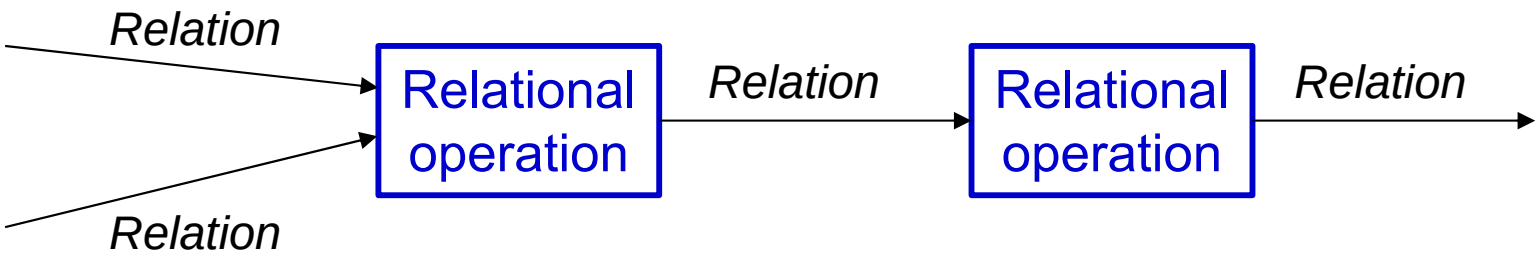
Relational algebra

Operations on relations

- Access data (“read” operations)
  - Retrieve tuples from a relation (Select)
  - Retrieve specific attributes of all tuples from a relation (Project)
  - Retrieve combined tuples from various relations (Join)
  - …
- Create/Modify data (“write” operations)
  - Add tuples to a relation (Insert)
  - Remove tuples from a relation (Delete)
  - Modify tuples in a relation (Delete + Insert)

Relational operations and queries

- Relational operations are a collection of operations to manipulate relations
  - Input and output of operations are relations
- A query is a composition of relations using relational operations
  - The result (output) of the query execution is also a relation
  - Schema of the input is fixed for a query
  - Schema of the output is determined by the query specifics



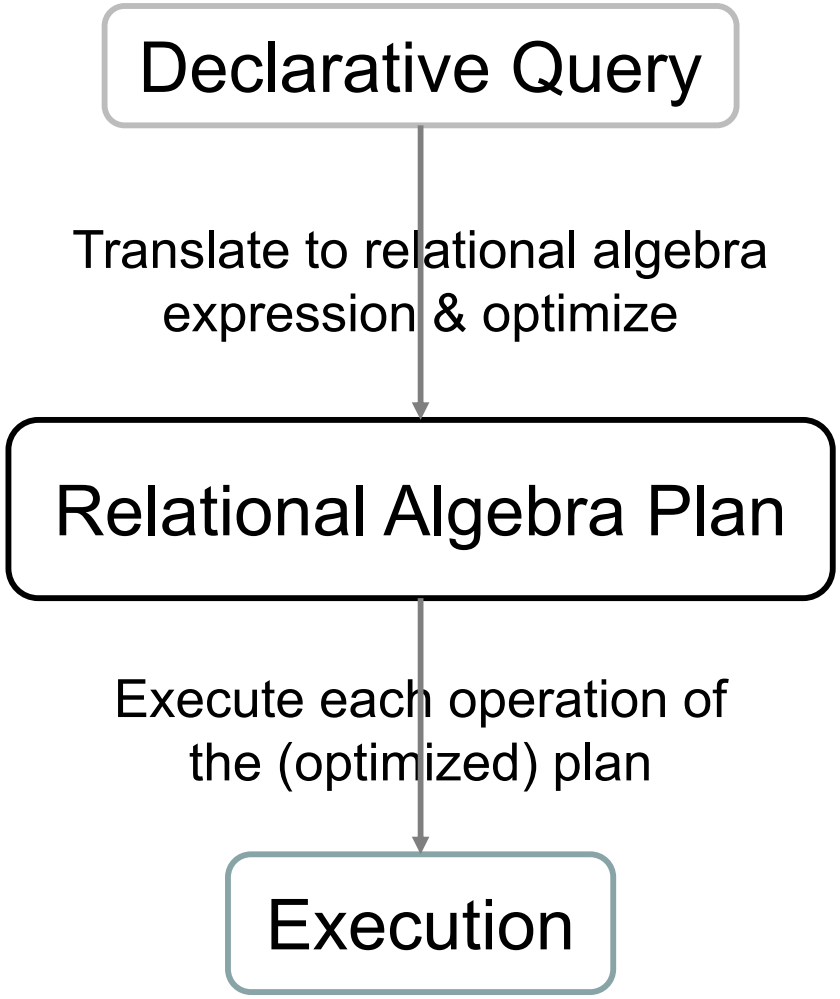
Relational algebra: Formal relational query language

