CS2102: Database Systems

Relational Algebra

Announcement

Announcement

Assignments

Updated Assignment Deadlines

#	Components	Due Week	Day	Time
1	SQL	*Week 6.5* *(recess week)*	Saturday	12:00 *(noon)*
2	Refinement	*Week 14* *(reading week)*	Monday	12:00 *(noon)*

Note

- Assignment 2 deadline is on reading week because the original week 12 is well being day
  - We keep it "free of work" for week 12
- L00 Overview online slide has been updated
- We have also updated the Course Syllabus page

Announcement

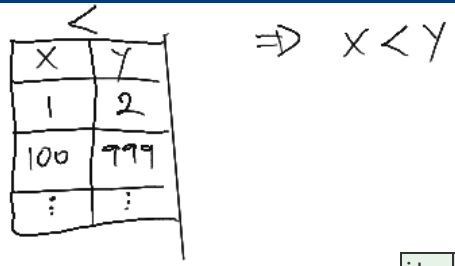
Project

Project Group Formation

- Opens now on Canvas > People > Groups > Project
  - Self sign-up
  - Can be from different tutorial group
  - You may advertise on Canvas discussion or telegram
- Close on Week 3, Saturday, 28 January 2023 at 12:00 noon
  - Afterwards, we will randomly allocate people to groups and/or merge groups to form groups of at least 3 people

Recap

Recap



Relational Model

Basic Concept: **Relations**

- Unified representation of all data using tables with rows and tables
- Definitions
  - **Domains** *Possible values of an attribute*
    - NULL *Unknown/invalid values*
  - **Tuples** *A single row on the table*
  - **Relations** *Set of tuples (all rows)*

Structural Integrity Constraints

- Condition that restricts what constitutes valid data
  - **Domain** *(what type can be inserted?)*
  - **Key** *(which column(s) identifies others?)*
  - **Foreign Key** *(does it exist somewhere else?)*

Table "Movies"

id	title	genre	opened
101	Aliens	Action	1986
102	Logan	Drama	2017
⋮	⋮	⋮	⋮

Table "Cast"

movie_id	actor_id	role
101	20	Ellen Ripley
102	21	Logan
⋮	⋮	⋮

Table "Actors"

id	name	dob
20	Sigourney Weaver	08-10-1949
21	Hugh Jackman	12-10-1968
⋮	⋮	⋮

Recap

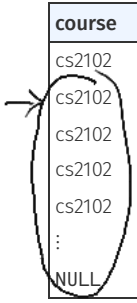
(Structural) Integrity Constraints

Possible Misconception

- *(Foreign)* key constraints are not an intrinsic property of a relation
  - Constraints are specified by the database designer to define what constitutes valid data

Example

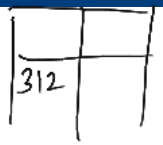
- Without any key constraints, the relation on the right is perfectly valid *(DBMS does not complain)*
  - However, the meaning is problematic from the application perspective
    (e.g., CS2102 gives different credits, with and without exam?)
- ***Goal:*** Avoid different values for "mc" and "exam" for the same course
  (i.e., "course" must uniquely identify both "mc" and "exam")
  - How? Pick *{course}* as the primary key



course	mc	exam
cs2102	2	yes
cs2102	2	no
cs2102	2	NULL
cs2102	4	yes
cs2102	4	no
⋮	⋮	⋮
NULL	NULL	NULL

Recap

Relational Model
Integrity
- Misconception
- Foreign Key



(313, NULL)

(Structural) Integrity Constraints

Foreign Key Condition

Each foreign key in R_1 must satisfy one of the following:

- Appear as ***primary key*** in R_2
- Be a NULL value *(or a tuple containing at least one NULL value)*

Specification

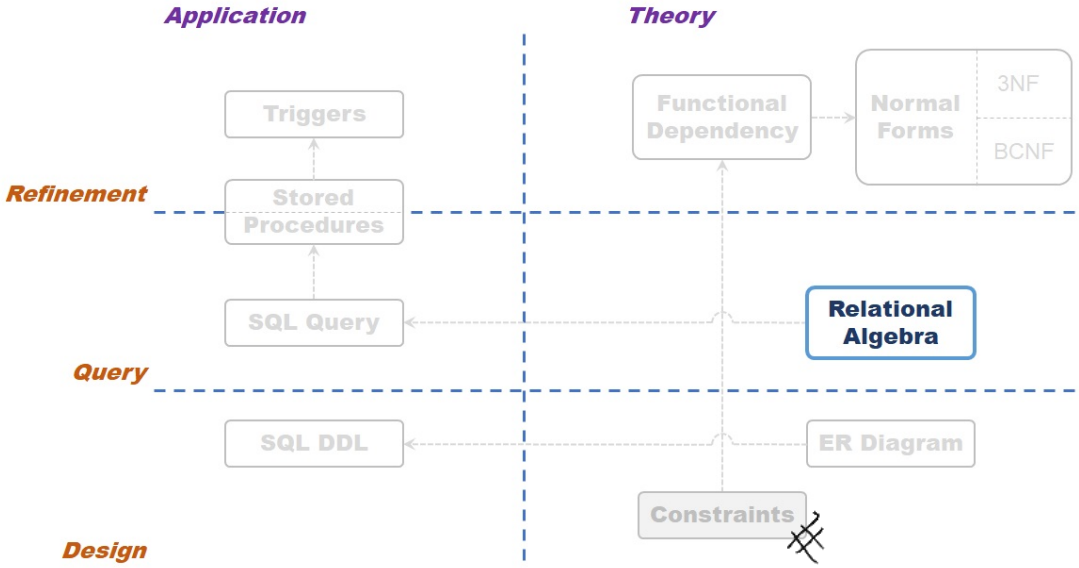
A referential constraint is satisfied if one of the following conditions is true, depending on the <match option> specified in the <referential constraint definition>:

- If no <match type> was specified then, for each row R1 of the referencing table, either at least one of the values of the referencing columns in R1 shall be a null value, or the value of each referencing column in R1 shall be equal to the value of the corresponding referenced column in some row of the referenced table.

- If MATCH FULL was specified then, for each row R1 of the referencing table, either the value of every referencing column in R1 shall be a null value, or the value of every referencing column in R1 shall not be null and there shall be some row R2 of the referenced table such that the value of each referencing column in R1 is equal to the value of the corresponding referenced column in R2.

- at least one of the values of the referencing columns in *R1* shall be a NULL value, or
- the value of each referencing column in *R1* shall be equal to the value of the corresponding referenced column in some row of the referenced table

Roadmap


Application
Theory
Triggers
Functional Dependency
Normal Forms
3NF
BCNF
Refinement
Stored Procedures
SQL Query
Relational Algebra
Query
SQL DDL
ER Diagram
Constraints
Design

Roadmap

Overview

Overview

Relational Algebra

- Preliminary
- Closure

Three-Valued Logic

- Logical
- Relational
- Arithmetic

Basic Operators

- Unary Operators
- Set Operators

Join Operators

- Inner Joins
- Outer Joins

Complex Expressions

Relational Algebra

Relational Algebra

1 + 2 + × −

Preliminary

Algebra

An **algebra** is a mathematical system consisting of:

- **Operands** Variables or values from which new values can be constructed
- **Operators** Symbols denoting procedures that construct new values from the given values

Relational Algebra

- **Operands** *Relations (or variables representing relations)*
- **Operators** Transformation from one or more input relations into *one* output relation

one arg

Operators

- **Unary** selection (σ), projection (π), renaming (ρ)
- **Binary** cross product (×), union (∪), intersection (∩), difference (−)

two args

1

Note

All other operators* can be expressed using these basic operators**.

