

Relational Algebra

Dr Paolo Guagliardo



THE UNIVERSITY *of* EDINBURGH
informatics

v26.0

Changelog

v26.0 Initial version

Data model

- ▶ A **relation** is a **set** of **records**
over the same **set** of (distinct) attribute names
- ▶ A **record** is a **total function** from attribute names to values

Schema

- ▶ Set of **relation names**
- ▶ Set of **distinct attributes** for each table
Note that columns are not ordered

Instance

- ▶ Actual data (that is, the records in each relation)

Relational algebra

Procedural query language

A relational algebra expression

- ▶ takes as input one or more relations
- ▶ applies a **sequence of operations**
- ▶ returns a relation as output

Operations:

Projection (π)	Union ($\cup$)
Selection (σ)	Intersection ($\cap$)
Product ($\times$)	Difference ($-$)
Renaming (ρ)	

The application of each operation results in a new (virtual) relation that can be used as input to other operations

Projection

- ▶ **Vertical operation**: choose some of the columns
- ▶ **Syntax**: $\pi_{\text{set of attributes}}(\text{relation})$
- ▶ $\pi_{A_1, \dots, A_n}(R)$ takes only the values of attributes $A_1, \dots, A_n$ for each tuple in R

Customer

CustID	Name	City	Address
cust1	Renton	Edinburgh	2 Wellington Pl
cust2	Watson	London	221B Baker St
cust3	Holmes	London	221B Baker St

$\pi_{\text{Name, City}}(\text{Customer})$

Name	City
Renton	Edinburgh
Watson	London
Holmes	London

Selection

- ▶ **Horizontal operation**: choose rows satisfying some condition
- ▶ **Syntax**: $\sigma_{\text{condition}}(\text{relation})$
- ▶ $\sigma_{\theta}(R)$ takes only the tuples in R for which θ is satisfied

term := attribute | constant

θ := **term** **op** **term** with **op** $\in \{=, \neq, >, <, \geq, \leq\}$
| $\theta \wedge \theta$ | $\theta \vee \theta$ | $\neg\theta$

Example of selection

Customer

CustID	Name	City	Age
cust1	Renton	Edinburgh	24
cust2	Watson	London	32
cust3	Holmes	London	35

$\sigma_{\text{City} \neq \text{'Edinburgh'} \wedge \text{Age} < 33}(\text{Customer})$

CustID	Name	City	Age
cust2	Watson	London	32

Efficiency (1)

Consecutive selections can be combined into a single one:

$$\sigma_{\theta_1}(\sigma_{\theta_2}(R)) \equiv \sigma_{\theta_1 \wedge \theta_2}(R)$$

↳ is equivalent to: two queries Q_1 and Q_2 are **equivalent** if Q_1 returns the same answers as Q_2 on every database

Example

$$Q_1 = \sigma_{\text{City} \neq \text{'Edinburgh'}}(\sigma_{\text{Age} < 33}(\text{Customer}))$$

$$Q_2 = \sigma_{\text{City} \neq \text{'Edinburgh'} \wedge \text{Age} < 33}(\text{Customer})$$

$$Q_1 \equiv Q_2 \text{ but } Q_2 \text{ **faster** than } Q_1 \text{ in general}$$

Efficiency (2)

Projection can be pulled in front of selection

$$\sigma_{\theta}(\pi_{\alpha}(R)) \equiv \pi_{\alpha}(\sigma_{\theta}(R))$$

only if all attributes mentioned in θ appear in α

Example

$$Q_1 = \pi_{\text{Name, City, Age}}(\sigma_{\text{City} \neq \text{'Edinburgh'} \wedge \text{Age} < 33}(\text{Customer}))$$

$$Q_2 = \sigma_{\text{City} \neq \text{'Edinburgh'} \wedge \text{Age} < 33}(\pi_{\text{Name, City, Age}}(\text{Customer}))$$

Question: Which one is more efficient?

Cartesian product

$R \times S$ **concatenates** each tuple of R with all the tuples of S

Note: the relations must have **disjoint** sets of attributes

Example

R	A	B	$\times$	S	C	D	$=$	$R \times S$	A	B	C	D
	1	2			1	a			1	2	1	a
	3	4			2	b			1	2	2	b
					3	c			1	2	3	c
									3	4	1	a
									3	4	2	b
									3	4	3	c

Expensive operation:

- ▶ $\text{card}(R \times S) = \text{card}(R) \times \text{card}(S)$
- ▶ $\text{arity}(R \times S) = \text{arity}(R) + \text{arity}(S)$

Joining relations

Combining Cartesian product and selection

Customer: ID, Name, City, Address

Account: Number, Branch, CustID, Balance

We can join customers with the accounts they own as follows

$$\sigma_{ID=CustID}(Customer \times Account)$$

Renaming

Gives new names to the attribute of a relation

Syntax: $\rho_{\text{replacement}_1, \dots, \text{replacement}_k}(\text{relation})$

where each **replacement** has the form **old** $\rightarrow$ **new**

Requirements

Let X is the set of attributes of the input relation, then

- ▶ The l.h.s. of each replacement belongs to X
- ▶ No two replacements have the same l.h.s.
- ▶ Composing the identity on X with all replacements results in an injective function

Examples of renaming

$$\blacktriangleright \rho_{B \rightarrow E} \left(\overbrace{\begin{array}{c|ccc} & \mathbf{A} & \mathbf{B} & \mathbf{C} \\ \hline a & b & c \\ 1 & 2 & 3 \end{array}}^R \right) = \begin{array}{c|ccc} & \mathbf{A} & \mathbf{E} & \mathbf{C} \\ \hline a & b & c \\ 1 & 2 & 3 \end{array}$$

$\blacktriangleright \rho_{D \rightarrow E}(R)$ and $\rho_{B \rightarrow A}(R)$ are **illegal** (why?)

