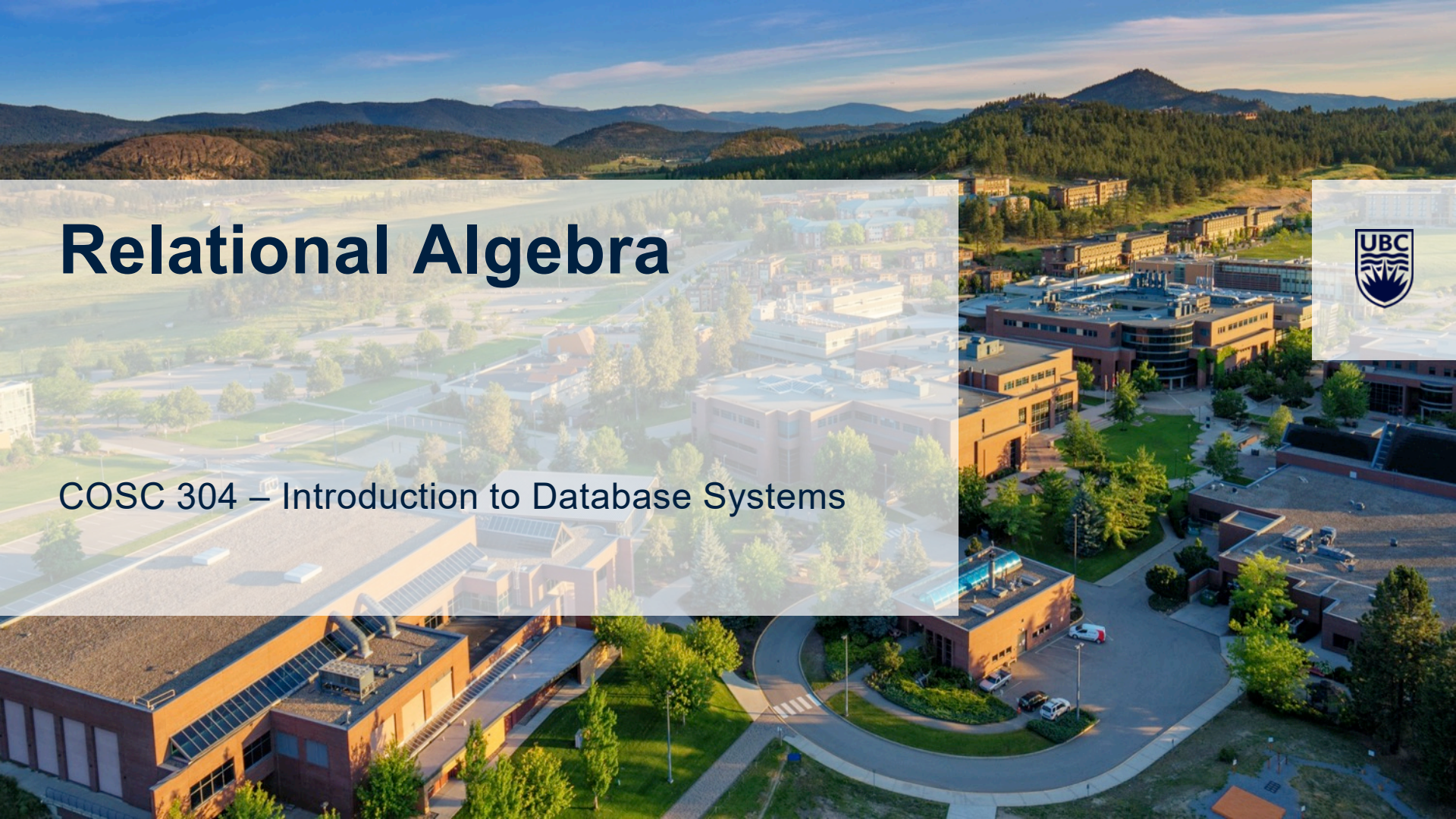


Relational Algebra

COSC 304 – Introduction to Database Systems

Relational Algebra Query Language

A ***query language*** is used to update and retrieve data that is stored in a data model.

