CSC 365

Introduction to Database Systems

Relational Algebra

CAL POLY

- Operands in Relational Algebra are relations (or expressions that yield relations)
- A few additional derived operators are not listed: additional join types (Equijoin, Semijoin, Antijoin) and Division

(*derived* operators are marked with ✱)

Operator Name	Symbol
Selection	σ
Projection	π
Rename	ρ
Union	$\cup$
Difference	$-$
Intersection ✱	$\cap$
Cartesian Product	$\times$
Natural Join ✱	$\bowtie$
Theta Join ✱	$\bowtie_\theta$

Relational Algebra - Joins

CAL POLY

- Several types of join operations
  - Natural join
  - Theta join
  - **Equijoin**
  - **Semijoin**
  - **Antijoin**

- Selectively pair tuples from two relations.

- Joins can be expressed in terms of other basic operators we have seen (cartesian product, selection, projection)

Relational Algebra - Equijoin

A theta join where the θ condition is a conjunction of equality comparisons

FACULTY(<u>ID</u>, Name, Dept, Salary)
CHAIR(<u>Dept</u>, FacultyID)

Find salaries of department chairs:

$$\pi_{FACULTY.Dept,\ FACULTY.Salary}(FACULTY \bowtie_{FACULTY.ID\ =\ CHAIR.FacultyID} CHAIR)$$

Relational Algebra - Semijoin (⋉)

CAL POLY

A specialized form of natural join. The result includes attributes from only one of the relations (in the case of left semijoin: $R \ltimes S$, attributes from R) Given:

EMPLOYEE(EmpID, Name, DeptID)
DEPARTMENT(DeptID, DeptName)

EMPLOYEE ⋉ DEPARTMENT has the same schema as EMPLOYEE, lists all employees who are assigned to a department.

$$R \ltimes S = \boldsymbol{\pi}_{R.A1,\, R.A2,\, \ldots\,,\, R.An} (R \bowtie S)$$

Relational Algebra - Antijoin ($\triangleright$)

The result of $R \triangleright S$ (sometimes written as $R \overline{\ltimes} S$) includes only those tuples from R for which there is ***no tuple in S*** that is equal on the common attribute names. Antijoin is the complement of Semijoin:

$$R \triangleright S = R - R \ltimes S$$

Consider EMPLOYEE(Name, Dept) and CUSTOMER(Name, Address).
EMPLOYEE $\triangleright$ CUSTOMER would list all employees who *don't share* a name with any customers.

Relational Algebra - Semijoin & Antijoin - Why?

CAL POLY

- Useful shorthand, sometimes seen when discussing distributed databases, where one key goal is to minimize network transfer
- Note: Semijoin is ***not*** the same as SQL's `OUTER JOIN`
- Semijoin appears in SQL as `EXISTS` or `IN`
- Antijoin appears in SQL as `NOT EXISTS` or `NOT IN`

Relational Algebra - Division (/, ÷)

CAL POLY

Division allows us to compactly represent queries such as:

- Find the names of people who have visited *every* State Park in California
- Find the names of Netflix customers who have watched *every* film which features your favorite movie star.

Relational Algebra Operators

CAL POLY

- Five fundamental operators
  - Selection
  - Projection
  - Union
  - Difference
  - Cartesian Product
- Derived operators
  - Intersection
  - Joins: natural, theta, semi, anti
  - Division
- Utility operator
  - Rename

Relational Algebra - Algebraic Laws

CAL POLY

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

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

$$\sigma_{\theta 1 \wedge \theta 2}(R) = \sigma_{\theta 1}(\sigma_{\theta 2}(R)) = \sigma_{\theta 2}(\sigma_{\theta 1}(R))$$

$$\sigma_{\theta 1 \vee \theta 2}(R) = \sigma_{\theta 1}(R) \cup \sigma_{\theta 2}(R)$$

Relational Algebra - Algebraic Laws (Set Semantics)

