects

B

food
grain
insects

Eats ÷ B

species
chimp
hen



Eats

species	food
chimp	grain
chimp	insects
chimp	fruit
chimp	meat
hen	grain
hen	insects
hyena	meat
hyena	insects

$\pi_{\{species\}}$ Eats

species
chimp
hen
hyena

B

food
grain
insects

Eats ÷ B

species
chimp
hen

$\pi_{\{species\}}$ Eats × B

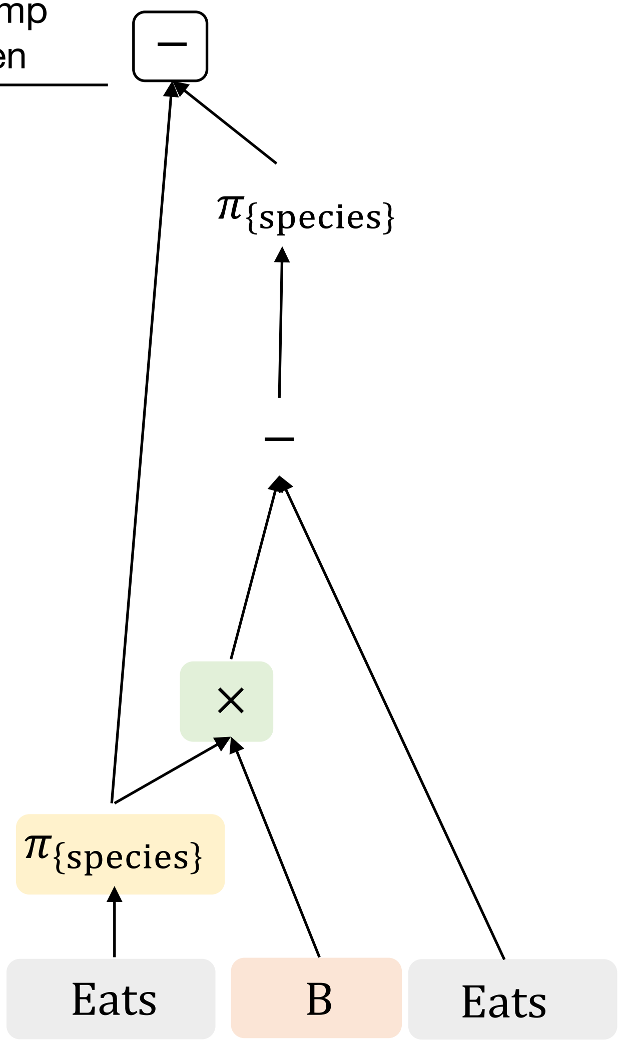
species	food
chimp	grain
chimp	insects
hen	grain
hen	insects
hyena	grain
hyena	insects

Eats

species	food
chimp	grain
chimp	insects
chimp	fruit
chimp	meat
hen	grain
hen	insects
hyena	meat
hyena	insects

$\pi_{\{species\}}$ Eats

species
chimp
hen
hyena

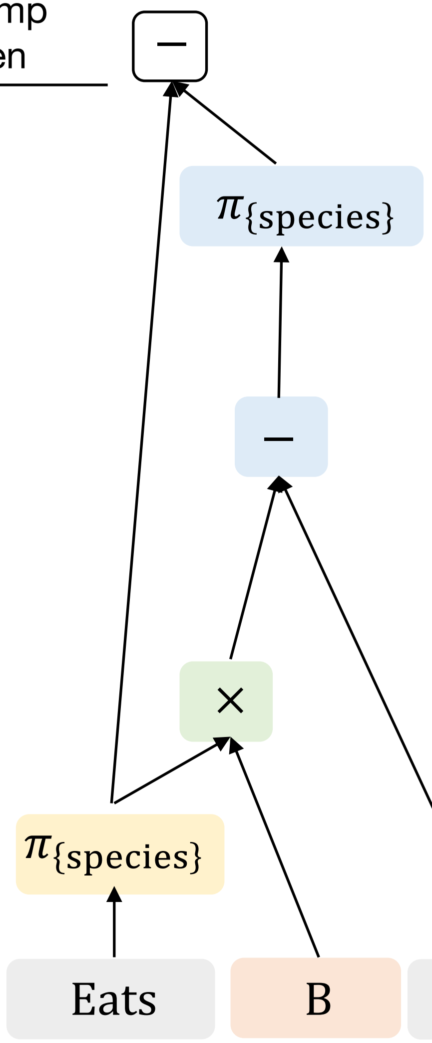


B

food
grain
insects

Eats ÷ B

species
chimp
hen



$\pi_{\{species\}}(\pi_{\{species\}}$ Eats × B – Eats)

species
hyena

$\pi_{\{species\}}$ Eats × B

species	food
chimp	grain
chimp	insects
hen	grain
hen	insects
hyena	grain
hyena	insects

