Database Systems

Relational Algebra

Prof. Dr. Agnès Voisard   Muhammed-Ugur Karagülle

Institute of Computer Science, Databases and Information Systems Group

Fraunhofer FOKUS

2025

Notes

Notes

Collection of operations for manipulating relations.
The result of each operation is a new relation.
2 groups of operations:

- Set operations:
  - Union ($\cup$)
  - Intersection ($\cap$)
  - Difference ($\setminus$)
  - Cartesian product ($\times$)
- Operations developed specifically for relational databases:
  - Projection (π)
  - Selection (σ)
  - Join ($\bowtie$)

Reference relations:

Relation R1

A	B	C
a	b	c
d	a	f
c	b	d

Relation R2

D	E	F
b	g	a
d	a	f

HOBBY

PersonName	Age	HobbyName
Jane	24	Fly fishing
Melony	25	Singing
Brian	29	Tennis
Brian	29	Jogging
Charlie	31	Dancing
Steve	24	Singing

Projection (Π)

Unary operation.

Elimination of some attributes of a relation.

$\Pi_{<Attribute-List>}$(Relation)

$$\Pi_X(r) = \{ t[X] \mid t \in r \}, X \subseteq R.$$

Examples of projection

Example:

$\Pi_{A,C}(R1)$

A	C
a	c
d	f
c	d

Examples of projection (cont'd)

Example:

$\Pi_{PersonName,Age}$ (HOBBY). Duplicate elimination.

PersonName	Age
Jane	24
Melony	25
Brian	29
~~Brian~~	~~29~~
Charlie	31
Steve	24

Selection (σ)

Get the tuples of R that satisfy a certain condition C.
$\sigma_{<C>}(R)$.

Condition: logical formula made up of clauses of the form:

$A_i \; \theta \; A_j$ HobbyName="Singing"
$A_i \; \theta \; k$

where A_i, A_j are attributes of R,
k constant value from the domain of A_i,
$\theta \in \{=, <, \leq, >, \geq, \neq\}$.

Clauses connected by the Boolean operators "and" ($\wedge$), "or" ($\vee$). The negation "not" ($\neg$) can also be used.

$$Cond = (A_i = k),$$
$$\sigma_{A_i = k}(r) = \{t \in r \mid t[A_i] = k\}.$$

Example:

$\sigma_{B=b}(R1)$.

A	B	C
a	b	c
c	b	d

Example:

Select the *people who are singing*: in the HOBBY database: $\sigma_{HobbyName=``Singing''}$(HOBBY).

PersonName	Age	HobbyName
Melony	25	Singing
Steve	24	Singing

Example:

Who are the 24 year-old people who are singing?:

$\sigma_{Age=24 \wedge HobbyName=``Singing''}$ (HOBBY).

PersonName	Age	HobbyName
Steve	24	Singing

Selection is **commutative**.

$\sigma_{<cond1>}(\sigma_{<cond2>}$ (R)) = $\sigma_{<cond2>}(\sigma_{<cond1>}$ (R))

Cascade of selections $\equiv$ selection with conjunctive condition (“and”).

$\sigma_{<cond1>}(\sigma_{<cond2>}(\ldots\sigma_{<condn>}))$ (R) = $\sigma_{<cond1>\wedge cond2\wedge\ldots\wedge condn>}$ (R).

Sequences of operations and renaming

Several operations one after the other:

1. One operation at a time and store the intermediate result, or
2. Single **relational algebra expression** by nesting the operations.

Example:

Name of the people who sing?

HOBBY-TEMP $\leftarrow \sigma_{HobbyName=''Singing''}$(HOBBY))

RESULT $\leftarrow \Pi_{PersonName}$(HOBBY-TEMP).

Example:

$\Pi_{PersonName}(\sigma_{HobbyName=''Singing''}$(HOBBY).

Renaming attributes in intermediate and result relations: list the new attributes.

Example:

HOBBY-TEMP $\leftarrow \sigma_{HobbyName=''Singing''}$(HOBBY)
RESULT(Name) $\leftarrow \Pi_{PersonName}$(HOBBY-TEMP).

Input: 2 relations:

$R(A_1, \ldots, A_n)$ of arity n
$S(B_1, \ldots, B_m)$ of arity m.

