NUS | Computing
National University of Singapore

CS2102: Database Systems

Lecture 2 — Relational Algebra

Course Policies — Breaking News

- Right Infringements on NUS Course Materials

All course participants (including permitted guest students) who have access to the course materials on LumiNUS or any approved platforms by NUS for delivery of NUS modules are not allowed to re-distribute the contents in any forms to third parties without the explicit consent from the module instructors or authorized NUS officials.

Quick Recap: The Relational Model

- Basic concept: **relations**
  - Unified representation of all data using tables with rows and tables
  - Relation = set of tuples (row of table) filled with atomic values (or *null*)

- Structural integrity constraints
  (condition that restricts what constitutes valid data)
  - Domain constraints
  - Key constraints
  - Foreign key constraints

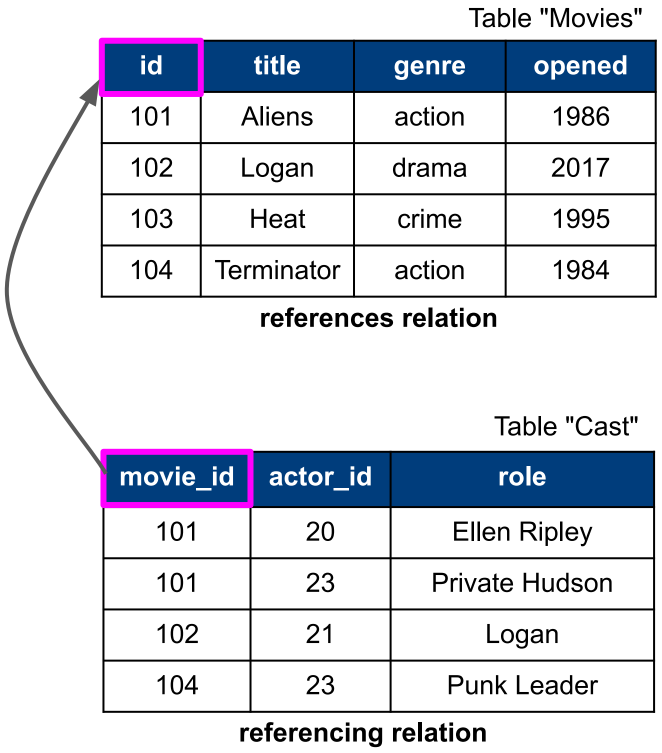


Table "Movies"

id	title	genre	opened
101	Aliens	action	1986
102	Logan	drama	2017
103	Heat	crime	1995
104	Terminator	action	1984

references relation

Table "Cast"

movie_id	actor_id	role
101	20	Ellen Ripley
101	23	Private Hudson
102	21	Logan
104	23	Punk Leader

referencing relation

Missing: formal method to process and query relations ➜ **Relational Algebra**

Quick Recap: (Structural) Integrity Constraints

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

- Example from Lecture 1
  - Without any key constraints, relation on the right is perfectly valid ➜ DBMS does not complain
  - Problematic semantics from an application perspective (e.g., CS2021 gives different credits, with and without an exam???)
  - Goal: avoid different values for "mc" and "exam" for the same course ➜ Pick {course} as primary key

course	mc	exam
cs2102	2	yes
cs2102	2	no
cs2102	4	yes
cs2102	4	no
cs3223	2	yes
...	...	...
null	4	no
null	null	no
null	null	null

Overview

- **Relation Algebra (RA)**
  - Motivation & overview
  - Closure property
- Basic operators
  - Unary operators: selection, projection, renaming
  - Set operators
  - Cross product
- Join operators
  - Inner joins
  - Outer joins
- Complex RA expressions

Relational Algebra

- **Algebra** — mathematical system consisting of
  - Operands: variables or values from which new values can be constructed
  - Operators: symbols denoting procedures that construct new values from given values

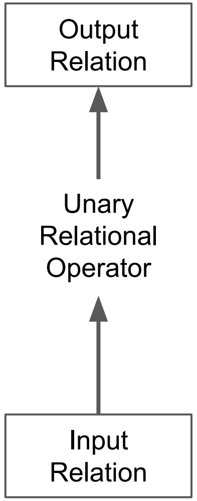
- **Relation Algebra** — procedural query language
  - Operands = relations (or variables representing relations)
  - Operators = transform one or more input relations into <u>one</u> output relation