**In fact, intersection can be constructed using union and difference.
*Except for "aggregate" and "sorting" operations.

Relational Algebra

Preliminary

Why Algebra?

- Allows for *query optimization**
- Rewrite a query such that:
  - It is *equivalent* to the original query
  - It can be evaluated faster than the original query

Example

> **Question**
>
> Which of the following operation is "faster"?

A. (a × b) + (a × c)
B. a × (b + c)

> **Quiz #1**
>
> Is it equivalent?

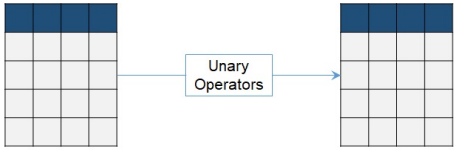
*While we do not penalize for inefficient queries, it is useful to know.

Relational Algebra

Closure Property

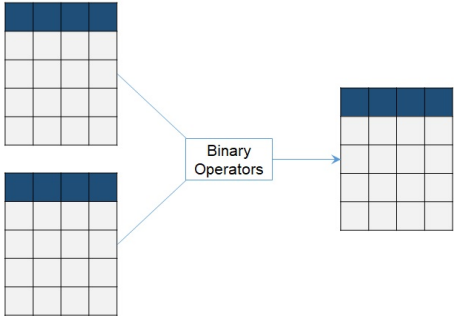
Definition

We say that a *set* of values is **closed** under the set of operators if any combination of the operators *produces only values in the given set*.



Example

- $\mathbb{Z}^+$ is closed under {+}
  - but not under {+, –}
- $\mathbb{Z}$ is closed under {+, –, x}
  - but not under {+, –, x, /}
- $\mathbb{R}$ is closed under {+, –, x, /}*
  - but not under {+, –, x, /, √}
- $\mathbb{C}$ is *algebraically* closed



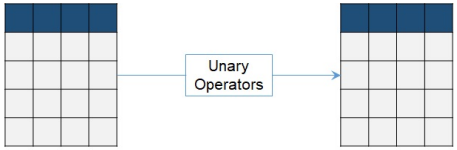
*Division operation (~~i.e., /~~) should not be a division by zero.

Relational Algebra

Closure Property

Definition

We say that a *set* of values is **closed** under the set of operators if any combination of the operators *produces only values in the given set*.



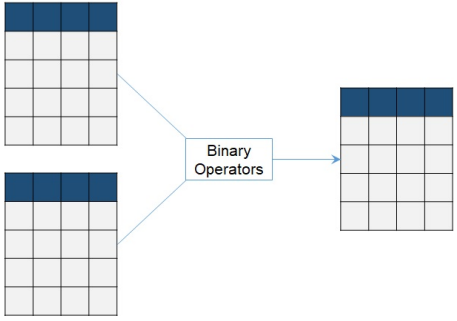
Theorem

Relations are *closed* under Relational Algebra.

Proof

- Inputs are relations
- Outputs are relations

⇒ Output of one relation → input to another relation
⇒ No other output results are possible



Relational Algebra

(a x b) ⊕ (a x c)

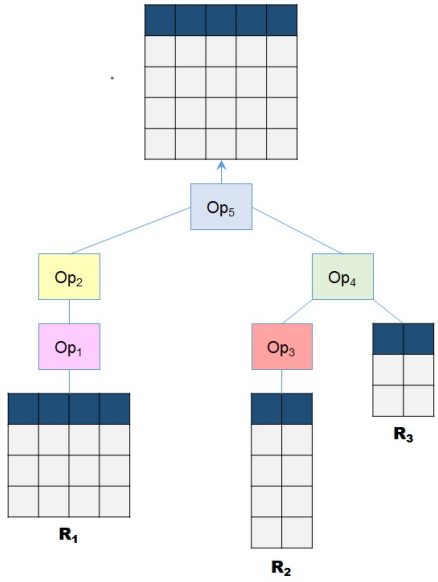
Closure Property

Implication

> We can chain *operations* because all inputs and outputs are ***relations*** *(output of one operators can be input of another operators)*.

Example

> Using the image on the right we get
>
> Op_5 (
> Op_2 (Op_1 (R_1)),
> Op_4 (
> Op_3 (R_2),
> R_3
>)
>)



Relational Algebra

Example Data

Schema

> A simplified company schema
>
> *Employees(<u>name: **TEXT**</u>, age: **INT**, role: **TEXT**)*
> *Managers(<u>name: **TEXT**</u>, office: **TEXT**)*
> *Teams(<u>ename: **TEXT**, pname: **TEXT**</u>, hours: **INT**)*
> *Projects(<u>name: **TEXT**</u>, manager: **TEXT**, start_year: **INT**, end_year: **INT**)*

Foreign Key Constraints

- *(Manager.name)* ⇝ *(Employees.name)*
- *(Teams.ename)* ⇝ *(Employees.name)*
- *(Teams.pname)* ⇝ *(Projects.name)*
- *(Projects.manager)* ⇝ *(Manager.name)*

Relational Algebra

Example Data

Instance

Table "Teams"

ename	pname	hours
Sarah	BigAI	10
Sam	BigAI	5
Bill	BigAI	15
Judy	GlobalDB	20
Max	GlobalDB	5
Sarah	GlobalDB	10
Emma	GlobalDB	35
Max	CoreOS	40
Bill	CoreOS	30
Sam	CoolCoin	40
Sarah	CoolCoin	25
Emma	CoolCoin	10

Table "Projects"

name	manager	start_year	end_year
BigAI	Judy	2020	2025
FastCash	Judy	2018	2025
GlobalDB	Jack	2019	2023
CoreOS	Judy	2020	2020
CoolCoin	Jack	2015	2020

Table "Employees"

name	age	role
Sarah	25	dev
Judy	35	sales
Max	52	dev
Marie	36	hr
Sam	30	sales
Bernie	19	NULL
Emma	28	dev
Jack	40	dev
Bill	45	dev

Table "Managers"

name	office
Judy	#03-20
Jack	#03-10

L02.tbl

Boolean : { True, False, NULL }

Three-Valued Logic

Three-Valued Logic

Logical
- Truth Table
- Alternate
Relational &
Arithmetic

Logical Operations

Truth Table

c_1	c_2	$c_1 \wedge c_2$	$c_1 \vee c_2$	$\neg c_1$
False	False	False	False	True
False	NULL	False	NULL	True
False	True	False	True	True
NULL	False	False	NULL	NULL
NULL	NULL	NULL	NULL	NULL
NULL	True	NULL	True	NULL
True	False	False	True	False
True	NULL	NULL	True	False
True	True	True	True	False

Three-Valued Logic

Logical
- Truth Table
- Alternate
Relational &
Arithmetic

Logical Operations

Alternate Truth Table

Conjunction

$c_1 \wedge c_2$		c_1		
		False	NULL	**True**
c_2	**False**	False	False	False
	NULL	False	NULL	NULL
	True	False	NULL	True

Disjunction

$c_1 \vee c_2$		c_1		
		False	NULL	**True**
c_2	**False**	False	NULL	True
	NULL	NULL	NULL	True
	True	True	True	True