Relational algebra is a set of relational operations for retrieving data.

- Just like algebra with numbers, relational algebra consists of operands (which are relations) and a set of operators.

Every relational operator takes as input one or more relations and produces a relation as output.

A sequence of relational algebra operators is called a ***relational algebra expression***.

Relational algebra is the foundation of all relational database systems. SQL gets translated into relational algebra.

Relational Algebra Operators

Operator	Symbol	Purpose
Selection	σ	Filter rows
Projection	Π	Keep only certain columns
Cartesian Product	$\times$	Combine two tables in all possible ways
Join	$\bowtie$	Combine two tables based on a condition
Union	$\cup$	Keep rows in either of two tables
Difference	-	Keep rows in first table that are not in second
Intersection	$\cap$	Keep rows that are in both tables

Relational algebra operators are fundamental to data processing and occur in other data systems (even non-relational).

Selection Operation

The ***selection operation*** takes a relation as input and returns a new relation as output that contains tuples that satisfy a condition, called a ***predicate***.

- The predicate is similar to a condition in an `if` statement.
- The output relation has the same number of columns as the input relation, but may have less rows.

Selection operation on relation *R* with predicate *F* is denoted by $\sigma_F(R)$.

- The predicate uses operands that are attributes or constants/expressions, comparison operators (<, >, =, ≠, ≤, ≥), and logical operators (AND, OR, NOT).

Selection Example

Emp Relation

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

$\sigma_{title = 'EE'}(Emp)$

eno	ename	title	salary
E1	J. Doe	EE	30000
E6	L. Chu	EE	30000

$\sigma_{salary > 35000\ OR\ title = 'PR'}(Emp)$

eno	ename	title	salary
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

Selection Question

Question: Given this table and the query:

$$\sigma_{salary > 50000 \text{ or } title='PR'}(Emp)$$

How many rows are returned?

A) 0

B) 1

C) 2

D) 3

Emp Relation

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

Selection Question (2)

Question: Given this table and the query:

$$\sigma_{salary > 50000 \text{ or } title='PR'}(Emp)$$

How many columns are returned?

A) 0

B) 2

C) 3

D) 4

Emp Relation

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

Projection Operation

The ***projection operation*** takes a relation as input and returns a new relation as output that contains a subset of the attributes of the input relation and all non-duplicate tuples.

- The output relation has the same number of tuples as the input relation unless removing the attributes caused duplicates to be present.
- Question: When are we guaranteed to never have duplicates when performing a projection operation?

Projection on relation R with output attributes $A_1,\ldots,A_m$ is denoted by $\boldsymbol{\Pi_{A_1,\ldots,A_m}(R)}$.

- Order of $A_1,\ldots, A_m$ is significant in the result.
- Cardinality of $\Pi_{A_1,\ldots,A_m}(R)$ may not be the same as R due to duplicate removal.

Projection Example

Emp Relation

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

$\Pi_{eno,ename}(Emp)$

eno	ename
E1	J. Doe
E2	M. Smith
E3	A. Lee
E4	J. Miller
E5	B. Casey
E6	L. Chu
E7	R. Davis
E8	J. Jones

$\Pi_{title}(Emp)$

title
EE
SA
ME
PR

Projection Question

Question: Given this table and the query:

$$\Pi_{title}(\text{Emp})$$

How many rows are returned?

A) 0

B) 2

C) 4

D) 8

Emp Relation

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

Selection and Projection Questions

WorksOn Relation

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P5	Engineer	23
E8	P3	Manager	40

Write the relational algebra expression that:

1) Returns all rows with an employee working on project P2.
2) Returns all rows with an employee who is working as a manager on a project.
3) Returns all rows with an employee working as a manager for more than 40 hours.
4) Returns only attributes *resp* and *hours*.
5) Returns only *eno*.
6) Returns only *pno*.