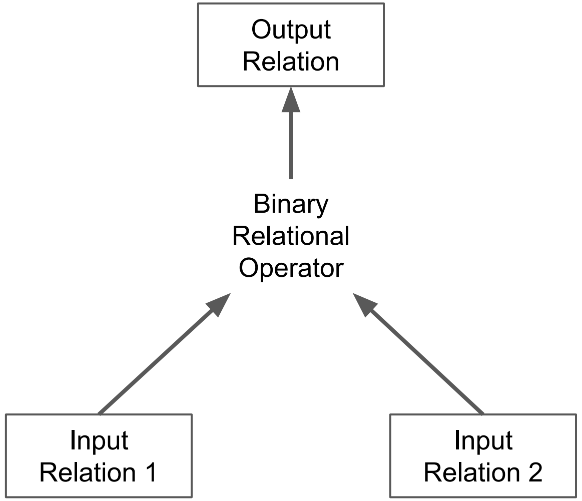
- Basic operators of the Relational Algebra
  - Unary operators: selection σ, projection π, (renaming ρ)
  - Binary operators: cross-product $\times$, union $\cup$, set difference $-$

All other existing operators can be expressed using these basic operators

Closure Property

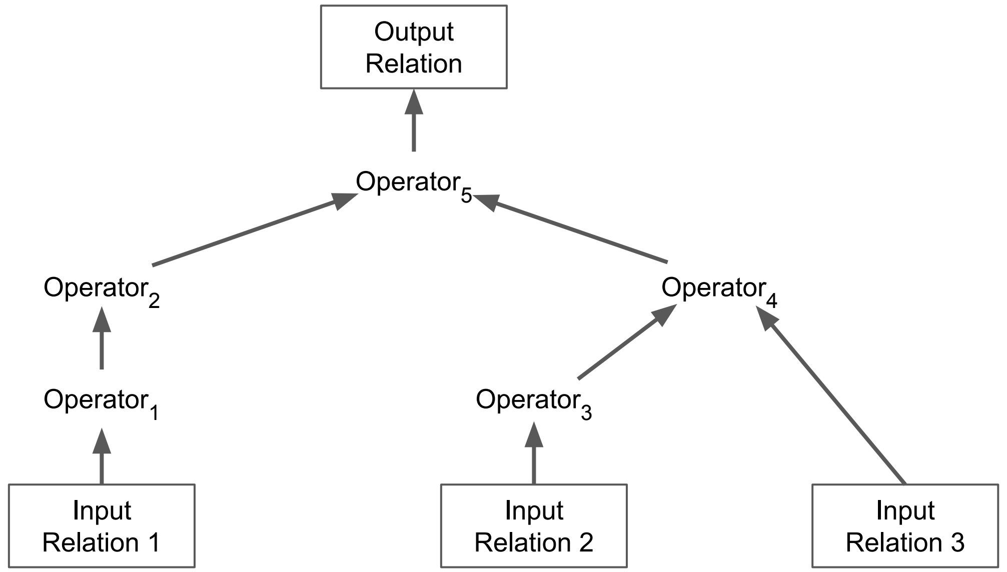
- **Closure:** relations are *closed* under the Relational Algebra
  - All input operands and the outputs of all operators are relations
  - The output of one operator can serve as input for subsequent operators





Closure Property

- Closure property allows for the nesting of relational operators
  ➜ **relational algebra expressions**



Example Database

- Simplified company database schema (primary keys are underlined)

  Employees (<u>name: **text**</u>, age: **integer,** role**: text**)

  Managers (<u>name: **text**</u>, office: **text**)

  Teams (<u>ename: **text**, pname: **text**</u>, hours: **integer**)

  Projects (<u>name: **text**</u>, manager: **text**, start_year: **integer**, end_year: **integer**)

- Foreign key constraints
  - Manager.name → Employees.name
  - Teams.ename → Employees.name
  - Teams.pname → Projects.name
  - Projects.manager → Manager.name

Example Database

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

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

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

Managers

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

Overview

- Relation Algebra (RA)
  - Motivation & Overview
  - Closure Property
- **Basic operators**
  - **Unary operators:** selection, projection, renaming
  - Set operators
  - Cross product
- Join operators
  - Inner joins
  - Outer joins
- Complex RA expressions

Selection σ_c

- $\sigma_c(R)$ selects all tuples from a relation R (i.e., rows from a table) that satisfy the *selection condition c*
  - For each tuple $t \in R$, $t \in \sigma_c(R)$ iff c evaluates to **true** on t
  - Input and output relation have the same schema

Example: Find all projects where Judy is the manager.

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

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

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

Selection Conditions

- A **selection condition** is boolean expression of one of the following forms:

attribute **op** constant	$\sigma_{\text{start\_year}=2020}(\text{Projects})$	***constant selection***
attribute$_1$ **op** attribute$_2$	$\sigma_{\text{start\_year}=\text{end\_year}}(\text{Projects})$	***attribute selection***
expr$_1$ $\wedge$ expr$_2$	$\sigma_{\text{start\_year}=2020\ \wedge\ \text{manager}=\text{'Judy'}}(\text{Projects})$	
expr$_1$ $\vee$ expr$_2$	$\sigma_{\text{start\_year}=2020\ \vee\ \text{manager}=\text{'Judy'}}(\text{Projects})$	
$\neg$ expr	$\sigma_{\neg(\text{start\_year}=2020)}(\text{Projects})$	
(expr)		

- with
  - **op** $\in \{=, <>, <, \leq, \geq, >\}$
  - Operator precedence: (), **op**, $\neg$, $\wedge$, $\vee$

Selection with *null* Values

- Rules of handling *null* values
  - The result of a comparison operation with *null* is ***unknown***
  - The result of an arithmetic operation with *null* is ***null***

- Examples: assume that the value of x is *null*

x < 2020	➜ ***unknown***
x = null	➜ ***unknown***
x <> null	➜ ***unknown***
x + 5	➜ ***null***

Employees

name	age	role
...	...	...
Sam	30	sales
Bernie	19	*null*
Emma	28	dev
...	...	...