Three-Valued Logic

NULL + n → NULL

Logical
Relational & Arithmetic

Relational & Arithmetic Operations

<>

Basic Relational & Arithmetic

Any relational/arithmetic operation with **NULL** produces **NULL** values

Operation	Result	Operation	Result
NULL < 2023	NULL	NULL = NULL	NULL
NULL ≥ 0	NULL	NULL <> NULL	NULL

NULL as Values

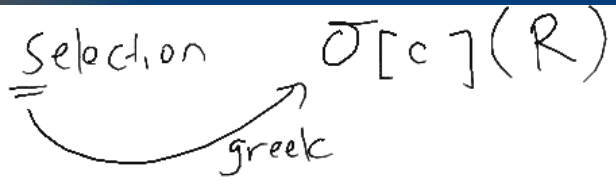
We add operators to treat **NULL** as values

≡ ≢

V_1	V_2	$V_1 \equiv V_2$	$V_1 \not\equiv V_2$
NULL	NULL	True	False
V_1	NULL	False	True
NULL	V_2	False	True
V_1	V_2	**$V_1 = V_2$**	**$V_1 <> V_2$**

Basic Operators

Basic Operators



Selection

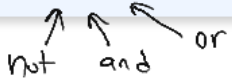
Syntax

$\sigma_{[c]}(R)$ where c is a condition that returns a boolean value:

	Expr	Example	Name
1	(expr)	$\sigma_{[(start\_year=2020)]}(Projects)$	***precedence***
2	attr **op** const	$\sigma_{[start\_year=2020]}(Projects)$	***constant selection***
2	$attr_1$ **op** $attr_2$	$\sigma_{[start\_year=end\_year]}(Projects)$	***attribute selection***
3	¬ expr	$\sigma_{[\neg(start\_year=2020)]}(Projects)$	***negation***
4	$expr_1 \wedge expr_2$	$\sigma_{[start\_year=2020 \wedge manager='Judy']}(Projects)$	***conjunction***
5	$expr_1 \vee expr_2$	$\sigma_{[start\_year=2020 \vee manager='Judy']}(Projects)$	***disjunction***

with **op** ∈ {=, <>, <, ≤, ≥, >, ≡, ≢}.

Operator Precedence: (), **op**, ¬, ∧, ∨*



*Without the parentheses and **op**, this is often called Disjunctive Normal Form.

Basic Operators

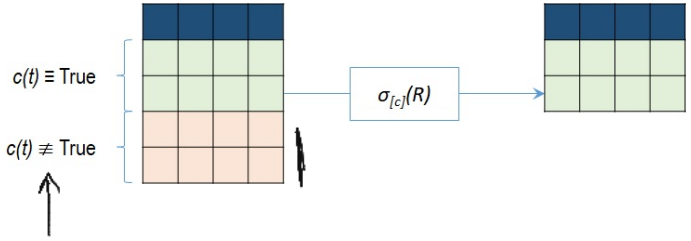
Accept Reject

Selection

Semantics

$\sigma_{[c]}(R)$ selects all tuples from a relation R *(i.e., rows from a table)* that satisfy the selection condition c.

Visualization



Properties

- The result have the same schema as the input relation
- The number of rows are often smaller

[#]We can always "rearrange" the rows such that the visual above works because we are working with a ***set*** of tuples.

Basic Operators

Selection

Example

Find all projects where Judy is the manager.

1) Look at your data
→ find table(s) that contains data
2) Figure out the condition
3) Find
fails
counter-example

$T1 := \sigma_{(manager = 'Judy')}(Projects)$

name	manager	start_year	end_year
"BigAI"	"Judy"	2020	2025
"FastCash"	"Judy"	2018	2025
"CoreOS"	"Judy"	2020	2020

3 Rows

Projects

name	manager	start_year	end_year
"BigAI"	"Judy"	2020	2025
"FastCash"	"Judy"	2018	2025
"GlobalDB"	"Jack"	2019	2023
"CoreOS"	"Judy"	2020	2020
"CoolCoin"	"Jack"	2015	2020

5 Rows

Basic Operators

Selection

$$\overline{\sigma}[(role \neq \text{'Sales'} \wedge age \leq 30) \vee role \equiv \text{'HR'}](Emps)$$

Exercise #1

Find all employees that (a) do not work in Sales and are younger than 30 or (b) work in HR.

Basic Operators

Projection

Syntax

$\pi_{[\ell]}(R)$ where ℓ is *an* ***ordered*** *list* of attributes

Note:

- For simplicity, we do not allow the following on ℓ:
  - *Operations* *(e.g., $\pi_{[A_1 + A_2]}(R)$)*
  - *Duplicate* *(e.g., $\pi_{[A_1, A_1]}(R)$)*
- The ordering of the attribute matters *(i.e., $\pi_{[A_1, A_2]}(R) \neq \pi_{[A_2, A_1]}(R)$)*

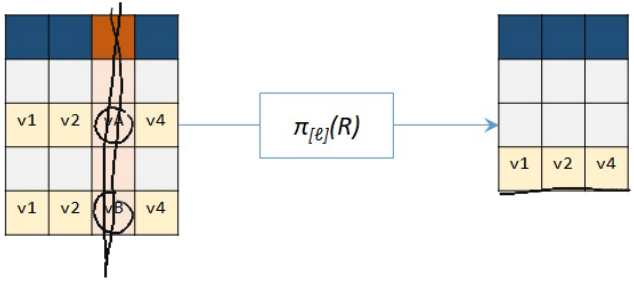
Basic Operators

Projection

Semantics

$\pi_{[\ell]}(R)$ keeps only the column *(i.e., projects)* specified in the ordered list ℓ and in the same order.

Visualization



Properties

- The resulting schema is specified by ℓ
- The number of rows may be smaller
  - Because relation is defined as a ***set*** of tuples

Basic Operators

Projection

Example

Find all project names that have team members.

$T1 := $ $\pi_{[pname]}$(Teams)

pname
"BigAI"
"GlobalDB"
"CoreOS"
"CoolCoin"

4 Rows

Teams

ename	pname	hours
"Sarah"	"BigAI"	10
"Sam"	"BigAI"	5
"Bill"	"BigAI"	15
"Judy"	"GlobalDB"	20
"Max"	"GlobalDB"	5
"Sarah"	"GlobalDB"	10
"Emma"	"GlobalDB"	35
"Max"	"CoreOS"	40
"Bill"	"CoreOS"	30
"Sam"	"CoolCoin"	40
"Sarah"	"CoolCoin"	25
"Emma"	"CoolCoin"	10

12 Rows

Basic Operators

Projection

Exercise #2

Which of the following relational algebra expression resulted in the output below?

name	manager	start_year	end_year
BigAI	Judy	2020	2025
FastCash	Judy	2018	2025
GlobalDB	Jack	2019	2023
CoreOS	Judy	2020	2020
CoolCoin	Jack	2015	2020

manager	name
Judy	FastCash
Jack	GlobalDB
Jack	CoolCoin

	Choice	Comment
A	$\sigma_{[start\_year \leq 2019]}(\pi_{[manager,name]}(Projects))$	
B	$\pi_{[manager,name]}(\sigma_{[start\_year < 2020]}(Projects))$	
C	$\pi_{[manager,name]}(\sigma_{[manager='Jack']}(Projects))$	
D	$\pi_{[name,manager]}(\sigma_{[start\_year \leq 2019]}(Projects))$	

Basic Operators

Renaming

Syntax

$\rho_{[\mathfrak{R}]}(R)$ where $\mathfrak{R}$ is *a collection* of attribute renaming which is *either*:

1. $\mathfrak{R}$ is an ***unordered*** collection of $B_i \leftarrow A_i$
   - **Example:** $\rho_{[B_1 \leftarrow A_1,\ B_2 \leftarrow A_2]}(R)$
   - The order of attributes does not matter
2. $\mathfrak{R}$ is an ***ordered*** list of attributes
   - **Example:** $\rho_{(B_1,\ B_2,\ ...,\ B_n)}(R)$
   - The order of attributes matters
   - The list must specify **all** attributes in the original schema
     - If not renamed, set $B_i = A_i$

new ← old

We will follow ***option (1)*** *(unordered collection of arrows (new ← old))*.