WHAT

Return related data from relations R1, R2, R3, …, satisfying given conditions c1, c2, …

HOW

Iterate over R1…
…join with R2…
…join with R3…
…
…select tuples satisfying c1,c2…
…eliminate duplicates…
…and that's the final answer !



Relational algebra:

- Provides a formal/precise and optimizable framework to execute declarative queries
- Underlies most query languages for the relational model

Algebras examples

Abstract algebra	Algebra of arithmetic	Relational algebra
Operands	Variables (e.g., *x*) and constants (e.g., *2*)	Relations (e.g., *Person(Name,Age)* and associated instances)
Operations	Addition (+), subtraction (-), etc.	Selection (σ), projection (π), etc.
Properties of operations	E.g., addition is commutative	E.g., selection is commutative
Expressions*	((x + 7)/(2 / -3)) + x	$\pi_{\text{Name}}(\sigma_{\text{Age} \geq 20 \wedge \text{Age} < 30}(\text{Person}))$

*Obtained by applying operators to atomic operands and/or other expressions.

Relational operations

Basic

- Selection (σ)
- Projection (π)
- Cartesian product ($\times$)
- Set operations
  - Union ($\cup$)
  - Difference ($-$ or $\setminus$)

Derived and Auxiliary

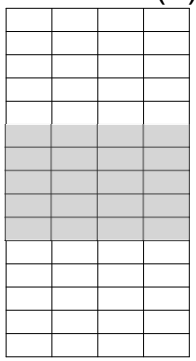
- Renaming (ρ)
- Join ($\bowtie$)
  - Theta, equi, natural, etc.
- Set operations
  - Intersection ($\cap$)
  - Division ($\div$)

Extended

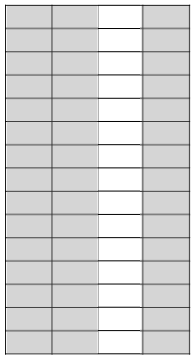
- Aggregate functions and grouping (γ)
- Generalized projection
- Sort (τ)
- Duplicate elimination (δ)

Relational operations – graphical overview

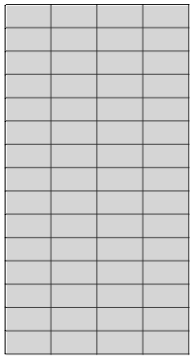
Selection (σ)



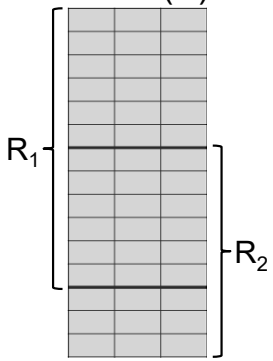
Projection (π)



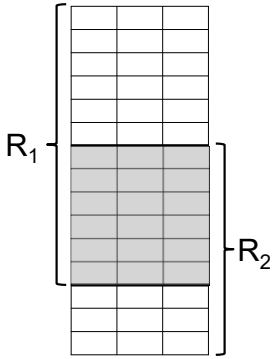
Renaming (ρ)



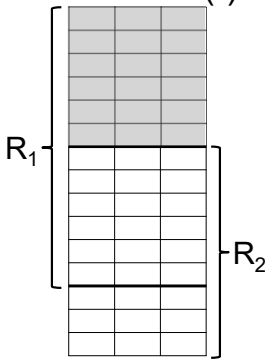
Union ($\cup$)