Three-Valued Logic: true, false, unknown

c_1	c_2	$c_1 \wedge c_2$	$c_1 \vee c_2$	$\neg c_1$
false	false	false	false	**true**
false	unknown	false	unknown	**true**
false	true	false	**true**	**true**
unknown	false	false	unknown	unknown
unknown	unknown	unknown	unknown	unknown
unknown	true	unknown	**true**	unknown
true	false	false	**true**	false
true	unknown	unknown	**true**	false
true	true	**true**	**true**	false

Recall: For each tuple $t \in R,\ t \in \sigma_c(R)$ iff c evaluates to **true** on t

Selection — Example

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

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

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

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

Projection π_ℓ

- $\pi_\ell(R)$ projects all the attributes of a relation specified in list ℓ
  - i.e., projects all columns of a table specified in list ℓ
  - The order of attributes in ℓ matters

Example: Find all projects and their team members.

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

$\pi_{\text{pname,ename}}(\text{Teams})$

pname	ename
BigAI	Sarah
BigAI	Sam
BigAI	Bill
GlobalDB	Judy
GlobalDB	Max
GlobalDB	Sarah
GlobalDB	Emma
CoreOS	Max
CoreOS	Bill
CoolCoin	Sam
CoolCoin	Sarah
CoolCoin	Emma

Projection π_ℓ

- Relation = set of tuples ➜ duplicate tuples are removed from output relation

Example: Find all projects that have team members.

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

⇨

$$\pi_{\text{pname}}(\text{Teams})$$

pname
BigAI
GlobalDB
CoreOS
CoolCoin

Quick Quiz

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

Which **algebra expression** resulted in the output below?

manager	name
Judy	FastCash
Jack	GlobalDB
Jack	CoolCoin

A $\sigma_{\text{start\_year} \leq 2019}(\pi_{\text{manager,name}}(\text{Projects}))$

B $\pi_{\text{manager,name}}(\sigma_{\text{start\_year} < 2020}(\text{Projects}))$

C $\pi_{\text{manager,name}}(\sigma_{\text{manager} = 'Jack'}(\text{Projects}))$

D $\pi_{\text{name,manager}}(\sigma_{\text{start\_year} \leq 2019}(\text{Projects}))$

Renaming ρ_ℓ

- $\rho_\ell(R)$ renames the attributes of a relation *R* — 2 popular formats for ℓ
  assume that *R* is a relation with schema *R*(*A_1*, *A_2*, …, *A_n*)
  - ℓ is the new schema in terms of the new attribute names
    For example, $\ell = (B_1, B_2, ..., B_n)$
    (note that $B_i = A_i$ if attribute A_i does not get renamed)
  - ℓ is a list of attribute renamings of the form: $B_i \leftarrow A_i, ..., B_k \leftarrow A_k$
    Each renaming $B_j \leftarrow A_j$ renames attribute A_j to B_j
    (note that order of the attribute renamings does not matter)
- Renaming is relevant for set and join operations (discussed later...)

Renaming — Example

$$\rho_{(\text{name},\text{title},\text{hours})}(\text{Teams})$$

or

$$\rho_{\text{name}\leftarrow\text{ename},\text{title}\leftarrow\text{pname}}(\text{Teams})$$

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

$\Rightarrow$

name	title	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

Overview

- Relation Algebra (RA)
  - Motivation & Overview
  - Closure Property
- **Basic operators**
  - Unary operators: selection, projection, renaming
  - **Set operators**
  - Cross product
- Join operators
  - Inner joins
  - Outer joins
- Complex RA expressions

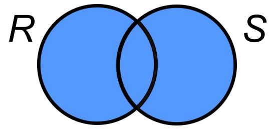
Set Operators

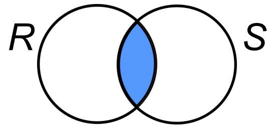
Note: The intersection operator is not fundamentally required as it can be expressed with union & difference

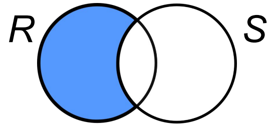
$$R \cap S = (R \cup S) - ((R - S) \cup (S - R))$$

- Recall: a relation is a set of tuples

- Three set operators (given two relation *R* and *S*)
  - **Union** $R \cup S$ returns a relation with all tuples that are in both *R* or *S*
  - **Intersection** $R \cap S$ returns a relation with all tuples that are in both *R* and *S*
  - **Set difference** $R - S$ returns a relation with all tuples that are in *R* but not in *S*







- Requirement for all set operators: *R* and *S* must be **union-compatible**

Union Compatibility (also: type compatibility)

Note: Just because two relations are union-compatible does not mean that a set operation is semantically meaningful.