Note:

- Renaming will be relevant for set and join operations

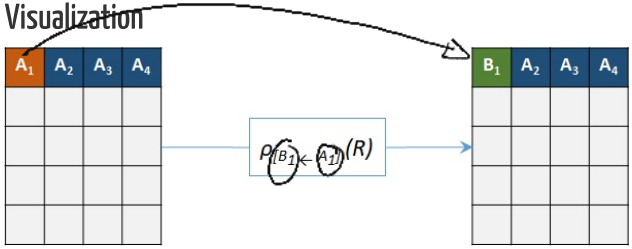
Basic Operators

Renaming

Semantics

$\rho_{[\mathfrak{R}]}(R)$ renames the attributes listed in $\mathfrak{R}$.

Visualization



Properties

- The resulting schema is modified by $\mathfrak{R}$
- The order columns is unchanged *(modulo renaming)*
- The number of rows remains the same

Basic Operators

Renaming

Example

Rename the teams such that the project name becomes simple "project".

$T1 := $\rho_{[pname \leftarrow project]}(Teams)$

ename	project	hours
"Sarah"	"BigAI"	10
"Sam"	"BigAI"	5
"Bill"	"BigAI"	15
"Judy"	"GlobalDB"	20
"Max"	"GlobalDB"	5
"Sarah"	"GlobalDB"	10
"Emma"	"GlobalDB"	35
"Max"	"CoreOS"	40
"Bill"	"CoreOS"	30
"Sam"	"CoreOS"	40
"Sarah"	"CoolCoin"	25
"Emma"	"CoolCoin"	10

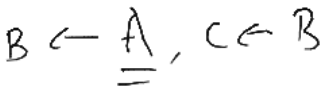
12 Rows

Teams

ename	pname	hours
"Sarah"	"BigAI"	10
"Sam"	"BigAI"	5
"Bill"	"BigAI"	15
"Judy"	"GlobalDB"	20
"Max"	"GlobalDB"	5
"Sarah"	"GlobalDB"	10
"Emma"	"GlobalDB"	35
"Max"	"CoreOS"	40
"Bill"	"CoreOS"	30
"Sam"	"CoolCoin"	40
"Sarah"	"CoolCoin"	25
"Emma"	"CoolCoin"	10

12 Rows

Basic Operators



Caution

Selection

$\sigma_{[c]}(R)$

The condition c must specify only attributes in R

Projection

$\pi_{[\ell]}(R)$

The ordered list of attributes ℓ must specify only attributes in R

Renaming

$\rho_{[\mathfrak{R}]}(R)$

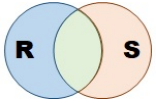
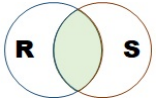
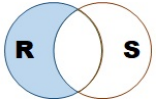
No two different attributes may be renamed to the same name
No attributes may be renamed more than once in a single operation

Set Operators

Set Operators

{(1, 2), (3, 4)} ∪ {('A', 2022-01-01, 3)}

Operations

Operation	Operator	Visualization	Description
Union	R ∪ S		A relation containing all tuples that are in *R* or *S*
Intersection	R ∩ S		A relation containing all tuples that are in *R* and *S*
Difference	R – S		A relation containing all tuples that are in *R* but not in *S*

Problem

What should be the result of union between {1, 2} and {'A', 'B', 'C'}?

Set Operators

Operations

Union Compatibility

Definition

Two relations *R* and *S* are **union-compatible** if it satisfies both of the following:

- *R* and *S* have the *same number of attributes*
- The corresponding attributes have the *same or compatible domains*

Note:

- Just because two relations are union-compatible, does not mean the set operation is meaningful

Example

Union-Incompatible

- *Employees(name:* ***TEXT****, age:* ***INT****, role:* ***TEXT****)*
- *Teams(ename:* ***TEXT****, pname:* ***TEXT****, hours:* ***INT****)*

Union-Compatible

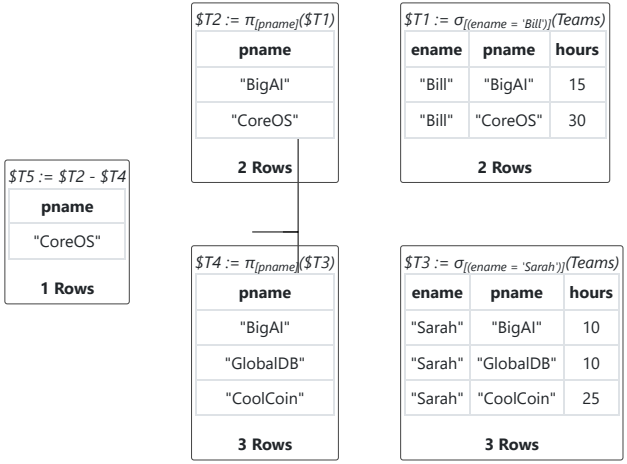
- *Employees(name:* ***TEXT****, role:* ***TEXT****, age:* ***INT****)*
- *Teams(ename:* ***TEXT****, pname:* ***TEXT****, hours:* ***INT****)*

Set Operators

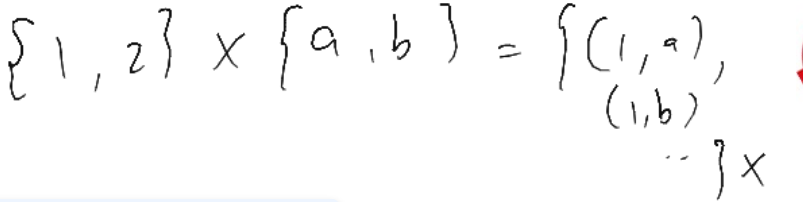
Operations

Example

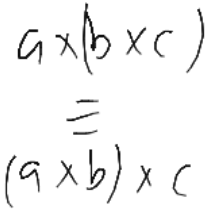
Find all projects that Bill is working on but Sarah is not working on.



Set Operators



Operations
Product
- Definition
- Example
- Exercise
- Discussion



Cross Product

Definition

The **cross product** of two relations ***(R × S)*** is a relation formed by combining all pairs of tuples *(i.e., rows)* from the two input relations.