$\pi_{\{species\}}$ Eats × B - Eats

species	food
hyena	grain

Eats

species	food
chimp	grain
chimp	insects
chimp	fruit
chimp	meat
hen	grain
hen	insects
hyena	meat
hyena	insects

$\pi_{\{species\}}$ Eats

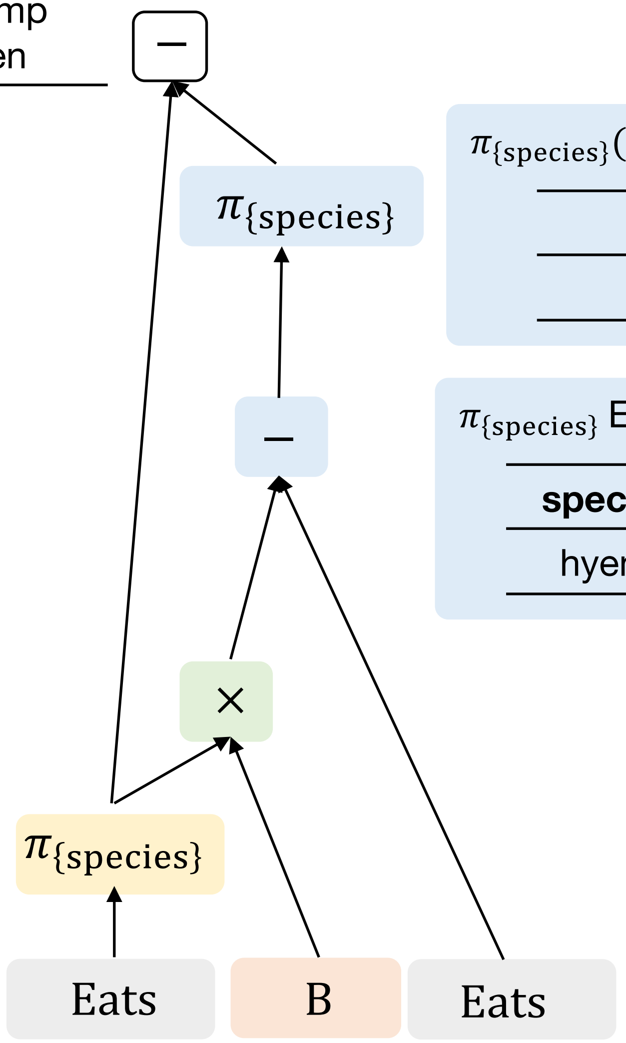
species
chimp
hen
hyena

B

food
grain
insects

Eats ÷ B

species
chimp
hen



$\pi_{\{species\}}(\pi_{\{species\}}$ Eats × B – Eats)

species
hyena

$\pi_{\{species\}}$ Eats × B - Eats

species	food
hyena	grain

$\pi_{\{species\}}$ Eats × B

species	food
chimp	grain
chimp	insects
hen	grain
hen	insects
hyena	grain
hyena	insects

Eats

species	food
chimp	grain
chimp	insects
chimp	fruit
chimp	meat
hen	grain
hen	insects
hyena	meat
hyena	insects

$\pi_{\{species\}}$ Eats

species
chimp
hen
hyena

B

food
grain
insects

Join Operations

Theta-Join $\bowtie_\theta$

$R \bowtie_\theta S = \sigma_\theta(R \times S)$

A compound operation that computes a cross-product and then applies a selection operator with the selection condition θ

Animals

aid	name	species	age	feedtime
678	Squeaky	dolphin	6	10:30
325	Happy	monkey	1	8:30
327	Grumpy	monkey	1	9:00
921	Moma	orangutan	10	8:40

LivesIn

aid	eid
678	90
167	18
325	89
921	89

$\text{Animals} \bowtie_{\text{Animals.aid=LivesIn.aid}} \text{LivesIn}$

aid	name	species	age	feedtime	aid	eid
678	Squeaky	dolphin	6	10:30	678	90
325	Happy	monkey	1	8:30	325	89
921	Moma	orangutan	10	8:40	921	89