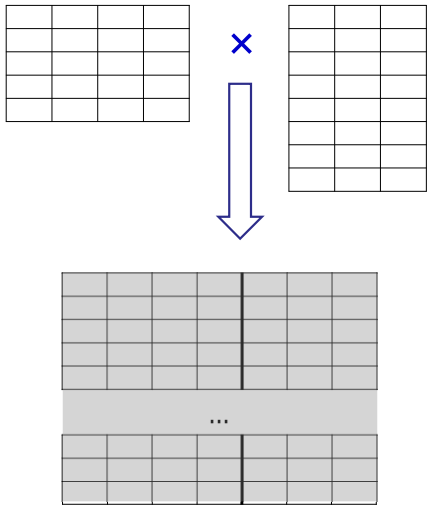
Intersection ($\cap$)



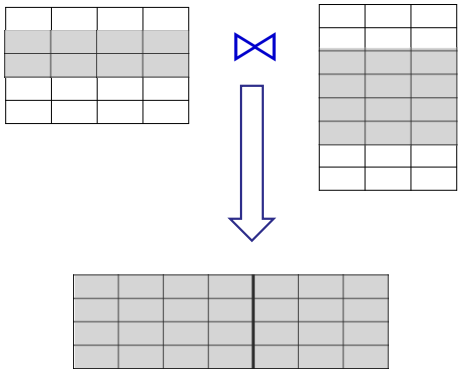
Difference (-)



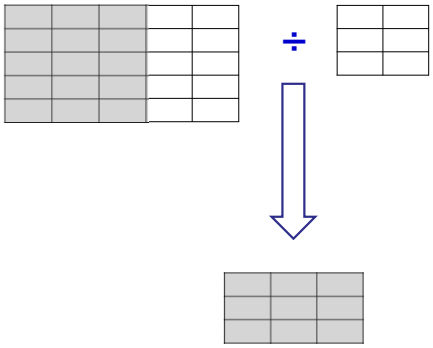
Cartesian product ($\times$)



Join ($\bowtie$)



Division ($\div$)



Selection

- Return all tuples that satisfy a condition
- Notation: $\sigma_C(R)$
  - C: boolean condition that output tuples should satisfy
    - =, ≠, <, >, ≥, ≤, AND, OR, NOT, …
  - R: input relation
  - Output schema: same as input schema (i.e., R's schema)
- Selection σ is commutative

  $\sigma_{C1}(\sigma_{C2}(R)) = \sigma_{C2}(\sigma_{C1}(R))$
  - A sequence of select operations may be applied in any order
  - A cascade of select operations may be replaced by a single selection with a conjunction of all the conditions

Selection example

$\sigma_{\text{Age} > 22}(\text{Employee})$

EID	Name	Age	Department
6	Ann	24	SD
25	John	23	IT

$\sigma_{\text{Age} > 22}$

Employee

EID	Name	Age	Department
19	Joe	22	OM
6	Ann	24	SD
3	Paul	21	OS
25	John	23	IT
5	Peter	21	SD

Projection

- Return specific attributes of all tuples
- Notation: $\pi_{A1, \ldots, An}(R)$
  - Input schema: R(B1, …, Bm)
  - A1, …, An: list of attributes to project onto, s.t. {A1, …, An} ⊆{B1, …, Bm}
  - Output schema: S(A1, …, An)
- The project operation *removes any duplicate tuples* (the result of project is a relation, i.e., a set of tuples)
- Projection is *not* commutative

Projection example

$\pi_{\text{Name, Department}}(\text{Employee})$

Name	Department
Joe	OM
Ann	SD
Paul	OS
John	IT

$\pi_{\text{Name, Department}}$

Employee

EID	Name	Age	Department
19	Joe	22	OM
6	Ann	24	SD
3	Paul	21	OS
25	John	23	IT
5	Ann	21	SD

Cartesian product

- Used to combine tuples from two relations in a combinatorial fashion
- Return the concatenation of every tuple in R_1 with every tuple in R_2
- Notation: $R_1 \times R_2$
  - Input schemas: $R_1(A1,\ldots,An)$, $R_2(B1,\ldots,Bm)$
  - Condition: $\{A1,\ldots,An\} \cap \{B1,\ldots,Bm\} = \emptyset$
  - Output schema: $S(A1,\ldots,An,B1,\ldots,Bm)$
- If R_1 has n_{R1} tuples and R_2 has n_{R2} tuples, then $R_1 \times R_2$ will have $n_{R1} * n_{R2}$ tuples
- Very rare in practice; mainly used to express joins