CAL POLY

Selection distributes over the union, intersection, and set difference operations	$\sigma_\theta (R \cup S) = \sigma_\theta (R) \cup \sigma_\theta (S)$ $\sigma_\theta(R - S) = \sigma_\theta(R) - S = \sigma_\theta(R) - \sigma_\theta(S)$
Multiple applications of the same select have no effect (selection is idempotent)	$\sigma_\theta (\sigma_\theta (R)) = \sigma_\theta (R)$
Selection is commutative (order of selections does not matter)	$\sigma_{\theta 1} (\sigma_{\theta 2} (R)) = \sigma_{\theta 2} (\sigma_{\theta 1} (R))$
Projection is distributive over *set* union	$\pi_L (R \cup S) = \pi_L (R) \cup \pi_L (S)$
Projection *does not* distribute over intersection or difference	$\cancel{\pi_L (R \cap S) = \pi_L (R) \cap \pi_L (S)}$ $\cancel{\pi_L (R - S) = \pi_L (R) - \pi_L (S)}$

Relational Algebra - Algebraic Laws & Optimization

CAL POLY

- Laws that highlight equivalent expressions are useful during query optimization (an important task of the DBMS)

- Two equivalent expressions may have *very* different computational costs (when considering I/O & CPU)

Relational Algebra - Other Notations

Three ways to represent relational algebra expressions:

- One long expression with parentheses
- Tree representation
- Sequence of assignment statements (AKA "linear notation")

Relational Algebra - Tree Representation

We can represent a relational algebra expression as a tree:

- Internal nodes are operators
- Leaf nodes are relations

Sample Relation Instances

AIRPLANE

TailNum	Make	Model	MaxSpeed
C97W	Boeing	797	*null*
R53Q	Cessna	FG	220
T80H	Airbus	A380	634
G59K	Airbus	A320	450
P88T	Piper	Arrow	180
K30W	Boeing	707	450

FLIGHT

TailNum	PilotID	CopilotID	Runway	Date
R53Q	K407	D342	S-2	9/1/17
T80H	K407	null	W-2	9/21/17
C97W	D342	*null*	W-2	8/9/21
T80H	D342	K407	W-3	9/9/17

PILOT

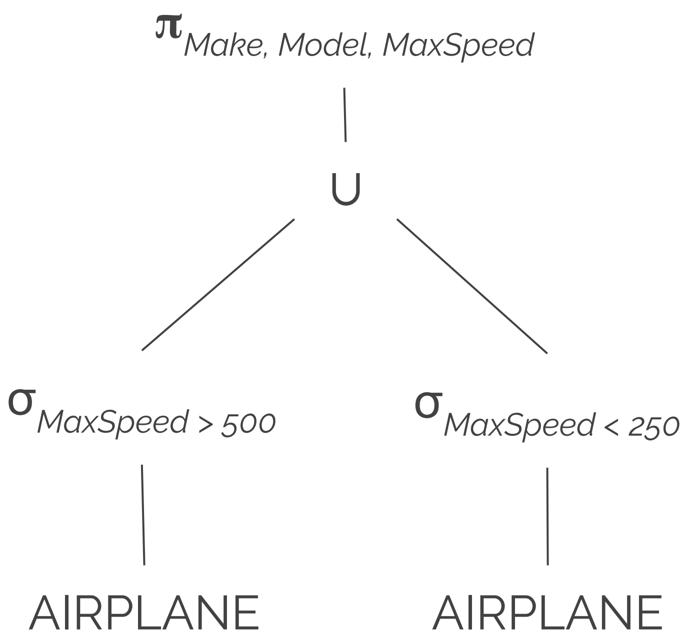
PilotID	Name
D342	Charlie
K407	Juliett
H452	Piper

Relational Algebra - Tree Representation

$$\pi_{Make,\ Model,\ MaxSpeed}(\sigma_{MaxSpeed\ >\ 500}(\text{AIRPLANE})$$
$$\cup$$
$$\sigma_{MaxSpeed\ <\ 250}(\text{AIRPLANE}))$$

Line breaks have no significance. This could be shown on one long line.

Equivalent expression?



Relational Algebra - Assignment / Linear Notation

CAL POLY

$\pi_{Make,\ Model,\ MaxSpeed}(\sigma_{MaxSpeed\ >\ 500}(\text{AIRPLANE})\ \cup\ \sigma_{MaxSpeed\ <\ 250}(\text{AIRPLANE}))$

Linear Notation:

R(TailNum, Make, Model, MaxSpeed) := $\sigma_{MaxSpeed\ >\ 500}$(AIRPLANE)

S(TailNum, Make, Model, MaxSpeed) := $\sigma_{MaxSpeed\ <\ 250}$(AIRPLANE)

T(TailNum, Make, Model, MaxSpeed) := R $\cup$ S

Answer(Make, Model, MaxSpeed) := $\pi_{Make,\ Model,\ MaxSpeed}$(T)

Attribute names may be omitted for brevity.