- Two relations *R* and *S* are **union-compatible** if
  - *R* and *S* have the same number of attributes and
  - The corresponding attributes have the same or compatible domains
  - But: R and S do not have to use the same attribute names

Employees (name: **text**, age: **integer,** role**: text**)

Teams (ename: **text**, pname: **text**, hours: **integer**)

Teams

ename	pname	hours
Sarah	BigAI	10
Sam	BigAI	5
...	...	...

Employees

name	age	role
Sarah	25	dev
Judy	35	sales
...	...	...

not union-compatible

Employees (name: **text**, role**: text,** age: **integer**)

Teams (ename: **text**, pname: **text**, hours: **integer**)

Teams

ename	pname	hours
Sarah	BigAI	10
Sam	BigAI	5
...	...	...

Employees

name	role	age
Sarah	dev	25
Judy	sales	35
...	...	...

union-compatible

Set Operators — Example

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

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

$$\pi_{\text{pname}}(\sigma_{\text{ename='Bill'}}(\text{Teams}))$$

pname
BigAI
CoreOS

$$\pi_{\text{pname}}(\sigma_{\text{ename='Sarah'}}(\text{Teams}))$$

pname
BigAI
GlobalDB
CoolCoin

$$\pi_{\text{pname}}(\sigma_{\text{ename='Bill'}}(\text{Teams})) - \pi_{\text{pname}}(\sigma_{\text{ename='Sarah'}}(\text{Teams}))$$

pname
CoreOS

Overview

- Relation Algebra (RA)
  - Motivation & Overview
  - Closure Property
- Basic operators
  - Unary operators: selection, projection, renaming
  - Set operators
  - **Cross product**
- Join operators
  - Inner joins
  - Outer joins
- Complex RA expressions

Cross Product $\times$ (also: Cartesian Product)

- The **cross product** combines two relations R and S by forming all pairs of tuples from the two relations

- More formally, given two relations R(A, B, C) and S(X, Y)
  - $R \times S$ returns a relation with schema (A, B, C, X ,Y) defined as
  - $R \times S = \{(a, b, c, x, y) \mid (a, b, c) \in R, (x, y) \in S\}$

- Example:

R

A	B	C
1	0.5	m
2	2.3	f

S

X	Y
a	30
b	10
c	20

$R \times S$

A	B	C	X	Y
1	0.5	m	a	30
1	0.5	m	b	10
1	0.5	m	c	20
2	2.3	f	a	30
2	2.3	f	b	20
2	2.3	f	c	10

Cross Product — Example

Example: Find all pairs of senior employees (age ≥ 45) and junior employees (age ≤ 25).

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

$$\mathrm{S} = \pi_{name}(\sigma_{\mathrm{age} \geq 45}(\mathrm{Employees}))$$

name
Max
Bill

$$\mathrm{J} = \pi_{name}(\sigma_{\mathrm{age} \leq 25}(\mathrm{Employees}))$$

name
Sarah
Bernie

$$\mathrm{S} \times \rho_{\mathrm{jname} \leftarrow \mathrm{name}}(\mathrm{J})$$

name	jname
Max	Sarah
Max	Bernie
Bill	Sarah
Bill	Bernie

Cross Product — Example

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

Managers

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

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

$$\mathrm{M} = \mathrm{Projects} \times (\rho_{\mathrm{mname} \leftarrow \mathrm{name}}(\mathrm{Managers}))$$

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

$$\pi_{\mathrm{name,office}}(\sigma_{\mathrm{manager=mname}}(\mathrm{M}))$$

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

Cross Product — Discussion

- Observation:
  - Given two relations *R* and *S*, the size of the cross product is |*R*|*|*S*|
  - In practice, many to most queries requiring a cross product also require a attribute selection that removes formed pairs of tuples

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

- Goal: Make use of this observation to
  - simplify Relational Algebra expressions and
  - avoid generating all |*R*|*|*S*| output tuples
    (when implementing all algebra operators within a DBMS)

$\Big\}$ ➜ **Join operators**

Overview

- Relation Algebra (RA)
  - Motivation & Overview
  - Closure Property
- Basic operators
  - Unary operators: selection, projection, renaming
  - Set operators
  - Cross product
- **Join operators**
  - Inner joins
  - Outer joins
- Complex RA expressions