How many tuples in each output relation?

Union

Union takes two relations *R* and *S* as input and produces an output relation that includes all tuples that are either in *R*, or in *S*, or in both *R* and *S*. Duplicate tuples are eliminated. Syntax: $R \cup S$

R and *S* must be ***union-compatible***:

- 1) Both relations have same number of attributes.
- 2) Each attribute pair, R_i and S_i, have compatible data types for all attribute indexes *i*.
- Note that attributes do not need to have the same name.
- Result has attribute names of first relation.

Union Example

Emp

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

WorksOn

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P5	Engineer	23

$\Pi_{eno}(\text{Emp}) \cup \Pi_{eno}(\text{WorksOn})$

eno
E1
E2
E3
E4
E5
E6
E7
E8

Set Difference

Set difference takes two relations *R* and *S* as input and produces an output relation that contains all the tuples of *R* that are not in *S*.

Syntax: ***R – S***

Note:

- R – S ≠ S – R
- R and S must be union compatible.

Set Difference Example

Emp Relation

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

WorksOn Relation

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P5	Engineer	23

$\Pi_{eno}(\text{Emp}) - \Pi_{eno}(\text{WorksOn})$

eno
E4
E8

Question: What is the meaning of this query?

Question: What is $\Pi_{eno}(\text{WorksOn}) - \Pi_{eno}(\text{Emp})$?

Intersection

Intersection takes two relations R and S as input and produces an output relation which contains all tuples that are in both R and S.

- R and S must be union-compatible.

Syntax: $R \cap S$

Note that $R \cap S = R - (R - S) = S - (S - R)$.

Intersection Example

Emp

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

WorksOn

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P5	Engineer	23

$\Pi_{eno}(\text{Emp}) \cap \Pi_{eno}(\text{WorksOn})$

eno
E1
E2
E3
E5
E6
E7

Set Operations: Union-compatible Question

Question: Two tables have the same number of fields in the same order with the same types, but the names of some fields are different.
True or false: The two tables are union-compatible.

A) true

B) false

Cartesian Product

Cartesian product of relations R (of degree k_1) and S (of degree k_2) is a relation of degree $(k_1 + k_2)$ that consists of all $(k_1 + k_2)$-tuples where each tuple is a concatenation of one tuple of R with one tuple of S.
Syntax: $R \times S$

The cardinality of $R \times S$ is $|R| * |S|$.

The Cartesian product is also known as ***cross product***.

Cartesian Product Example

Emp Relation

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000

Proj Relation

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000

Emp × *Proj*

eno	ename	title	salary	pno	pname	budget
E1	J. Doe	EE	30000	P1	Instruments	150000
E2	M. Smith	SA	50000	P1	Instruments	150000
E3	A. Lee	ME	40000	P1	Instruments	150000
E4	J. Miller	PR	20000	P1	Instruments	150000
E1	J. Doe	EE	30000	P2	DB Develop	135000
E2	M. Smith	SA	50000	P2	DB Develop	135000
E3	A. Lee	ME	40000	P2	DB Develop	135000
E4	J. Miller	PR	20000	P2	DB Develop	135000
E1	J. Doe	EE	30000	P3	CAD/CAM	250000
E2	M. Smith	SA	50000	P3	CAD/CAM	250000
E3	A. Lee	ME	40000	P3	CAD/CAM	250000
E4	J. Miller	PR	20000	P3	CAD/CAM	250000

Cartesian Product Question

Question: *R* is a relation with 10 rows and 5 columns. *S* is a relation with 8 rows and 3 columns.

What is the degree and cardinality of the Cartesian product?

A) degree = 8, cardinality = 80

B) degree = 80, cardinality = 8

C) degree = 15, cardinality = 80

D) degree = 8, cardinality = 18

θ-Join

Theta (θ) join is a derivative of the Cartesian product. Instead of taking all combinations of tuples from *R* and *S*, we only take a subset of those tuples that match a given condition *F*. Syntax: $\boldsymbol{R \bowtie_F S}$

Note that $R \bowtie_F S = \sigma_F(R \times S)$.