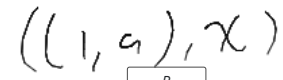
More formally, let $R(A_1, A_2, ..., A_n)$ and $S(B_1, B_2, ..., B_n)$, then

- $R \times S$ produce a schema
  $\boldsymbol{(R \times S)(A_1, A_2, ..., A_n, B_1, B_2, ..., B_n)}$
- $\boldsymbol{R \times S = \{(a_1, a_2, ..., a_n, b_1, b_2, ..., b_n)}$
  ~~$\boldsymbol{| (a_1, a_2, ..., a_n) \in R \times (b_1, b_2, ..., b_n) \in S\}}$~~

Note:

- The set of attributes in R and S must be ***disjoint***
  $(Attr(R) \cap Attr(S) = \emptyset)$, where
  $Attr(R)$ is the set of attributes in R
  $Attr(S)$ is the set of attributes in S

Example



$T1 := R × S

A	B	C	X	Y
1	5	"m"	"a"	30
1	5	"m"	"b"	10
1	5	"m"	"c"	20
2	3	"f"	"a"	30
2	3	"f"	"b"	10
2	3	"f"	"c"	20

6 Rows

R

A	B	C
1	5	"m"
2	3	"f"

2 Rows

S

X	Y
"a"	30
"b"	10
"c"	20

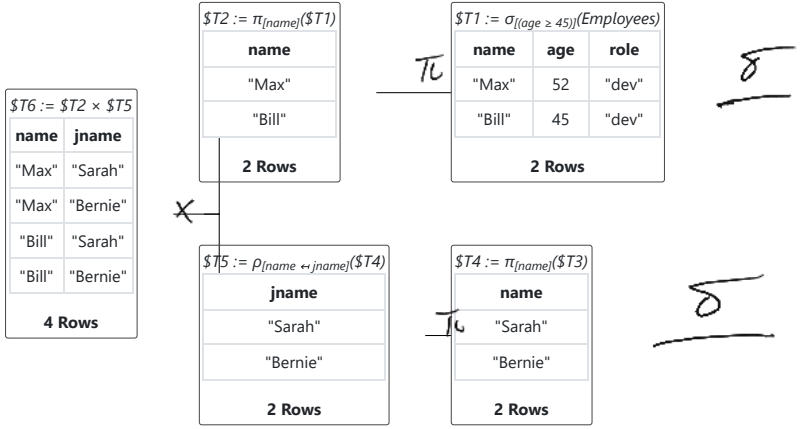
3 Rows

Set Operators

Cross Product

Example

Find all pairs of senior employee names *(i.e., age ≥ 45)* and junior employee names *(i.e., age ≤ 25)*.



Set Operators

Cross Product

Exercise #3

For all the project names, find the offices of the managers.

Set Operators

Join

Operations
Product
- Definition
- Example
- Exercise
- Discussion

Cross Product

Discussion

Observation

- Given two relations *R* and *S*, the size of the cross product is $|R| \times |S|$
- In practice, many to most queries requiring a cross product *also* require
  - Selection operation to remove unnecessary rows
  - Projection to remove unnecessary/duplicate columns *(optional)*

$$\pi_{[name,office]}(\sigma_{[manager=mname]}(Projects \times (\rho_{[mname \leftarrow name]}(Managers))))$$

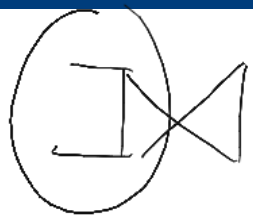
Join Operators

- Simplify relational algebra expression
  - Combine cross product, selection, and *(optionally)* projection
- Avoid generating all $|R| \times |S|$ intermediate tuples

$$R \bowtie_{[\theta]} S$$

Join Operators

Join Operators



Preliminary

Basic Idea

- Combine cross product, selection, and *(optionally)* projection into a single operator
- Typically results in simpler relational algebra expressions when formulating queries

Base Types

Join *(for each element in R, combine with each element in S)*

Inner Join

Includes ***only*** tuples that satisfy the condition

- $\bowtie_{[\theta]}$ **θ-join**
- $\bowtie_{=}$ **equi join**
- $\bowtie$ **natural join**

Outer Join

Also include tuples that do ***not*** satisfy the condition

- $\mathbin{⟕}_{[\theta]}$ **left outer join**
- $\mathbin{⟖}_{[\theta]}$ **right outer join**
- $\mathbin{⟗}_{[\theta]}$ **full outer join**

Join Operators

Inner Joins

θ-Join

Definition

The θ-join $(R \bowtie_{[\theta]} S)$ of two relations R and S is defined as: $R \bowtie_{[\theta]} S = \sigma_{[\theta]}(R \times S)$

Note:

- The condition θ can use attributes that appears in R or S

Example

For all the projects, find the offices of the managers.

$$\text{Projects} \bowtie_{[\text{manager}=\text{mname}]} (\rho_{[\text{mname} \leftarrow \text{name}]}(\text{Managers}))$$

$T2 := Projects ⋈[(manager = mname)] $T1

name	manager	start_year	end_year	mname	office
"BigAI"	"Judy"	2020	2025	"Judy"	"#03-20"
"FastCash"	"Judy"	2018	2025	"Judy"	"#03-20"
"GlobalDB"	"Jack"	2019	2023	"Jack"	"#03-10"
"CoreOS"	"Judy"	2020	2020	"Judy"	"#03-20"
"CoolCoin"	"Jack"	2015	2020	"Jack"	"#03-10"

5 Rows

Join Operators

Inner Joins

Equi Join

Definition

Equi join ($R \bowtie_{=} S$) is a special θ-join where the only relational operator that can be used is equality *(e.g., = or ≡)*

Note:

- θ-Join can use arbitrary relational operator *(e.g., =, <>, <, ≤, >, ≥, ≡, ≢)*
- Dedicated operation because attribute selection using equality operator are the most common
  - Especially when performing inner join following foreign key constraints
  - In many cases, *hash-tables* can be used to increase speed

Join Operators

$R \bowtie S$

Inner Joins

Natural Join

Definition

Let $R(A_1, A_2, ..., A_i, B_1, B_2, ..., B_j)$ and $S(B_1, B_2, ..., B_j, C_1, C_2, ..., C_k)$. We define **natural join** ($R \bowtie S$):

$$R \bowtie S = \{(a_1, a_2, ..., a_i, b_1, b_2, ..., b_j, c_1, c_2, ..., c_k) \mid (a_1, a_2, ..., a_i, b_1, b_2, ..., b_j) \in R \wedge (b_1, b_2, ..., b_j, c_1, c_2, ..., c_k) \in S\}$$

In other words:

- The join is performed over all ***common attributes*** between R and S *(i.e., $(B_1, B_2, ..., B_j)$)*
  - We only combine if the value of the common attributes are ***equal***
  - The output relations contains the common attribute of R and S ***only once*** *(via projection)*

Alternatively, $R \bowtie S = \pi_{[\ell]}(R \bowtie_{[\theta]} S)$, where:

- $\ell = Attr(R) \cup (Attr(S) - Attr(R))$*
- $\theta = \forall A_i \in Attr(R) \cap Attr(S): R_{A_i} = S_{A_i}$

*Equivalent to $Attr(R) \cup Attr(S)$ except that we want to show it more explicitly all attribute in S appears only after R.

Join Operators

Outer Joins

Motivation

- Inner joins *eliminate* all tuples that do not satisfy the condition
  (i.e., selection operation)
- Sometimes the tuple in *R* or *S* that do ***NOT*** match are also of interest
  (i.e., dangling tuple)

> **Example**
>
> Find all employees that are not assigned to any project.

> **Note**
>
> Inner join can only find all employees that are assigned to *at least one project*.