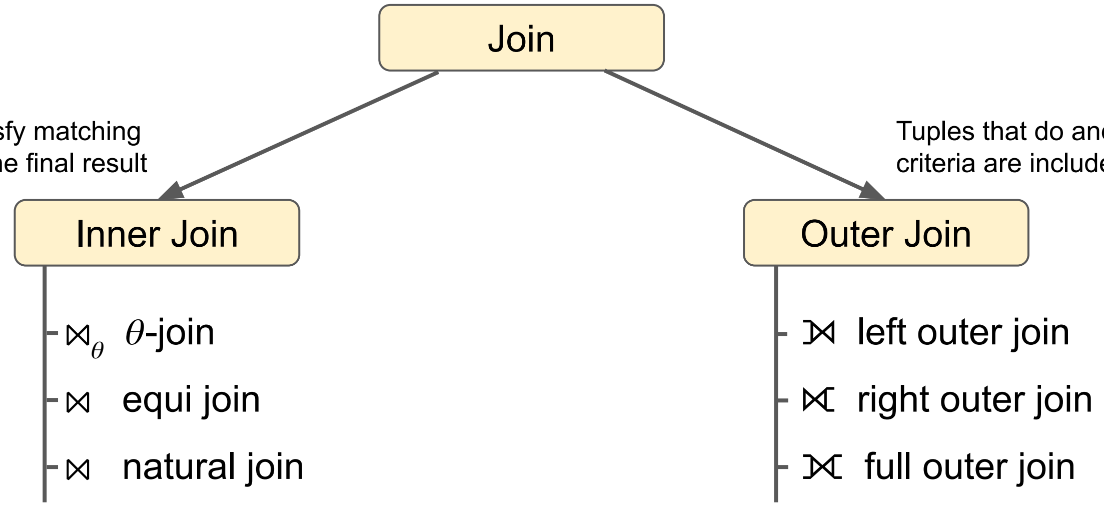
Join Operators

- Join operator — basic idea
  - Combines cross product, attribute selection and possibly projection into a single operator
  - Typically results in simpler relational algebra expressions when formulating queries

- Base types

Only the tuples that satisfy matching criteria are included in the final result

Tuples that do and do not satisfy the matching criteria are included in the final result



Inner Joins — θ-Join

- The θ-join $R \bowtie_{\theta} S$ of two relations R and S is defined as

$$\mathrm{R} \bowtie_{\theta} \mathrm{S} = \sigma_{\theta}(\mathrm{R} \times \mathrm{S})$$

Managers

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

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

Note: final projection omitted here to show result of θ-join

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

$$\mathrm{Projects} \bowtie_{\text{manager=name}} \mathrm{Managers}$$

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

Inner Joins — Equi Join $\bowtie$

- Difference between θ-join and equi join is only w.r.t. matching criteria
  - The θ-join $\bowtie_\theta$ allows arbitrary comparison operators for the attribute selection (e.g., =, <>, <, ≤, ≥, >)
  - The equi join $\bowtie$ is a special case of θ-join by defined over the equality operator (=) only
  - Dedicated term since attribute selections using the equality operator are most common (particularly when joining along foreign key constraints)

- Example
  - see previous slide

Natural Join $\bowtie$

- Same as equi join (i.e., only equality operator) but:
  - The join is performed over all attributes that R and S have in common
    (this means that no explicit matching criteria has to specified)
  - The output relations contains the common attributes of R and S only once
    (compared with the θ-join and equi join that contain all attributes of both relations)

- More formally, the **natural join** of two relation R and S is defined as

$$R \bowtie S = \pi_{\ell}(R \bowtie_c \rho_{b_i \leftarrow a_i, \ldots, b_k \leftarrow a_k}(S))$$

  - $A = \{a_i, \ldots, a_k\}$ is the set of attributes that R and S have in common
  - $c = ((a_i = b_i) \land \ldots \land (a_k = b_k))$
  - $\ell =$ list of all attributes of *R* + list of all attributes in *S* that are not in *A*

Natural Join — Example

Quick Quiz: What would be the result of

$$\text{Projects} \bowtie \text{Managers}$$

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

Note: final projection omitted
to show result of natural join

Managers

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

$$\rho_{\text{manager} \leftarrow \text{name}}(\text{Managers})$$

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

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

$$\text{Projects} \bowtie (\rho_{\text{manager} \leftarrow \text{name}}(\text{Managers}))$$

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

Outer Joins

Note: This simple example can easily be solved using projection and set difference.

- Motivation
  - Inner joins eliminate all tuples that do not satisfy matching criteria (i.e., attribute selection)
  - Sometimes the tuples in *R* or *S* that do <u>not</u> match with tuples in the other relation are of interest

    ➜ ***dangling tuples***

Example: Find all employees that are <u>not</u> assigned to any project.

An inner join can only find all employees that are assigned to at least one project.