θ-Join Example

WorksOn Relation

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P4	Engineer	23

Proj Relation

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

WorksOn $\bowtie_{hours*10000 > budget}$ Proj

eno	pno	resp	hours	P.pno	pname	budget
E2	P1	Analyst	24	P1	Instruments	150000
E2	P1	Analyst	24	P2	DB Develop	135000
E3	P4	Engineer	48	P1	Instruments	150000
E3	P4	Engineer	48	P2	DB Develop	135000
E3	P4	Engineer	48	P3	CAD/CAM	250000
E3	P4	Engineer	48	P4	Maintenance	310000
E5	P2	Manager	24	P1	Instruments	150000
E5	P2	Manager	24	P2	DB Develop	135000
E6	P4	Manager	48	P1	Instruments	150000
E6	P4	Manager	48	P2	DB Develop	135000
E6	P4	Manager	48	P3	CAD/CAM	250000
E6	P4	Manager	48	P4	Maintenance	310000
E7	P3	Engineer	36	P1	Instruments	150000
E7	P3	Engineer	36	P2	DB Develop	135000
E7	P3	Engineer	36	P3	CAD/CAM	250000
E7	P4	Engineer	23	P1	Instruments	150000
E7	P4	Engineer	23	P2	DB Develop	135000

Types of Joins

The θ-Join is a general join that allows any expression for condition *F*. However, there are more specialized joins that are frequently used.

A ***equijoin*** only contains the equality operator (=) in formula *F*.

- e.g. WorksOn $\bowtie_{WorksOn.pno = Proj.pno}$ Proj

A ***natural join*** over two relations *R* and *S* denoted by $R \bowtie S$ is the equijoin of *R* and *S* over a set of attributes common to both *R* and *S*.

- It removes the "extra copies" of the join attributes.
- The attributes must have the same name in both relations.

Equijoin Example

WorksOn Relation

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P4	Engineer	23

Proj Relation

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

WorksOn $\bowtie_{WorksOn.pno = Proj.pno}$ Proj

eno	pno	resp	hours	P.pno	pname	budget
E1	P1	Manager	12	P1	Instruments	150000
E2	P1	Analyst	24	P1	Instruments	150000
E2	P2	Analyst	6	P2	DB Develop	135000
E3	P4	Engineer	48	P4	Maintenance	310000
E5	P2	Manager	24	P2	DB Develop	135000
E6	P4	Manager	48	P4	Maintenance	310000
E7	P3	Engineer	36	P3	CAD/CAM	250000
E7	P4	Engineer	23	P4	Maintenance	310000

What is the meaning of this join?

Natural join Example

WorksOn Relation

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P4	Engineer	23

Proj Relation

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

WorksOn ⋈ Proj

eno	pno	resp	hours	pname	budget
E1	P1	Manager	12	Instruments	150000
E2	P1	Analyst	24	Instruments	150000
E2	P2	Analyst	6	DB Develop	135000
E3	P4	Engineer	48	Maintenance	310000
E5	P2	Manager	24	DB Develop	135000
E6	P4	Manager	48	Maintenance	310000
E7	P3	Engineer	36	CAD/CAM	250000
E7	P4	Engineer	23	Maintenance	310000

Natural join is performed by comparing *pno* in both relations.

Join Practice Questions

Emp Relation

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

Proj Relation

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

WorksOn Relation

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P5	Engineer	23
E8	P3	Manager	40

Compute the following joins (counts only):

1) Emp $\bowtie_{\text{title='EE' and budget > 400000}}$ Proj
2) Emp $\bowtie$ WorksOn
3) Emp $\bowtie$ WorksOn $\bowtie$ Proj
4) $\text{Proj}_1 \bowtie_{\text{Proj1.budget > Proj2.budget}} \text{Proj}_2$

Outer Joins

Outer joins are used in cases where performing a join "loses" some tuples of the relations. These are called *dangling tuples*.