Table "Teams"

ename	pname	hours
Sarah	BigAI	10
Sam	BigAI	5
Bill	BigAI	15
Judy	GlobalDB	20
Max	GlobalDB	5
Sarah	GlobalDB	10
Emma	GlobalDB	35
Max	CoreOS	40
Bill	CoreOS	30
Sam	CoolCoin	40
Sarah	CoolCoin	25
Emma	CoolCoin	10

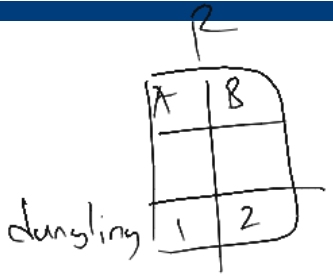
Table "Employees"

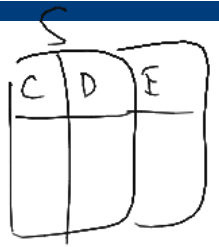
name	age	role
Sarah	25	dev
Judy	35	sales
Max	52	dev
Marie	36	hr
Sam	30	sales
Bernie	19	NULL
Emma	28	dev
Jack	40	dev
Bill	45	dev

Join Operators

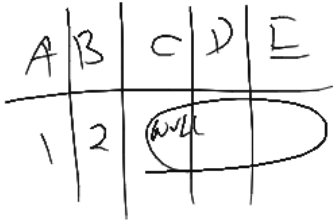
Outer Joins

Informal Steps





1. Perform inner join $M = R \bowtie_\theta S$
2. Add *dangling tuples* to M from:
   - *R* in case of a ***left outer join***
   - *S* in case of a ***right outer join***
   - *R* and *S* in case of a ***full outer join***
3. "Pad" missing attribute values of dangling tuples with `NULL`



Join Operators

Outer Joins

Definition

- $R \ltimes_{[\theta]} S = R \bowtie_{[\theta]} S$
- $R \rtimes_{[\theta]} S = R \bowtie_{[\theta]} S$
- $R \ltimes\!\rtimes_{[\theta]} S = R \bowtie_{[\theta]} S$

$$\cup\ (dangle(R \bowtie_{[\theta]} S) \times \{null(S)\})$$
$$\cup\ (dangle(\{null(S)\} \times R \bowtie_{[\theta]} S))$$
$$\cup\ (dangle(R \bowtie_{[\theta]} S) \times \{null(S)\}) \cup (dangle(\{null(S)\} \times R \bowtie_{[\theta]} S))$$

where

- *dangle*$(R \bowtie S)$ = set of dangling tuples in R with respect to $R \bowtie S$
- *null*(R) = n-component tuple of `NULL` values where n is the number of attributes of R

Example

Find all employees that are not assigned to any project.

Possible Solution

$\sigma_{[ename = \texttt{NULL}]}(Employees \ltimes_{[name=ename]} Teams)$

Note

There are other solutions involving set operations as well.

Join Operators

Outer Joins

Natural Outer Joins

> Analog to natural *(inner)* join
>
> - The condition θ is not specified
> - Equality operation is performed over all attributes that *R* and *S* have in common

Kinds

- **Natural left outer join** $R \ ⟕ \ S$
- **Natural right outer join** $R \ ⟖ \ S$
- **Natural full outer join** $R \ ⟗ \ S$

Join Operators

(Judy, #03-20 / Judy, GlobalDB, 20)
(Jack, #03-10 / NULL, NULL, NULL)

Outer Joins

Exercise #4

Recap the relations "Teams" and "Managers" on the right. How many **row** and **columns** are in the result of the relational algebra expression below?

$\sigma_{[ename \equiv NULL]}(Managers \bowtie_{[name=ename]} Teams)$

Table "Teams"

ename	pname	hours
Sarah	BigAI	10
Sam	BigAI	5
Bill	BigAI	15
Judy	GlobalDB	20
Max	GlobalDB	5
Sarah	GlobalDB	10
Emma	GlobalDB	35
Max	CoreOS	40
Bill	CoreOS	30
Sam	CoolCoin	40
Sarah	CoolCoin	25
Emma	CoolCoin	10

Table "Managers"

name	office
Judy	#03-20
Jack	#03-10

dangling

	Choice	Comment	
A	1 row, 5 columns		?
B	3 rows, 5 columns		?
C	1 row, 3 columns		?
D	3 rows, 3 columns		?

Complex Expressions

Complex Expressions

Motivation

Question

Find all managers *(with their offices)* of projects that started in 2020 or later, where at least one member of the project team has to work 30 hours or more on that project per week.

Complex Expressions

Motivation

Question

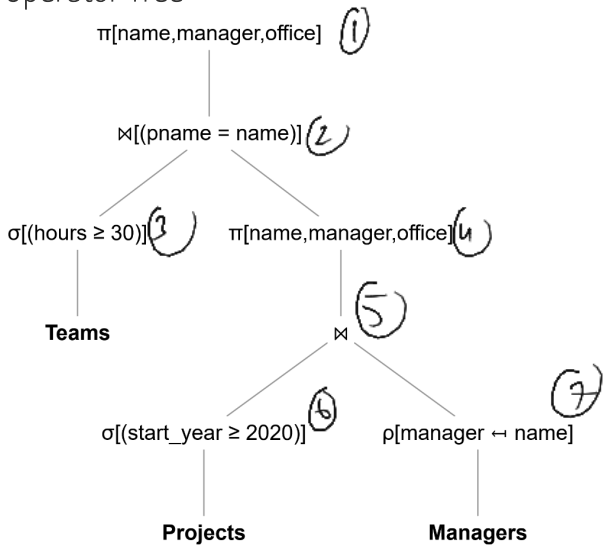
Find all managers *(with their offices)* of projects that started in 2020 or later, where at least one member of the project team has to work 30 hours or more on that project per week.

Possible Solution #1

$M := \rho[manager \leftarrow name](Managers)$
$P := \sigma[start\_year \geq 2020](Projects)$
$T := \sigma[hours \geq 30](Teams)$
$Q_1 := \pi[name, manager, office](P \bowtie M)$
$Q_2 := T \bowtie[pname = name] Q_1$
$Q := \pi[name, manager, office](Q_2)$

```
π[name, manager, office](
  σ[hours ≥ 30](Teams)
    ⋈[pname = name]
  π[name, manager, office](
    σ[start_year ≥ 2020](Projects)
      ⋈
    ρ[manager ← name](Managers)))
```

Operator Tree



Complex Expressions

Motivation

Question

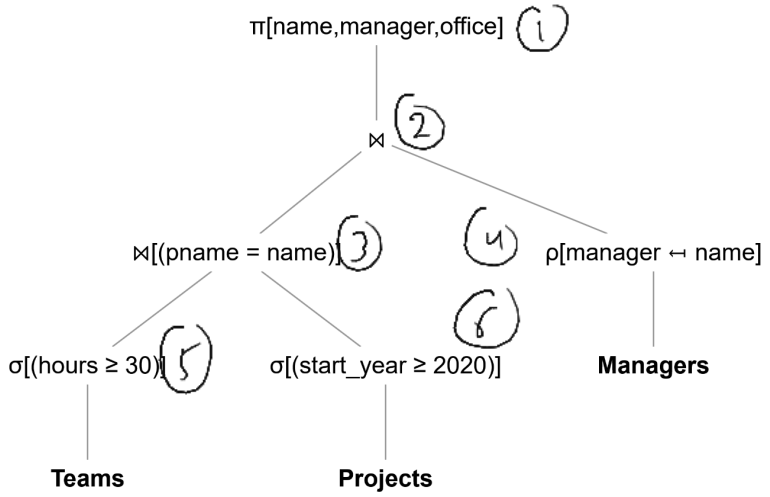
Find all managers *(with their offices)* of projects that started in 2020 or later, where at least one member of the project team has to work 30 hours or more on that project per week.