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

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

Outer Joins

Quick Quiz: Why is there $\pi_{\text{ename}}(\text{Projects})$ and do we really need it?

- Processing steps of an outer join between *R* and *S* (informal)
  - Perform inner join $M = R \bowtie_\theta S$
  - To *M*, add dangling tuples to result of

R	in case of a **left outer join** $⟕_\theta$
S	in case of a **right outer join** $⟖_\theta$
R and *S*	in case of a **full outer join** $⟗_\theta$

  - "Pad" missing attribute values of dangling tuples with *null*

Example: Find all employees that are <u>not</u> assigned to any project.

$$\text{Employees} ⟕_{\text{name=ename}} (\pi_{\text{ename}}(\text{Projects}))$$

name	age	role	ename
...	...	...	...
Jack	40	dev	Jack
Bill	45	dev	Bill
Marie	36	hr	***null***
Bernie	19	*null*	***null***

inner join result

dangling tuples with *null* padding

Outer Joins — Formal Definitions

name	age	role	ename
...	...	...	...
Jack	40	dev	Jack
Bill	45	dev	Bill
Marie	36	hr	***null***
Bernie	19	*null*	***null***

- Auxiliary definitions
  - $dangle(R \bowtie_\theta S)$ = set of dangling tuples in R w.r.t. $R \bowtie_\theta S$

    ➜ $dangle(R \bowtie_\theta S) \subseteq R$

  - $null(R)$ = n-component tuple of null values where n is the number of attributes of R

    e.g., *null*(Teams) = (*null*, *null*, *null*)

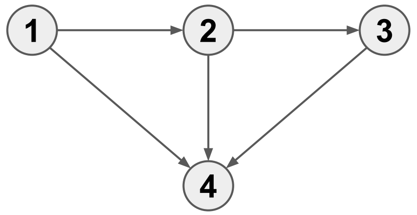
- Definitions (outer joins)

Left outer join $R ⟕_\theta S = R \bowtie_\theta S \cup (dangle(R \bowtie_\theta S) \times \{null(S)\})$

Right outer join $R ⟖_\theta S = R \bowtie_\theta S \cup (\{null(R)\} \times dangle(S \bowtie_\theta R))$

Full outer join $R ⟗_\theta S = \underbrace{R \bowtie_\theta S}_{\text{inner join}} \cup \underbrace{(dangle(R \bowtie_\theta S) \times \{null(S)\}) \cup (\{null(R)\} \times dangle(S \bowtie_\theta R))}_{\text{dangling tuples with } null \text{ padding}}$

Full Outer Join — Example



Example: Find all nodes with not incoming or outgoing edge.

$$\text{Edges} ⟗_{\text{t=in}} \rho_{(\text{in},\text{out})}(\text{Edges})$$

Edges

s	t
1	2
1	4
2	3
2	4
3	4

s	t	in	out
null	*null*	1	2
null	*null*	1	4
1	2	2	3
1	2	2	4
2	3	3	4
1	4	*null*	*null*
2	4	*null*	*null*
3	4	*null*	*null*

Natural Outer Joins

- Analog to natural (inner) join
  - Only the equality operator is used for the join condition
  - The join is performed over all attributes that *R* and *S* have in common
  - The output relations contains the common attributes of *R* and *S* only once

Natural left outer join $R ⟕ S$

Natural right outer join $R ⟖ S$

Natural full outer join $R ⟗ S$

Edges

s	t
1	2
1	4
2	3
2	4
3	4

Quick Quiz: What is the result of the expression:

$\text{Edges} ⟗ \text{Edges}$

Quick Quiz

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

Managers

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

How many **rows & columns** has the result of the algebra expression below?

$$\sigma_{\text{ename}=null}(\text{Managers} \rtimes_{\text{name=ename}} \text{Teams})$$

A — 1 row, 5 cols

B — 2 rows, 5 cols

C — 1 row, 3 cols

D — 2 rows, 3 cols

Overview

- Relation Algebra (RA)
  - Motivation & Overview
  - Closure Property
- Basic operators
  - Unary operators: selection, projection, renaming
  - Set operators
  - Cross product
- Join operators
  - Inner joins
  - Outer joins
- **Complex RA expressions**

Complex Relational Expressions (Queries)

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

$$\mathrm{P} = \sigma_{\mathrm{start\_year} \geq 2020}(\mathrm{Projects})$$

name	manager	start_year	end_year
BigAI	Judy	2020	2025
CoreOS	Judy	2020	2020

$$\mathrm{M} = \pi_{\mathrm{name,manager,office}}(\mathrm{P} \bowtie \rho_{\mathrm{manager} \leftarrow \mathrm{name}}(\mathrm{Managers}))$$

name	manager	office
BigAI	Judy	#03-20
CoreOS	Judy	#03-20

$$\mathrm{W} = \sigma_{\mathrm{hours} \geq 30}(\mathrm{Teams})$$

ename	pname	hours
Emma	GlobalDB	35
Max	CoreOS	40
Bill	CoreOS	30
Sam	CoolCoin	40

$$\mathrm{T} = \mathrm{W} \bowtie_{\text{pname=name}} \mathrm{M}$$

ename	pname	hours	name	manager	office
Max	CoreOS	40	CoreOS	Judy	#03-20
Bill	CoreOS	30	CoreOS	Judy	#03-20