There are three types of outer joins:

- 1) ***Left outer join*** - $R \mathbin{⟕} S$ - The output contains all tuples of *R* that match with tuples of *S*. If there is a tuple in *R* that matches with no tuple in *S*, the tuple is included in the final result and is padded with nulls for the attributes of *S*.
- 2) ***Right outer join*** - $R \mathbin{⟖} S$ - The output contains all tuples of *S* that match with tuples of *R*. If there is a tuple in *S* that matches with no tuple in *R*, the tuple is included in the final result and is padded with nulls for the attributes of *R*.
- 3) ***Full outer join*** - $R \mathbin{⟗} S$ - All tuples of *R* and *S* are included in the result whether or not they have a matching tuple in the other relation.

Right Outer Join Example

WorksOn Relation

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P4	Engineer	23

Proj Relation

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

WorksOn $\overset{\subset}{\bowtie}_{WorksOn.pno = Proj.pno}$ Proj

eno	pno	resp	hours	P.pno	pname	budget
E1	P1	Manager	12	P1	Instruments	150000
E2	P1	Analyst	24	P1	Instruments	150000
E2	P2	Analyst	6	P2	DB Develop	135000
E3	P4	Engineer	48	P4	Maintenance	310000
E5	P2	Manager	24	P2	DB Develop	135000
E6	P4	Manager	48	P4	Maintenance	310000
E7	P3	Engineer	36	P3	CAD/CAM	250000
E7	P4	Engineer	23	P4	Maintenance	310000
null	null	null	null	P5	CAD/CAM	500000

Outer Join Question

Question: Given this table and the query:

$$\text{WorksOn} \overset{\text{WorksOn.pno = Proj.pno}}{\supset\!\bowtie} \text{Proj}$$

How many rows are returned?

A) 10

B) 9

C) 8

D) 7

WorksOn Relation

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P4	Engineer	36
E7	P4	Engineer	23

Proj Relation

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

Semi-Join and Anti-Join

A ***semi-join*** between tables returns rows from the first table where one or more matches are found in the second table.

- Semi-joins are used in `EXISTS` and `IN` constructs in SQL.

An ***anti-join*** between two tables returns rows from the first table where ***no*** matches are found in the second table.

- Anti-joins are used with `NOT EXISTS`, `NOT IN`, and `FOR ALL`.
- Anti-join is the complement of semi-join: $R \triangleright S = R - R \ltimes S$

Semi-Join Example

WorksOn Relation

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P4	Engineer	23

Proj Relation

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

Proj $\ltimes_{Proj.pno = WorksOn.pno}$ WorksOn

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

Anti-Join Example

WorksOn Relation

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P4	Engineer	23

Proj Relation

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

Proj $\triangleright_{Proj.pno = WorksOn.pno}$ WorksOn

pno	pname	budget
P5	CAD/CAM	500000

Combining Operations

Relational algebra operations can be combined in one expression by nesting them:

$$\Pi_{eno,pno,hours}(\sigma_{ename='J.\ Doe'}(Emp) \bowtie \sigma_{hours>16}(WorksOn))$$

- Return the eno, pno, and hours for employee 'J. Doe' when he has worked on a project for more than 16 months.

Can also use a temporary relation variables for intermediate results.

- Use the assignment operator $\leftarrow$ for indicating that the result of an operation is assigned to a temporary relation.

$$\begin{aligned}
\text{empdoe} &\leftarrow \sigma_{ename='J.\ Doe'}(\text{Emp}) \\
\text{wohours} &\leftarrow \sigma_{hours>16}(\text{WorksOn}) \\
\text{empwo} &\leftarrow \text{empdoe} \bowtie \text{wohours} \\
\text{result} &\leftarrow \Pi_{eno,pno,hours}(\text{empwo})
\end{aligned}$$

Rename Operation

Renaming can be applied when assigning a result:

$$\text{result(EmployeeNum, ProjectNum, Hours)} \leftarrow \Pi_{eno,pno,hours}\,(empwo)$$

Or by using the rename operator ρ (rho):

$$\rho_{\text{result(EmployeeName, ProjectNum, Hours)}}(empwo)$$

Operator Precedence

Just like mathematical operators, the relational operators have precedence.