Possible Solution #2

$M := \rho_{[manager \leftarrow name]}(Managers)$
$P := \sigma_{[start\_year \geq 2020]}(Projects)$
$T := \sigma_{[hours \geq 30]}(Teams)$
$Q_1 := T \bowtie_{[pname = name]} P$
$Q_2 := Q_1 \bowtie M$
$Q := \pi_{[name, manager, office]}(Q_2)$

```
π[name, manager, office](
  σ[hours ≥ 30](Teams)
    ⋈[pname = name]
  σ[start_year ≥ 2020](Projects)
    ⋈
  ρ[manager ← name](Managers)
)
```

Operator Tree



Complex Expressions

Equivalence

Definition

We say that two relational algebra expressions Q_1 and Q_2 are **equivalent** *(denoted by $Q_1 \equiv Q_2$)* if for the any input relations either:

both produces error ***or*** *both produces the same result*

Note:

- This may be called **strongly** equivalent
- **Weakly** equivalent can be defined as if there is no error then both produces the same result

Multiplicity

- In general, there are multiple ways to formulate a query to get the same result
- This includes:
  - Order in which join operations are performed
  - Order in which selection operations are performed *(e.g., before or after join operators)*
  - Inserting additional projections to minimize intermediate results
- **Query optimization:** finding the "best" relational algebra expression *(**CS3223**)*
  - Handled by the DBMS transparent to the user

Complex Expressions

Equivalence

Strong Equivalence *(both cause errors or both return the same result)*

Selection

- $\sigma_{[c_1]}(\sigma_{[c_2]}(R)) \equiv \sigma_{[c_2]}(\sigma_{[c_1]}(R))$
- $\sigma_{[c_1]}(\sigma_{[c_2]}(R)) \equiv \sigma_{[c_1 \wedge c_2]}(R)$

Projection

- $\pi_{\ell_1}(\pi_{\ell_2}(R)) \not\equiv \pi_{\ell_1}(R)$ *(unless $\ell_1 \subseteq \ell_2$)*

Combined

- $\pi_{[\ell]}(\sigma_{[\theta]}(R)) \not\equiv \sigma_{[\theta]}(\pi_{[\ell]}(R))$ *(unless θ uses only attributes in ℓ)*
- $\sigma_{[\theta]}(R \times S) \not\equiv \sigma_{[\theta]}(R) \times S$ *(unless θ uses only attributes in R)*

Cross Product and Joins

- $R \times S \not\equiv S \times R$ *(different column order*)*
- $R \bowtie S \not\equiv S \bowtie R$ *(i.e., non-commutative)*
- $R \times (S \times T) \equiv (R \times S) \times T$
- $R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T$ *(i.e., associative)*
- $R \bowtie_{[\theta_1]} (S \bowtie_{[\theta_2]} T) \not\equiv (R \bowtie_{[\theta_1]} S) \bowtie_{[\theta_2]} T$

  *(if θ_1 uses T **and** θ_2 uses R)*

*You may find some people allow this to be equivalent but we do not.

Complex Expressions

Equivalence

Weak Equivalence *(if both do not cause errors then both return the same result)*

Selection

- $\sigma_{[c_1]}(\sigma_{[c_2]}(R)) \equiv \sigma_{[c_2]}(\sigma_{[c_1]}(R))$
- $\sigma_{[c_1]}(\sigma_{[c_2]}(R)) \equiv \sigma_{[c_1 \wedge c_2]}(R)$

Projection

- $\pi_{\ell_1}(\pi_{\ell_2}(R)) \equiv \pi_{\ell_1}(R)$

Combined

- $\pi_{[\ell]}(\sigma_{[\theta]}(R)) \equiv \sigma_{[\theta]}(\pi_{[\ell]}(R))$
- $\sigma_{[\theta]}(R \times S) \equiv \sigma_{[\theta]}(R) \times S$

Cross Product and Joins

- $R \times S \not\equiv S \times R$ *(different column order*)*
- $R \bowtie S \not\equiv S \bowtie R$ *(i.e., non-commutative)*
- $R \times (S \times T) \equiv (R \times S) \times T$
- $R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T$ *(i.e., associative)*
- $R \bowtie_{[\theta_1]} (S \bowtie_{[\theta_2]} T) \equiv (R \bowtie_{[\theta_1]} S) \bowtie_{[\theta_2]} T$

*You may find some people allow this to be equivalent but we do not.

Complex Expressions

Alternatives

Invalid Relational Expressions

Attribute no longer available after projection

$\sigma_{[\ \textbf{role}\ = \text{'dev'}]}(\pi_{[\ \textbf{name, age}\ ]}(Employees))$

Attribute no longer available after renaming

$\sigma_{[\ \textbf{role}\ = \text{'dev'}]}(\rho_{[\ \textbf{position} \leftarrow \textbf{role}\ ]}(Employees))$

Incompatible attribute types

$\sigma_{[\ \textbf{age = role}\ ]}(Employees)$

We assume that there is no implicit type conversion *(e.g., `INT` to `TEXT`)*

Complex Expressions

Motivation
Observation
Alternatives
- Invalid
- Redundant

Alternatives

Valid but with Redundancy

Cross product + attribute selection ≡ join

$\sigma_{[\ \textbf{manager = mname}\ ]}(Projects\ \times\ \rho_{[mname \leftarrow name]}(Managers))$
$\equiv Projects\ \bowtie_{[\ \textbf{manager = mname}\ ]}\ \rho_{[mname \leftarrow name]}(Managers)$

Unnecessary operators

$\pi_{[\ \textbf{name}\ ]}(\pi_{[\ \textbf{name, age}\ ]}(Employees))$
$\equiv \pi_{[\ \textbf{name}\ ]}(Employees)$

Unoptimized query

$\textbf{σ[start_year = 2020]}\ (Projects \bowtie_{[manager = mname]} \rho_{[mname \leftarrow name]}(Managers))$
$\equiv\ \textbf{(σ[start_year = 2020](Projects))}\ \bowtie_{[manager = mname]} \rho_{[mname \leftarrow name]}(Managers)$

> **Quiz #2**
>
> Can you use *natural join* for the first and last operations?

Summary

Summary

Algebra

What It Is

- Formal method to query relational data
- Closure property for arbitrarily complex relational expressions
- Basis for database query language such as SQL

Operators

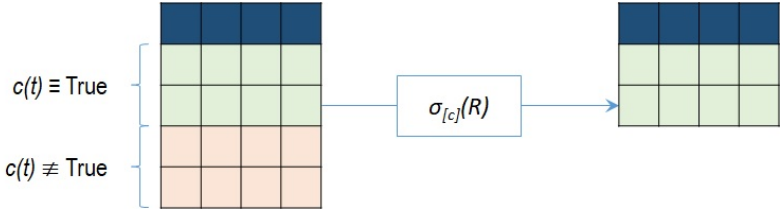
- **Unary** selection (σ), projection (π), renaming (ρ)
- **Binary** cross product (×), union (∪), intersection (∩), difference (−)