Managers

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

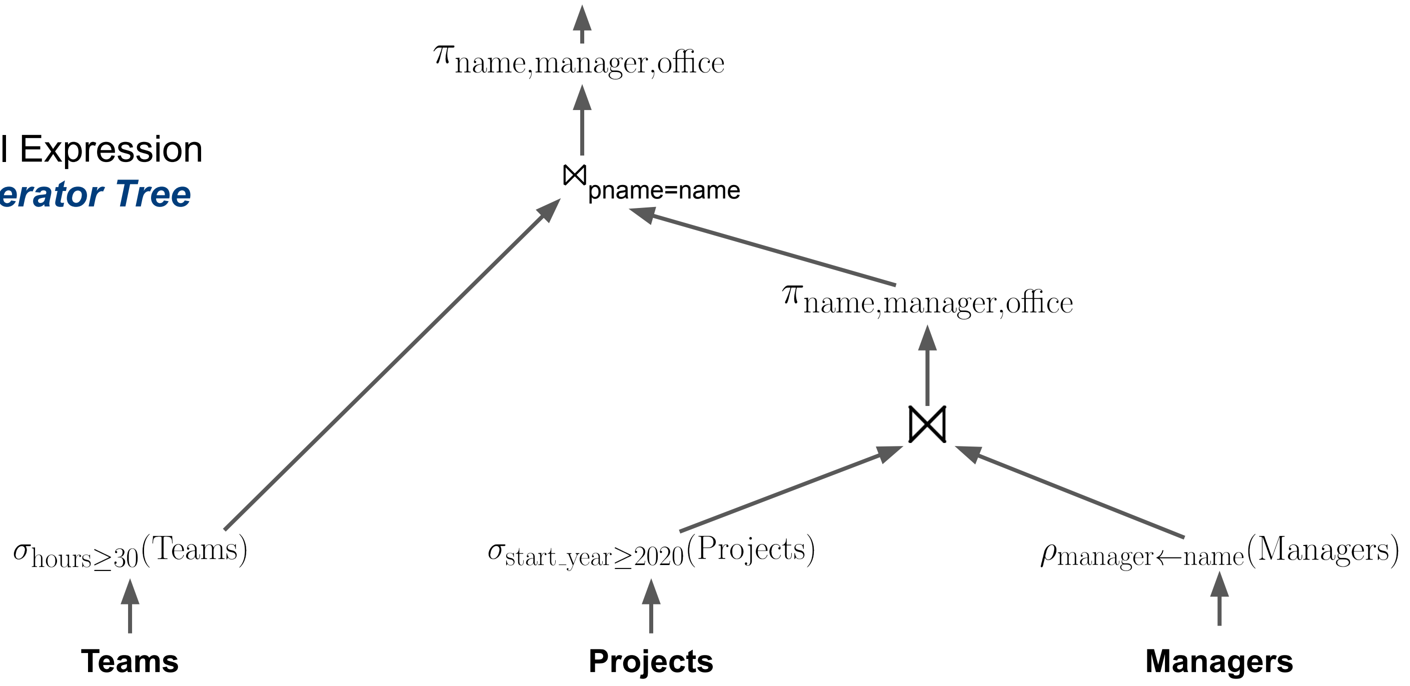
$$\pi_{\mathrm{name,manager,office}}(\mathrm{T})$$

name	manager	office
CoreOS	Judy	#03-20

Complex Relational Expressions (Queries)