The precedence of operators from highest to lowest is:

- unary operators - σ, Π, ρ
- Cartesian product and joins - X, $\bowtie$, division
- intersection
- union and set difference

Parentheses can be used to changed the order of operations.

It is a good idea to ***always*** use parentheses around the argument for both unary and binary operators.

Complete Set of Relational Algebra Operators

It has been shown that the relational operators $\{\sigma, \Pi, \times, \cup, -\}$ form a complete set of operators.

- That is, any of the other operators can be derived from a combination of these 5 basic operators.

Examples:

- Intersection - $R \cap S \equiv R \cup S - ((R - S) \cup (S - R))$
- We have also seen how a join is a combination of a Cartesian product followed by a selection.

Relational Algebra Query Examples

Consider the database schema

Emp (eno, ename, title, salary)

Proj (pno, pname, budget)

WorksOn (eno, pno, resp, hours)

Queries:

- List the names of all employees.
  - $\Pi_{ename}(Emp)$
- Find the names of projects with budgets over $100,000.
  - $\Pi_{pname}(\sigma_{budget>100000}(Proj))$

Practice Questions

Relational database schema:

branch (bname, address, city, assets)
customer (cname, street, city)
deposit (accnum, cname, bname, balance)
borrow (accnum, cname, bname, amount)

1) List the names of all branches of the bank.
2) List the names of all deposit customers together with their account numbers.
3) Find all cities where at least one customer lives.
4) Find all cities with at least one branch.
5) Find all cities with at least one branch or customer.
6) Find all cities that have a branch but no customers who live in that city.

Practice Questions (2)

branch (bname, address, city, assets)
customer (cname, street, city)
deposit (accnum, cname, bname, balance)
borrow (accnum, cname, bname, amount)

1) Find the names of all branches with assets greater than $2,500,000.

2) List the name and cities of all customers who have an account with balance greater than $2,000.

3) List all the cities with at least one customer but without any bank branches.

4) Find the name of all the customers who live in a city with no bank branches.

Practice Questions (3)

branch (bname, address, city, assets)
customer (cname, street, city)
deposit (accnum, cname, bname, balance)
borrow (accnum, cname, bname, amount)

1) Find all the cities that have both customers and bank branches.

2) List the customer name and loan and deposit amounts, who have a loan larger than a deposit account at the same branch.

3) Find the name and assets of all branches which have deposit customers living in Vancouver.

4) Find all the customers who have both a deposit account and a loan at the branch with name CalgaryCentral.

5) Your own?

Other Relational Algebra Operators

There are other relational algebra operators that we will not discuss. Most notably, we often need ***aggregate operations*** that compute functions on the data.

For example, given the current operators, we cannot answer the query:

- What is the total amount of deposits at the Kelowna branch?

We will see how to answer these queries when we study SQL.

Conclusion

Relational algebra is a set of operations for answering queries on data stored in the relational model.

Operator	Symbol	Purpose
Selection	σ	Filter rows
Projection	Π	Keep only certain columns
Cartesian Product	$\times$	Combine two tables in all possible ways
Join		Combine two tables based on a condition
Union	$\cup$	Keep rows in either of two tables
Difference	-	Keep rows in first table that are not in second
Intersection	$\cap$	Keep rows that are in both tables

By combining relational operators, queries can be answered over the base relations.