Cartesian product of R and S:

$T = R \times S$.

$R_1 \times R_2 = \{t = t_1 + t_2 \mid t_1 \in R_1, t_2 \in R_2\}$.

Schema of T: $T(A_1, \ldots, A_n, B_1, \ldots, B_m)$

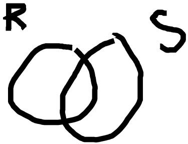
Value of T: set of possible $(n+m)$-tuples whose first n components form a tuple in R and last m components form a tuple in S.

Example:

R1 X R2

A	B	C	D	E	F
a	b	c	b	g	a
a	b	c	d	a	f
d	a	f	b	g	a
d	a	f	d	a	f
c	b	d	b	g	a
c	b	d	d	a	f

Input: 2 relations:



$R(A_1, \ldots, A_n)$ and
$S(B_1, \ldots, B_m)$

- Same arity
- Compatible domains

$T = R \cup S$.

Schema of T: $T(A_1, \ldots, A_n)$ or $T(B_1, \ldots, B_m)$ or other attribute names with same domain.

Value of T: $\{t \mid t \in R \text{ or } t \in S\}$,

Example:

R1 U R2

a	b	c
d	a	f
c	b	d
b	g	a
~~d~~	~~a~~	~~f~~

Input: 2 relations:

$R(A_1, \ldots, A_n)$ and
$S(B_1, \ldots, B_m)$

- Same arity
- Compatible domains

$T = R - S$.

Schema of T: $T(A_1, \ldots, A_n)$ or $T(B_1, \ldots, B_m)$ or other attribute names with same domain.

Value of T: $\{t \mid t \in R \text{ and } t \notin S\}$

Example:

R1 - R2

a	b	c
c	b	d

Input: 2 relations:

$R(A_1, \ldots, A_n)$ and
$S(B_1, \ldots, B_m)$

- Same arity
- Compatible domains

$T = R \cap S$

Schema of T: $T(A_1, \ldots, A_n)$ or $T(B_1, \ldots, B_m)$ or other attribute names with same domain.

Value of T: $\{t \mid t \in R \text{ and } t \in S\}$

Example:

R1 ∩ R2

d	a	f

Input: 2 relations with at least one attribute in common.

R1

A	B	C
a	b	c
d	b	c
b	b	f
c	a	d

R2

B	C	D
b	c	d
b	c	e
a	d	b

R1 ⋈ R2

A	B	C	D
a	b	c	d
a	b	c	e
d	b	c	d
d	b	c	e
c	a	d	b

$R \bowtie S$:

1. Compute $R \times S$ (schema (A,B,C,B,C,D))
2. For the attributes A_k common to R and S select the tuples with the same value for $R[A_k]$ and $S[A_k]$ (here B,C).
3. For each A_k project out column $S[A_k]$ and keep all other attributes.

Natural join $\bowtie$ (cont'd)

$R(A_1, A_2, \ldots, \underline{A_i}, \underline{A_{i+1}})$ and
$S(\underline{A_i}, \underline{A_{i+1}}, \ldots, A_{n-1}, A_n)$
V set of attributes in common.

If $U = \{A_1, A_2, \ldots, A_i, A_{i+1}, \ldots, A_n\}$

Schema of T: $T(U) = T(A_1, A_2, \ldots, A_i, A_{i+1}, \ldots A_n)$.

Value of T: $\Pi_U(\sigma_{\forall A_k \in V, R[A_k]=S[A_k]}(R \times S))$

Join (Θ-join)

Input: 2 relations

$R(A_1, \ldots, A_n)$
$S(B_1, \ldots, B_k$

$R \underset{A_i \theta B_j}{\bowtie} S$:

- Tuples in $R \times S$ such that $A_i \theta B_j$ is true
- $\theta \in \{=, <, \leq, >, \geq, \neq\}$

$T = R \underset{A_i \theta B_j}{\bowtie} S$

Schema of $T : T(A_1, \ldots, A_n, B_1, \ldots, B_k)$
Value of $T = \sigma_{A_i \theta B_j}(R \times S)$

Join (Θ-join) (cont'd)

Example:

$R \underset{B<D}{\bowtie} S$

R

A	B	C
1	2	3
4	5	6
7	8	9

S

D	E
3	1
6	2

$R \underset{B<D}{\bowtie} S$

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

Join (Equijoin, Θ is "=")

Example:

$R \underset{C=D}{\bowtie} S$

R

A	B	C
1	2	3
4	5	6
7	8	9

S

D	E
3	1
6	2

$R \underset{C=D}{\bowtie} S$

A	B	C	D	E
1	2	3	3	1
4	5	6	6	2

Example:

R

A	B
1	a
1	b
3	a

S

C	D	E
1	b	a
2	b	c
4	a	a

$R \bowtie S$
A =< C

A	B	C	D	E

$R \bowtie S$
B = D

A	B	C	D	E

Semijoin of $R(A_1, \ldots, A_n)$ by S:

$R \propto S$ is the projection of $R \bowtie S$ onto the attributes of R:

$$R \propto S = \Pi_{A_1,\ldots,A_n}(R \bowtie S)$$

R1

A	B	C
a	b	c
d	b	c
b	b	f
c	a	d

R2

B	C	D
b	c	d
b	c	e
a	d	b

R1 $\ltimes$ R2

A	B	C
a	b	c
d	b	c
c	a	d

More on joins!

The most expensive operation of the relational algebra.

Comparing all pairs of tuples: $O(n^2)$ time (relations of size n, number of tuples).

One way to compute equijoin or natural joins: sort both relations and merge the lists.

Time $O(m + 2nlogn)$, m number of tuples of the join (max n^2).

=> Avoid doing joins
Transform an algebraic expression into an equivalent one that can be evaluated faster.

=> **Query optimization techniques**.

R of arity m,
S of arity n.
$m > n$ and $S \neq \emptyset$.

$T = R \div S$: set of $(m-n)$-tuples $a_1, \ldots, a_{m-n}$ such that for all n-tuples $a_{m-n+1}, \ldots, a_m$ in S, the tuple $a_1, \ldots, a_m$ is in R.

R

A	B	C	D
a	b	c	d
a	b	e	f
b	c	e	f
e	d	c	d
e	d	e	f
a	b	d	e

S

C	D
c	d
e	f

R - S

A	B
a	b
e	d

Complete set of relational algebra operations

Set of relational algebra operations $\{\sigma, \pi, \cup, -, \times\}$ is a **complete set**: any of the other relational algebra operations can be expressed as a sequence of operations from this set.

Join
$R \bowtie S \equiv \sigma_{cond}(R \times S)$

Intersection not required:
$R \cap S \equiv (R \cup S) - ((R - S) \cup (S - R))$

Relational database schema:

EMPLOYEE(<u>EName</u>, Salary, Dept)
DEPARTMENT(<u>DName</u>, ManagerName, Location)

Retrieve all employees who work in the sales department.

$\sigma_{Dept=\text{``}sales\text{''}}$ (EMPLOYEE)

Give the name of all employees.

Give the name of Miller’s manager.

Give the name of the employees who work in Berkeley.

How about the query: ”Display all managers of Smith”’.

Aggregate functions

Mathematical aggregate functions on collections on values (tuples).

- SUM
- AVERAGE
- MAXIMUM
- MINIMUM or
- COUNT

Recursive closure

Applied to **recursive relationship** between tuples of the same type.

Example:

Employee relation relates an employee (supervisee) to another one.

Recursive operation: Retrieve all supervisees e' of an employee e **at all levels**.

Relational algebra: Ok at a specific level.
=> Retrieve employees e' step-by-step and take the union.
Problem: Where to stop? (looping mechanism).

Questions

What will come next?

1 Welcome to Database Systems
2 Introduction to Database Systems
3 Entity Relationship Design Diagram (ERM)
4 Relational Model
5 Relational Algebra
6 Structured Query Language (SQL)
7 Relational Database Design - Functional Dependencies
8 Relational Database Design - Normalization
9 Online Analytical Processing + Embedded SQL
10 Data Mining
11 Physical Representation - Storage and File Structure
12 Physical Representation - Indexing and Hashing
13 Transactions
14 Concurrency Control Techniques
15 Recovery Techniques
16 Query Processing and Optimization