Theta-Join $\bowtie_\theta$

$R \bowtie_\theta S = \sigma_\theta(R \times S)$

Find all animals younger than each animal

$A1 \bowtie_{A1.age > A2.age} A2$

A_1

name	age
Happy	1
Squeaky	6
Moma	10

A_2

name	age
Happy	1
Squeaky	6
Moma	10

Step 1: $A_1 \times A_2$

name	age	name	age
Happy	1	Happy	1
Happy	1	Squeaky	6
Happy	1	Moma	10
Squeaky	6	Happy	1
Squeaky	6	Squeaky	6
Squeaky	6	Moma	10
Moma	10	Happy	1
Moma	10	Squeaky	6
Moma	10	Moma	10

Theta-Join $\bowtie_\theta$

$R \bowtie_\theta S = \sigma_\theta(R \times S)$

Find all animals younger than each animal

$A1 \bowtie_{A1.age>A2.age} A2$

A_1

name	age
Happy	1
Squeaky	6
Moma	10

A_2

name	age
Happy	1
Squeaky	6
Moma	10

Step 2: $\sigma_{A_1.age>A_2.age}(A_1 \times A_2)$

name	age	name	age
~~Happy~~	~~1~~	~~Happy~~	~~1~~
~~Happy~~	~~1~~	~~Squeaky~~	~~6~~
~~Happy~~	~~1~~	~~Moma~~	~~10~~
Squeaky	6	Happy	1
~~Squeaky~~	~~6~~	~~Squeaky~~	~~6~~
~~Squeaky~~	~~6~~	~~Moma~~	~~10~~
Moma	10	Happy	1
Moma	10	Squeaky	6
~~Moma~~	~~10~~	~~Moma~~	~~10~~

A_1

name	age
Happy	1
Squeaky	6
Moma	10

A_2

name	age
Happy	1
Squeaky	6
Moma	10

Theta-Join $\bowtie_\theta$

$R \bowtie_\theta S = \sigma_\theta(R \times S)$

Find all animals younger than each animal

$\text{A1} \bowtie_{\text{A1.age>A2.age}} \text{A2}$

$A_2 \bowtie_{\mathrm{A_1.age>A_2.age}} \mathrm{A_2}$

name	age	name	age
Squeaky	6	Happy	1
Moma	10	Happy	1
Moma	10	Squeaky	6

Joins

$R \bowtie S$

Theta-join: Use any logical expression θ

Equi-Join: theta join with θ being a conjunction of equalities

Natural Join: equi-join on all matching column names. Commonly used for primary-key / foreign key joins.

$$R \bowtie S = \pi_{\text{unique fields}}(\sigma_{\text{equality on matching fields}}(R \times S))$$

We define these special operator variants because we design special algorithms to implement them. Avoid cartesian products!

Outer Joins

⟕, ⟖, ⟗

Keep tuples from the left (⟕) or right (⟖) or both tables (⟗) for which there are no matches.

Can you derive the expression for this compound operator?

A_1

name	age
Happy	1
Squeaky	6
Moma	10

A_2

name	age
Happy	1
Squeaky	6
Moma	10

A_2 ⟕$_{A_1.age>A_2.age}$ A_2

name	age	name	age
Happy	1	NULL	NULL
Squeaky	6	Happy	1
Moma	10	Happy	1
Moma	10	Squeaky	6

Semi Joins

$\ltimes, \rtimes$

Project only attributes from the left ($\ltimes$) or right ($\rtimes$) table after a natural join

$$R \ltimes S = \pi_{\mathbb{A}_R}(R \bowtie S)$$

Can you derive the expression for this compound operator?

Animals

aid	name	species	age	feedtime
678	Squeaky	dolphin	6	10:30
325	Happy	monkey	1	8:30
327	Grumpy	monkey	1	9:00
921	Moma	orangutan	10	8:40

LivesIn

aid	eid
678	90
167	18
325	89
921	89

Animals $\ltimes$ LivesIn

aid	name	species	age	feedtime
678	Squeaky	dolphin	6	10:30
325	Happy	monkey	1	8:30
921	Moma	orangutan	10	8:40

From RA to SQL

Practical Extensions

Sort τ: Relations are sets (no order) but we often want our results ordered!

Group By $\gamma_{A_1,\ldots,A_k} R$: partitions tuples of a relation into '*groups*' defined by unique values for $A_1, \ldots, A_k$

Aggregation $\gamma_{A_1,\ldots,A_k} \mathrm{op}_B(R)$: Compute for *each group* an aggregate $\mathbf{op}$ ($\mathbf{op}$ can be $\mathbf{min}, \mathbf{max}, \mathbf{count},$ etc.) over some attribute B in relation R. If no groups are given, then compute the aggregate over the entire relation R

Duplicate Elimination $\delta(R)$: Remove any duplicate records.

SQL Structured Query Language

Relational Algebra

Algebra on sets

Operational / Imperative: an order of execution or a plan that can be constructed directly from the relation algebra expression.

