Introduction to Databases

(INFR10080)

(Relational Algebra)

(Fall 2025)

THE UNIVERSITY of EDINBURGH

Changelog

v25.0 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: $\pi_{\text{set of attributes}}(\text{relation})$
- $\pi_{A_1,\ldots,A_n}(R)$ takes only the values of attributes $A_1, \ldots, A_n$ 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

$$\begin{aligned} \text{term} &:= \text{attribute} \mid \text{constant} \\ \theta &:= \text{term } \textbf{op} \text{ term with } \textbf{op} \in \{=, \neq, >, <, \geqslant, \leqslant\} \\ &\quad \mid \theta \wedge \theta \mid \theta \vee \theta \mid \neg\theta \end{aligned}$$

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

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

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

$Q_1 \equiv Q_2$ but Q_2 **faster** than Q_1 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
	1	2
	3	4

$\times$

S	C	D
	1	a
	2	b
	3	c

$=$

$R \times S$	A	B	C	D
	1	2	1	a
	1	2	2	b
	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

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

Renaming

Gives new names to the attribute of a relation

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

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

- $\rho_{B \to E}\left(\overbrace{\begin{array}{ccc} \textbf{A} & \textbf{B} & \textbf{C} \\ \hline \text{a} & \text{b} & \text{c} \\ 1 & 2 & 3 \end{array}}^{R}\right) = \begin{array}{ccc} \textbf{A} & \textbf{E} & \textbf{C} \\ \hline \text{a} & \text{b} & \text{c} \\ 1 & 2 & 3 \end{array}$

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

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

- $\rho_{A \to B,\, B \to A,\, A \to E}(R)$ 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

Customer $\bowtie$ Account $\equiv$

$$\pi_{X \cup Y}\Big(\sigma_{\text{CustID}=\text{CustID}'}\big(\text{Customer} \times \rho_{\text{CustID} \to \text{CustID}'}(\text{Account})\big)\Big)$$

where $X = \{$ all attributes of Customer $\}$
$Y = \{$ all attributes of Account $\}$

From SQL to relational algebra

SELECT $\mapsto$ projection π
FROM $\mapsto$ Cartesian product $\times$
WHERE $\mapsto$ selection σ

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

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

Set operations

Union

R	A	B
	a1	b1
	a2	b2

$\cup$

S	A	B
	a1	b1
	a3	b3

=

R ∪ S	A	B
	a1	b1
	a2	b2
	a3	b3

Intersection

R	A	B
	a1	b1
	a2	b2

$\cap$

S	A	B
	a1	b1
	a3	b3

=

R ∩ S	A	B
	a1	b1

Difference

R	A	B
	a1	b1
	a2	b2

$-$

S	A	B
	a1	b1
	a3	b3

=

R − S	A	B
	a2	b2

The relations must have the same set of attributes

Union and renaming

R	Father	Child
	George	Elizabeth
	Philip	Charles
	Charles	William

S	Mother	Child
	Elizabeth	Charles
	Elizabeth	Andrew

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

$$\rho_{\text{Father}\to\text{Parent}}(\text{R}) \cup \rho_{\text{Mother}\to\text{Parent}}(\text{S}) =$$

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 $R \ltimes_\theta S \equiv \pi_X\big(R \bowtie_\theta S\big)$
where X is the set of attributes of R

Antijoin $R \,\overline{\ltimes}_\theta\, S \equiv R - (R \ltimes_\theta S)$

Why use these operations?

- to write things more succintly
- they can be optimized independently

Division

R over set of attributes X

S over set of attributes $Y \subset X$

Let $Z = X - Y$

$$R \div S = \left\{\ r \in \pi_Z(R) \mid \text{for every } s \in S,\ rs \in R\ \right\}$$

$$= \left\{\ r \in \pi_Z(R) \mid \{r\} \times S \subseteq R\ \right\}$$

$$= \pi_Z(R) - \pi_Z\big(\pi_Z(R) \times S - R\big)$$

Division: Example

Exams

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

DPT

Course
Databases
Programming

$$\text{Exams} \div \text{DPT} \quad =$$

Student
Mary

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

How can we express division in SQL?

Further Reading:

Matos, V.M. and Grasser, R., 2002. *A simpler (and better) SQL approach to relational division*. Journal of Information Systems Education, 13(2)