Cartesian product example

Employee×Department

EID	Name	Age	Department	DID	DName	Address
19	Joe	22	OM	SD	Software Development	Addr_1
6	Ann	24	SD	OM	Online Marketing	Addr_2
19	Joe	22	OM	OS	Online Sales	Addr_3
6	Ann	24	SD	SD	Software Development	Addr_1
19	Joe	22	OM	OM	Online Marketing	Addr_2
6	Ann	24	SD	OS	Online Sales	Addr_3

×

Employee

EID	Name	Age	Department
19	Joe	22	OM
6	Ann	24	SD

Department

DID	DName	Address
SD	Software Development	Addr_1
OM	Online Marketing	Addr_2
OS	Online Sales	Addr_3

Union

- Return the union of all the tuples in R_1 and R_2
- Notation: $R_1 \cup R_2$
  - Input schemas: R_1 and R_2 have the same schema, with attributes A1,…,An
    - i.e. *union-compatible*
  - Output schema: the same as the input relations
- Union is commutative and associative

Union example

Group1 ∪ Group2

GroupID	Lead
5009	Per
6002	Peter
3001	John
5045	Joe
4000	Ann

∪

Group1

GroupID	Lead
5045	Joe
4000	Ann
6002	Peter

Group2

GroupID	Lead
5009	Per
6002	Peter
3001	John

Difference

- Return the the tuples in R_1 that are not in R_2
- Notation: $R_1 – R_2$ (or $R_1 \backslash R_2$)
  - Input schemas: R_1 and R_2 are union-compatible
  - Output schema: the same as the input relations
- Difference is *not* commutative, i.e., in general $R – S \neq S – R$

Difference example

Group1 – Group2

GroupID	Lead
5045	Joe
4000	Ann

–

Group1

GroupID	Lead
5045	Joe
4000	Ann
6002	Peter

Group2

GroupID	Lead
5009	Per
6002	Peter
3001	John

Renaming

- Return the same relation instance with the attributes renamed
  - Changes the schema, not the instance
- Notation: $\rho_{B1,\ldots,Bn}(R)$
  - Input schema: R(A1,…,An)
  - Output schema: R(B1,…,Bn)
- Another notation: $\rho_{\{Ai \rightarrow Bi\}}(R)$
- Useful when a query requires multiple operations
- Necessary in some cases (e.g., joins)

Renaming example

$\rho_{\text{PID, PName, PAge, PDpt}}(\text{Employee})$

PID	PName	PAge	PDpt
19	Joe	22	OM
6	Ann	24	SD
25	John	23	IT
5	Peter	21	SD

$\rho_{\text{PID, PName, PAge, PDpt}}$

Employee

EID	Name	Age	Department
19	Joe	22	OM
6	Ann	24	SD
25	John	23	IT
5	Peter	21	SD

Intersection

- Return the intersection of tuples in R_1 and R_2
- Notation: $R_1 \cap R_2$
  - Input schemas: R_1 and R_2 are union-compatible
  - Output schema: the same as the input relations
- Intersection is commutative and associative
- Intersection is derived
  - $R_1 \cap R_2 = R_1 - (R_1 - R_2)$

Intersection example

Group1 ∩ Group2

GroupID	Lead
3001	John
6002	Peter

∩

Group1

GroupID	Lead
3001	John
4000	Ann
6002	Peter

Group2

GroupID	Lead
5009	Per
6002	Peter
3001	John

Join

- One of the most important and well-studied operations for any relational database with more than a single relation
- Allows to *combine* related tuples from various relations
- The sequence of *cartesian product* followed by *selection* is often used to identify and select related tuples from various relations
  - Join combines this sequence into a single operation
- Comes in various flavors
  - Theta join
  - Natural join
  - Equi-join
  - Semi-join
  - Inner join
  - Outer join
  - Anti-join
  - …