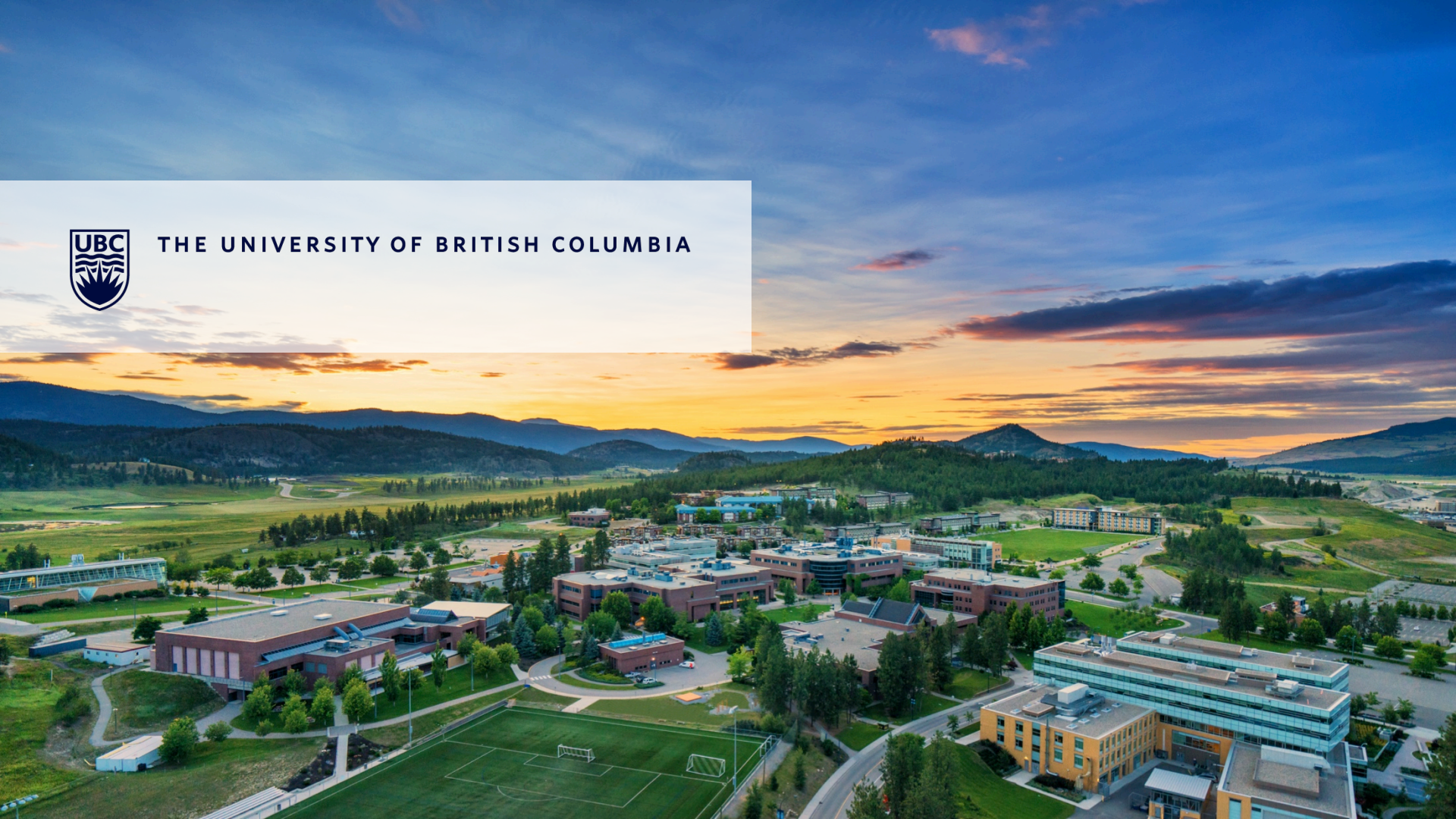
Objectives

Define: relational algebra, query language

Define and perform all relational algebra operators.

List the operators which form the complete set of operators.

Given a relational schema and instance be able to translate English queries into relational algebra and show the resulting relation.


UBC
THE UNIVERSITY OF BRITISH COLUMBIA