Considered difficult for mere mortals

SQL

Rooted in Relational Calculus: based on first order logic

$\{A \mid A \in \text{Animals} \ \wedge A.\text{age} < 2\}$

Declarative: say what you want not how to get it.

Enables query optimization!

Codd's theorem established an equivalence between relational algebra and relational calculus

SQL ↔ RA

```
SELECT   -- projections and aggregations
FROM     -- tables referenced
JOIN     -- join expressions
WHERE    -- select expressions
ORDER BY     -- sort applied after query
GROUP BY     -- list of attributes to group by
HAVING   -- conditions applied after grouping
         -- and aggregations
LIMIT n -- returns n first rows
```

There are often multiple ways to write the same query in SQL as there are multiple relational algebra plans for the same query

Relational Algebra Equivalences

Rewrite Rules

Combine or break apart selection cascades

$$\sigma_{c_1 \wedge c_2 \wedge \cdots \wedge c_k}(R)$$
$$\equiv \sigma_{c_1}\left(\sigma_{c_2}\left(\ldots\left(\sigma_{c_k}(R)\right)\ldots\right)\right)$$

$$\begin{array}{c} \sigma_{c_1 \wedge c_2 \wedge \cdots \wedge c_k} \\ \uparrow \\ R \end{array} \quad \equiv \quad \begin{array}{c} \sigma_{c_k} \\ \uparrow \\ \ldots \\ \uparrow \\ \sigma_{c_1} \\ \uparrow \\ R \end{array}$$

Reorder selections

$$\sigma_{c_1}\left(\sigma_{c_2}(R)\right)$$
$$\equiv \sigma_{c_2}\left(\sigma_{c_1}(R)\right)$$

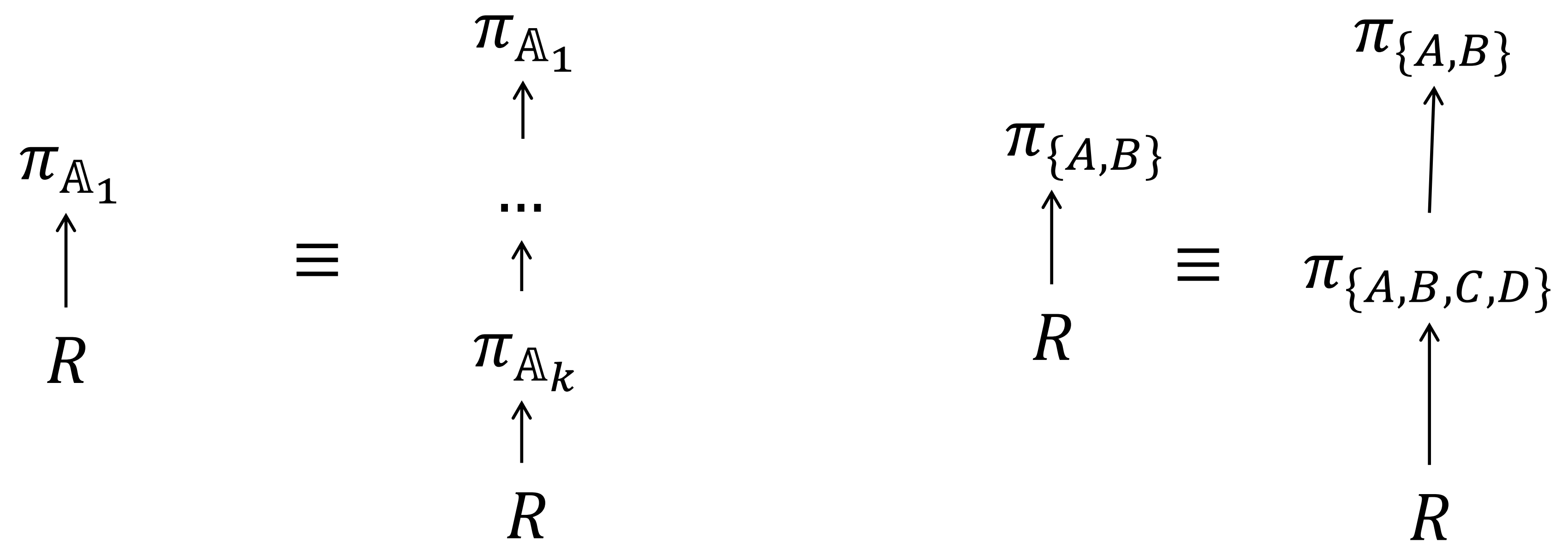
$$\begin{array}{c} \sigma_{c_1} \\ \uparrow \\ \sigma_{c_2} \\ \uparrow \\ R \end{array} \quad \equiv \quad \begin{array}{c} \sigma_{c_2} \\ \uparrow \\ \sigma_{c_1} \\ \uparrow \\ R \end{array}$$