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

Relational Expression as an ***Operator Tree***

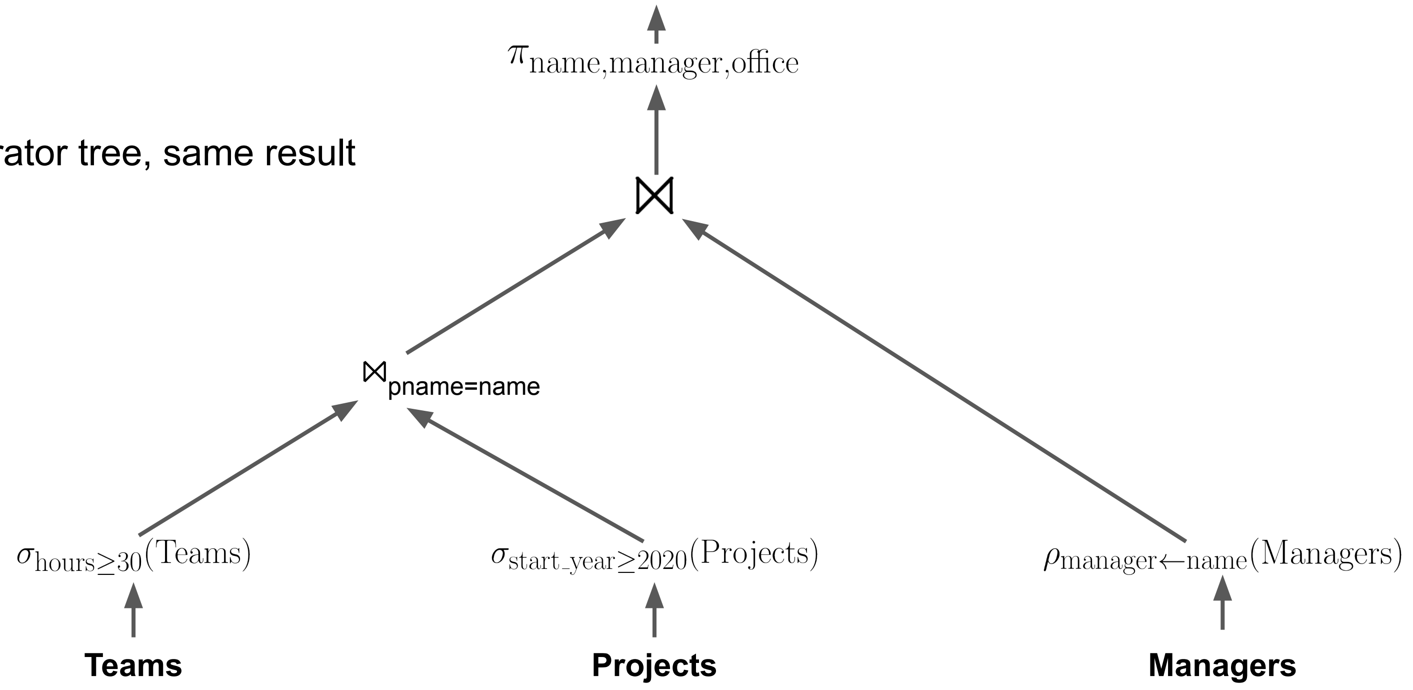


$\pi_{\text{name,manager,office}}$
$\bowtie_{\text{pname=name}}$
$\pi_{\text{name,manager,office}}$
$\bowtie$
$\sigma_{\text{hours}\geq 30}(\text{Teams})$
$\sigma_{\text{start\_year}\geq 2020}(\text{Projects})$
$\rho_{\text{manager}\leftarrow\text{name}}(\text{Managers})$
Teams
Projects
Managers

Complex Relational Expressions (Queries)

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

Different operator tree, same result



$\pi_{\text{name,manager,office}}$
$\bowtie$
$\bowtie_{\text{pname=name}}$
$\sigma_{\text{hours}\geq 30}(\text{Teams})$
$\sigma_{\text{start\_year}\geq 2020}(\text{Projects})$
$\rho_{\text{manager}\leftarrow\text{name}}(\text{Managers})$
Teams
Projects
Managers

Complex Relational Expressions — Observation

- In general, multiple ways to formulate a query to get the same result, e.g.,
  - 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
  - ...and many more

- Finding the "best" operator tree ➜ **query optimization**
  - Handled by the DBMS transparent to the user
  - Covered in, e.g., CS3223

Invalid Relational Expressions — Examples

- Attribute no longer available after projection

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

- Attribute no longer available after renaming

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

- Incompatible attribute types

$$\sigma_{\text{age}=\text{role}}(\text{Employees})$$

Valid but not "Smart" Expressions — Examples

- Cross product + attribute selection instead join

$$\sigma_{\text{manager}=\text{mname}}(\text{Projects} \times (\rho_{\text{mname} \leftarrow \text{name}}(\text{Managers}))) \rightarrow \text{Projects} \bowtie_{\text{manager=name}} \text{Managers}$$

- Unnecessary operators

$$\pi_{\text{name}}(\pi_{\text{name,age}}(\text{Employees})) \rightarrow \pi_{\text{name}}(\text{Employees})$$

- Query optimization (performance)

$$\sigma_{\text{start\_year}=2020}(\text{Projects} \bowtie_{\text{manager=name}} \text{Managers}) \rightarrow (\sigma_{\text{start\_year}=2020}(\text{Projects})) \bowtie_{\text{manager=name}} \text{Managers}$$

Note: query optimization is beyond the scope of CS2102 and covered in other modules (e.g., CS3223). A solid grasp of the Relational Algebra is very important for this topic.

Summary

- Relational Algebra
  - Formal method to query relational data
  - Closure property for arbitrarily complex relational expressions
  - Basis for DB query languages such as SQL

- Most common operators
  - Unary operators: selection, projection, renaming
  - Binary operators: set operators, (cross product), joins