Summary

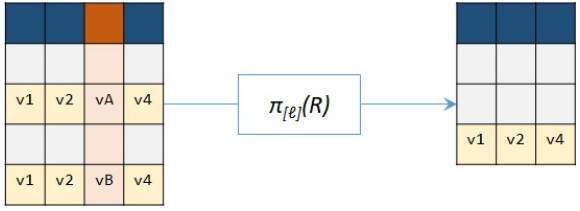
Algebra
Unary
Set
Join

Unary

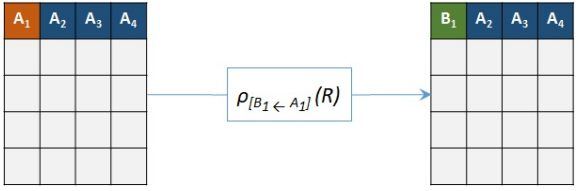
Selection



Projection

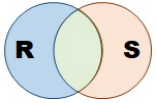
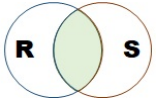
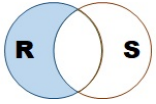


Renaming



Summary

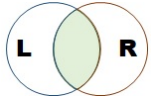
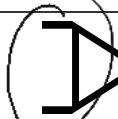
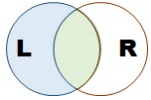
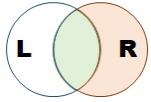
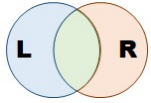
Set

Operation	Operator	Visualization	Description
Union	R ∪ S		A relation containing all tuples that are in *R* or *S*
Intersection	R ∩ S		A relation containing all tuples that are in *R* and *S*
Difference	R – S		A relation containing all tuples that are in *R* but not in *S*

Summary

Algebra
Unary
Set
Join

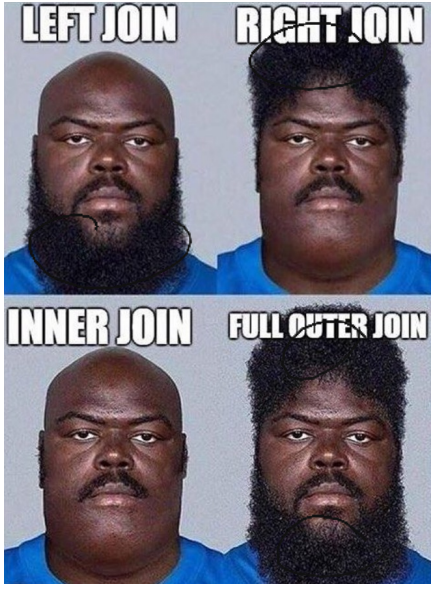
Joins

L	Operator	R	Visualization
			
Keep dangling			
		Keep dangling	
Keep dangling		Keep dangling	

Summary

Algebra
Unary
Set
Join

Joins



```
postgres=# exit
Press any key to continue . . .
```

Solutions

Preliminary

Why Algebra?

- Allows for *query optimization**
- Rewrite a query such that:
  - It is *equivalent* to the original query
  - It can be evaluated faster than the original query

Example

> **Question**
>
> Which of the following operation is "faster"?

A. (a × b) + (a × c)
B. a × (b + c)

> **Quiz #1**
>
> Is it equivalent?

Depends on the type

- ***YES*** if
  - a:`INT`
  - b:`INT`
  - c:`INT`
- ***NO*** if
  - a:`INT`
  - b:`TEXT`
  - c:`TEXT`

*While we do not penalize for inefficient queries, it is useful to know.

Solutions

Selection

Exercise #1

Find all **employees** that **(a) do not work in Sales and are younger than 30** or **(b) work in HR**.

Solution

$\sigma_{\text{(role<>'sales' } \wedge \text{ age<30)} \vee \text{(role='hr')}}(\text{Employees})$

$T1 := \sigma_{[((role \neq 'sales') \wedge (age < 30)) \vee (role = 'hr'))]}(Employees)$

name	age	role
"Sarah"	25	"dev"
"Marie"	36	"hr"
"Emma"	28	"dev"

3 Rows

Solutions

Projection

Exercise #2

Which of the following relational algebra expression resulted in the output below?

name	manager	start_year	end_year
BigAI	Judy	2020	2025
FastCash	Judy	2018	2025
GlobalDB	Jack	2019	2023
CoreOS	Judy	2020	2020
CoolCoin	Jack	2015	2020

manager	name
Judy	FastCash
Jack	GlobalDB
Jack	CoolCoin

	Choice	Comment	
A	$\sigma_{[start\_year \leq 2019]}(\pi_{[manager,name]}(Projects))$	*NO: after π, no start_year*	✗
B	$\pi_{[manager,name]}(\sigma_{[start\_year<2020]}(Projects))$	*YES: correct column order and correct condition*	✓
C	$\pi_{[manager,name]}(\sigma_{[manager='Jack']}(Projects))$	*NO: missing Row 1*	✗
D	$\pi_{[name,manager]}(\sigma_{[start\_year \leq 2019]}(Projects))$	*NO: wrong column order*	✗

Solutions

Cross Product

Exercise #3

For all the project names, find the offices of the managers.

Solution

$\pi_{name,office}(\sigma_{manager=mname}(Projects \times (\rho_{mname \leftarrow name}(Managers))))$

$T4 := $\pi_{[name,office]}$($T3)

name	office
"BigAI"	"#03-20"
"FastCash"	"#03-20"
"GlobalDB"	"#03-10"
"CoreOS"	"#03-20"
"CoolCoin"	"#03-10"

5 Rows

Solutions

Outer Joins

Exercise #4

Recap the relations "Teams" and "Managers" on the right. How many **row** and **columns** are in the result of the relational algebra expression below?

$$\sigma_{ename \equiv \texttt{NULL}}(Managers \bowtie_{name=ename} Teams)$$

Table "Teams"

ename	pname	hours
Sarah	BigAI	10
Sam	BigAI	5
Bill	BigAI	15
Judy	GlobalDB	20
Max	GlobalDB	5
Sarah	GlobalDB	10
Emma	GlobalDB	35
Max	CoreOS	40
Bill	CoreOS	30
Sam	CoolCoin	40
Sarah	CoolCoin	25
Emma	CoolCoin	10

Table "Managers"

name	office
Judy	#03-20
Jack	#03-10

	Choice	Comment	
A	1 row, 5 columns	*YES: Only Jack has no match*	✓
B	3 rows, 5 columns	*NO: Judy has no match*	✗
C	1 row, 3 columns	*NO: The number of columns is the sum of both*	✗
D	3 rows, 3 columns	*NO: The number of columns is the sum of both*	✗

Solutions

Alternatives

Valid but with Redundancy

Cross product + attribute selection ≡ join

$$\sigma_{[\text{manager = mname}]}(\text{Projects} \times \rho_{[\text{mname} \leftarrow \text{name}]}(\text{Managers})) \equiv \text{Projects} \bowtie \rho_{[\text{manager} \leftarrow \text{name}]}(\text{Managers})$$

Unnecessary operators

$$\pi_{[\text{name}]}(\pi_{[\text{name, age}]}(\text{Employees})) \equiv \pi_{[\text{name}]}(\text{Employees})$$

Unoptimized query

$$\sigma_{[\text{start\_year = 2020}]}(\text{Projects} \bowtie_{[\text{manager = mname}]} \rho_{[\text{mname} \leftarrow \text{name}]}(\text{Managers})) \equiv (\sigma_{[\text{start\_year = 2020}]}(\text{Projects})) \bowtie \rho_{[\text{manager} \leftarrow \text{name}]}(\text{Managers})$$

Quiz #2

Can you use *natural join* for the first and last operations?