Theta join

- The general case of join operation
- Return all the combinations of R_1 and R_2 tuples which satisfy the join condition θ
- Notation: $R_1 \bowtie_\theta R_2$
  - Equivalent expression: $\sigma_\theta(R_1 \times R_2)$
  - Input schemas: R_1(A1,…,An) and R_2(B1,…,Bm)
  - Condition θ: a boolean condition on A1,…,An,B1,…,Bm
  - Output schema: S(A1,…,An,B1,…,Bm)
- If R_1 has n_{R1} tuples and R_2 has n_{R2} tuples, then $R_1 \bowtie_\theta R_2$ will generally have less than $n_{R1} * n_{R2}$ tuples
  - Only related tuples (based on the join condition) will appear in the result

Theta Join example

$$\text{Employee} \bowtie_{\text{Department=DID} \wedge \text{Age=21}} \text{Department}$$

EID	SName	Age	Department	DID	DeptName	Address
19	Joe	21	OM	OM	Online Marketing	Addr_2
3	Paul	21	OS	OS	Online Sales	Addr_3

$\bowtie_{\text{Department=DID} \wedge \text{Age=21}}$

Employee

EID	SName	Age	Department
19	Joe	21	OM
6	Ann	24	SD
3	Paul	21	OS

Department

DID	DeptName	Address
SD	Software Development	Addr_1
OM	Online Marketing	Addr_2
OS	Online Sales	Addr_3

Equi-join

- The most common use of join involves join conditions with *equality comparisons* ("=") only
- Return all the combinations of tuples of R_1 and R_2 which satisfy the equality condition C=D
- Notation: $R_1 \bowtie_{C=D} R_2$
  - Equivalent expression: $\sigma_{C=D}(R_1 \times R_2)$
  - Input schemas: R_1(A1,…,An) and R_2(B1,…,Bm)
  - C⊆{A1,…,An} and D⊆{B1,…,Bm}
  - Output schema: S(A1,…,An,B1,…,Bm)
- Equi-join is a special case of theta join
- Most used variant of join in practice

Equi-join example

Employee $\bowtie_{\text{Department=DID}}$ Department

EID	Name	Age	Department	DID	DeptName	Address
19	Joe	21	OM	OM	Online Marketing	Addr_2
6	Ann	24	SD	SD	Software Development	Addr_1
3	Paul	21	OS	OS	Online Sales	Addr_3

$\bowtie_{\text{Department=DID}}$

Employee

EID	Name	Age	Department
19	Joe	21	OM
6	Ann	24	SD
3	Paul	21	OS

Department

DID	DeptName	Address
SD	Software Development	Addr_1
OM	Online Marketing	Addr_2
OS	Online Sales	Addr_3

Natural Join

- Return all the combinations of tuples of R_1 and R_2 which agree on the *join attributes*
- Notation: $R_1 \bowtie R_2$
  - Input schemas: $R_1(A1,\ldots,An)$ and $R_2(B1,\ldots,Bm)$
  - Join attributes: $\{A1,\ldots,An\} \cap \{B1,\ldots,Bm\}$
  - Output schema: $S(C1,\ldots,Cp)$ s.t. $\{C1,\ldots,Cp\}=\{A1,\ldots,An\} \cup \{B1,\ldots,Bm\}$
- Natural join is a special case of equi-join
- Natural join allows the combination of relations that are associated by a foreign key

Natural join example

Employee ⋈ Department

$$= \pi_{EID,Name,Age,DID,DeptName,Address}(\sigma_{DID=DID2}(Employee \times \rho_{DID2,DeptName,Address}(Department)))$$

EID	Name	Age	DID	DeptName	Address
19	Joe	21	OM	Online Marketing	Addr_2
6	Ann	24	SD	Software Development	Addr_1
3	Paul	21	OS	Online Sales	Addr_3

⋈

Employee

EID	Name	Age	DID
19	Joe	21	OM
6	Ann	24	SD
3	Paul	21	OS

Department

DID	DeptName	Address
SD	Software Development	Addr_1
OM	Online Marketing	Addr_2
OS	Online Sales	Addr_3

A complete set of relational operations

- {σ, π, $\times$, $\cup$, -, ρ} is a complete set of relational operations

- Any other relational operation can be expressed as a *sequence of operations form the above set*

Relational algebra query example

Person(PID, Name, Age)

Book(BID, Title, AuthorID, Year)

What are the titles of all books whose authors are over 22?

$$\pi_{\text{Title}}(\sigma_{\text{Age}>22}(\text{Person} \bowtie_{\text{PID}=\text{AuthorID}} \text{Book}))$$