$\blacktriangleright \rho_{A \rightarrow B, B \rightarrow A}(R)$ is **legal** (what does it do?)

$\blacktriangleright \rho_{A \rightarrow B, B \rightarrow A, A \rightarrow E}(R)$ is also **illegal** (why?)

Natural join

Joins two tables for equality on their **common attributes**

Example

Customer: **CustID**, Name, City, Address

Account: Number, Branch, **CustID**, Balance

$\text{Customer} \bowtie \text{Account} \equiv$

$$\pi_{X \cup Y} \left(\sigma_{\text{CustID} = \text{CustID}'} \left(\text{Customer} \times \rho_{\text{CustID} \rightarrow \text{CustID}'}(\text{Account}) \right) \right)$$

where $X = \{ \text{all attributes of Customer} \}$

$Y = \{ \text{all attributes of Account} \}$

From SQL to relational algebra

SELECT $\mapsto$ projection π

FROM $\mapsto$ Cartesian product $\times$

WHERE $\mapsto$ selection σ

SELECT $A_1, \dots, A_m$
FROM $T_1, \dots, T_n$ $\mapsto \pi_{A_1, \dots, A_m}(\sigma_{\langle \text{condition} \rangle}(T_1 \times \dots \times T_n))$
WHERE $\langle \text{condition} \rangle$

Common attributes in $T_1, \dots, T_n$ must be renamed

Set operations

Union

R	A	B	$\cup$	S	A	B	=	R $\cup$ S	A	B
	a1	b1			a1	b1			a1	b1
	a2	b2			a3	b3			a2	b2
									a3	b3

Intersection

R	A	B	$\cap$	S	A	B	=	R $\cap$ S	A	B
	a1	b1			a1	b1			a1	b1
	a2	b2			a3	b3				

Difference

R	A	B	$-$	S	A	B	=	R $-$ S	A	B
	a1	b1			a1	b1			a2	b2
	a2	b2			a3	b3				

The relations must have the same set of attributes

Union and renaming

R	Father	Child	S	Mother	Child
	George	Elizabeth		Elizabeth	Charles
	Philip	Charles		Elizabeth	Andrew
	Charles	William			

We want to find the relation **parent-child**

$\rho_{\text{Father} \rightarrow \text{Parent}}(\text{R}) \cup \rho_{\text{Mother} \rightarrow \text{Parent}}(\text{S})$	=	<table><tr><th>Parent</th><th>Child</th></tr><tr><td>George</td><td>Elizabeth</td></tr><tr><td>Philip</td><td>Charles</td></tr><tr><td>Charles</td><td>William</td></tr><tr><td>Elizabeth</td><td>Charles</td></tr><tr><td>Elizabeth</td><td>Andrew</td></tr></table>	Parent	Child	George	Elizabeth	Philip	Charles	Charles	William	Elizabeth	Charles	Elizabeth	Andrew
Parent	Child													
George	Elizabeth													
Philip	Charles													
Charles	William													
Elizabeth	Charles													
Elizabeth	Andrew													

Full relational algebra

Primitive operations: π , σ , $\times$, ρ , $\cup$, $-$

Removing any of these results in a **loss of expressive power**

Derived operations

$\bowtie$ can be expressed in terms of π , σ , $\times$, ρ

$\cap$ can be expressed in terms difference:

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

Other derived operations

Theta-join	$R \bowtie_{\theta} S \equiv \sigma_{\theta}(R \times S)$
Equijoin	$\bowtie_{\theta}$ where θ is a conjunction of equalities
Semijoin	$R \ltimes_{\theta} S \equiv \pi_X(R \bowtie_{\theta} S)$ where X is the set of attributes of R
Antijoin	$R \bar{\ltimes}_{\theta} S \equiv R - (R \ltimes_{\theta} S)$

Why use these operations?

- ▶ to write things more succinctly
- ▶ they can be optimized independently

Division

R over set of attributes X

S over set of attributes $Y \subset X$

Let $Z = X - Y$

$$\begin{aligned} R \div S &= \{ r \in \pi_Z(R) \mid \text{for every } s \in S, rs \in R \} \\ &= \{ r \in \pi_Z(R) \mid \{r\} \times S \subseteq R \} \\ &= \pi_Z(R) - \pi_Z(\pi_Z(R) \times S - R) \end{aligned}$$

Division: Example

Exams		DPT
Student	Course	Course
John	Databases	Databases
John	Networks	Programming
Mary	Programming	
Mary	Math	
Mary	Databases	

Exams ÷ DPT =

Student
Mary

= $\pi_{\text{Student}}(\text{Exams}) - \pi_{\text{Student}}(\pi_{\text{Student}}(\text{Exams}) \times \text{DPT} - \text{Exams})$