Relational Algebra - Examples

Given the database schema:

Relation	**Sample Tuple**
PRODUCT(maker, <u>model</u>, type)	(Dell, 1001, pc)
PC(<u>model</u>, speed, ram, hd, price)	(1001, 3.1, 8192, 512, 2114)
LAPTOP(<u>model</u>, speed, ram, hd, screen, price)	(2001, 1.7, 4096, 256, 13,3, 1699)
PRINTER(<u>model</u>, color, type, price)	(3001, true, laser, 239)

Relational Algebra - Examples

CAL POLY

PRODUCT

maker	<u>model</u>	type
Dell	1001	pc
Dell	2001	laptop
Orange	1002	pc
HP	1003	pc
HP	2002	laptop

PC

<u>model</u>	speed	ram	hd	price
1001	3.1	8	512	1200
1002	2.8	16	512	1300
1003	2.9	8	1024	1500

LAPTOP

<u>model</u>	speed	ram	hd	screen	price
2001	1.7	8	512	15	1599
2002	2.0	16	1024	13.3	999

Relational Algebra - Examples

CAL POLY

(1) What PC models have a speed of at least 3.00?

(2) Find the model number and price of all products (of any type) made by manufacturer *Orange*

Relational Algebra - Examples

CAL POLY

(3) Find the manufacturers (makers) that sell laptops, but not PCs.

Hint: Recall relational algebra's set operators ($\cup$, $\cap$, $-$)

Relational Algebra - Examples

CAL POLY

(3) Find the manufacturers (makers) that sell laptops, but not PCs.

Hint: Recall relational algebra's set operators (∪ , ∩, –)

$$\pi_{maker}(\text{PRODUCT} \bowtie \text{LAPTOP}) - \pi_{maker}(\text{PRODUCT} \bowtie \text{PC})$$

(4) Find hard-disk sizes that occur in two or more PCs.

Hint: It's entirely acceptable to join a relation to itself (a "self-join")

Relational Algebra - Examples

CAL POLY

(4) Find hard-disk sizes that occur in two or more PCs.

Hint: It's entirely acceptable to join a relation to itself (a "self-join")

$$\boldsymbol{\pi}_{hd}\,(\sigma_{\text{PC1.model != PC2.model} \wedge \text{PC1.hd = PC2.hd}}(\boldsymbol{\rho}_{\text{PC1}}(\text{PC}) \times \boldsymbol{\rho}_{\text{PC2}}(\text{PC})))$$

Relational Algebra - Examples

(5) Find the names of all manufacturers that sell products of <u>all</u> types (pc and laptop in our example; the query should not be specifically limited to these values)

Relational Algebra - Division (/, ÷)

Informally, division is related to the Cartesian produc as follows:

$C = A \times B$

$A = C \div B$

$B = C \div A$

(This is an oversimplification that does not hold in the case of empty tables.)

Relational Algebra - Division Example

CAL POLY

R

A
4
8

S

B
3
1
7

R x S

A	B
4	3
4	1
4	7
8	3
8	1
8	7

(R x S) ÷ S

A
4
8

(R x S) ÷ R

B
3
1
7

Relational Algebra - Division Example

CAL POLY

R

A
4
8

S

B
3
1
7

T

A	B
4	3
4	1
~~4~~	~~7~~
8	3
8	1
8	7

In this example, the row (4,7) has been removed from T

T ÷ S

A
~~4~~
8

T ÷ R

B
3
1
~~7~~

Relational Algebra - Division

Division can be represented as a combination of projection, difference and cross product.

$\boldsymbol{R \div S}$ =

$\boldsymbol{\pi}_{Schema(R)-Schema(S)}(R) - \boldsymbol{\pi}_{Schema(R)-Schema(S)}((\boldsymbol{\pi}_{Schema(R)-Schema(S)}(R) \times S) - R)$

(here, Schema(R) and Schame(S) represent the attribute sets of relation R and S, respectively)

Relational Model / SQL

Cal Poly

	Relational Model	SQL
Structure	Attributes, Tuples, Relations	Columns, Rows, Tables
Manipulation	Relational Algebra	DML, Queries

Constraints	Primary Key, Referential (Foreign Key), Domain, Relational Algebra expressions to define constraints	`CREATE TABLE`
		`ALTER TABLE`
		Triggers