Selections

Projections

Combine projection cascades

$$\pi_{\mathbb{A}_1}(R) \equiv \pi_{\mathbb{A}_1}\left(\pi_{\mathbb{A}_2}\left(\ldots\left(\pi_{\mathbb{A}_k}(R)\right)\ldots\right)\right), \text{if } \mathbb{A}_1 \subseteq \mathbb{A}_2 \subseteq \cdots \subseteq \cdots \mathbb{A}_k$$



Commutative

$$R \times S \equiv S \times R$$

$$R \bowtie S \equiv S \bowtie R$$

Associative

$$R \times (S \times T) \equiv (R \times S) \times T$$

$$R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T$$

All together now - join reordering:

$$R \bowtie (S \bowtie T) \equiv T \bowtie (S \bowtie R)$$

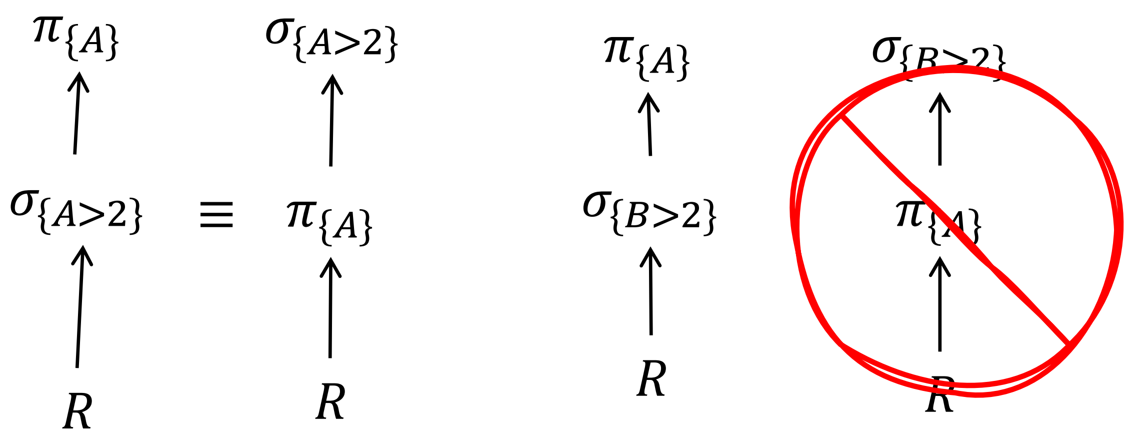
We are free to choose the outer (left) and inner (right) relation.

We are free to join the relations in any order we choose.

Joins & Cross Products

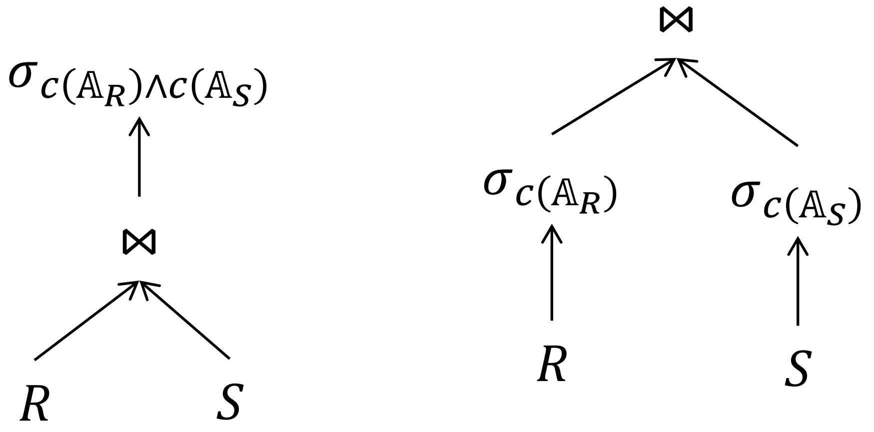
Projection push-down

$$\pi_{a_1}\left(\sigma_{c(a_1)}(R)\right) \equiv \sigma_{c(a_1)}\left(\pi_{a_1}(R)\right)$$



Selection push-down

$$\sigma_{c(\mathbb{A}_R)\wedge c(\mathbb{A}_S)}(R \bowtie S)$$
$$\equiv \sigma_{c(\mathbb{A}_R)}(R) \bowtie \sigma_{c(\mathbb{A}_S)}(S)$$



Select, Project & Join