Relational Algebra

Selection σ (row-wise operation)

STUDENTS TABLE

ID	Name	Age	School
1234	Alice	20	SCSE
5678	Bob	20	EEE
3742	Cathy	22	SCSE
9413	David	21	CEE

→ Query: "Find me the student named Alice"

→ $\sigma_{Name = \text{'Alice'}}$ Students

Results

ID	Name	Age	School
1234	Alice	20	SCSE

→ Query: "Find SCSE students under 21"

→ $\sigma_{School = \text{'SCSE'} \text{ AND } Age < 21}$ Students

Results

ID	Name	Age	School
1234	Alice	20	SCSE

→ Query: "Find students who are either in SCSE or under 21"

→ $\sigma_{School = \text{'SCSE'} \text{ OR } Age < 21}$ Students

Results

ID	Name	Age	School
1234	Alice	20	SCSE
5678	Bob	20	EEE
3742	Cathy	22	SCSE

Projection π (column-wise)

STUDENTS TABLE

ID	Name	Age	School
1234	Alice	20	SCSE
5678	Bob	20	EEE
3742	Cathy	22	SCSE
9413	David	21	CEE

Results

ID	Name
1234	Alice
5678	Bob
3742	Cathy
9413	David

→ Query: "Find the IDs & Names of all students"

→ $\pi_{ID, Name}$ Students

Combining Operators

STUDENTS TABLE

ID	Name	Age	School
1234	Alice	20	SCSE
5678	Bob	20	EEE
3742	Cathy	22	SCSE
9413	David	21	CEE

Results

ID	Name
1234	Alice
3742	Cathy

→ Query: "Find the IDs & Names of all student in SCSE"

→ $\pi_{ID, Name}(\sigma_{School = \text{'SCSE'}} \text{ Students})$

$\sigma_{School = \text{'SCSE'}}(\pi_{ID, Name} \text{ Students})$ is WRONG.

Projection to come before selection as projection eliminates "School" column

→ selection cannot be performed anymore

Union ∪

Students

Name	Age
Alice	20
Bob	21
Cathy	22
David	21

Volunteers

Name	Age
Cathy	22
David	21
Eddie	43
Fred	35

Results

Name	Age
Alice	20
Bob	21
Cathy	22
David	21
Eddie	43
Fred	35

→ Query: "Find people who are either students or volunteers"

→ Students ∪ Volunteers

Duplicate tupels automatically removed.
e.g. Cathy appears once on results table

→ Query: "Find the NAMES of people who are either students or volunteers"

→ Π_{Name} (Students ∪ Volunteer)

→ (Π_{Name} Students) ∪ (Π_{Name} Volunteer)

same results

Students

Name	Age
Alice	20
Bob	21
Cathy	22
David	21

Volunteers

Name
Cathy
David
Eddie
Fred

Results

Name	Age
Alice	20
Bob	21
Cathy	22
David	21
Eddie	43
Fred	35

→ Query: "Find people who are either students or volunteers"

→ Student ∪ Volunteers is WRONG

→ Correct Solution: (Π_{Name} Students) ∪ Volunteers

The two sides of a union must have the same schema (same set of attributes)

Intersection ∩

Students

Name	Age
Alice	20
Bob	21
Cathy	22
David	21

Volunteers

Name	Age
Cathy	22
David	21
Eddie	43
Fred	35

Results

Name	Age
Cathy	22
David	21

→ Query: "Find people who are both students & volunteers"

→ Students ∩ Volunteers

Duplicate tupels automatically removed.
e.g. Cathy appears once on results table

Students

Name	Age
Alice	20
Bob	21
Cathy	22
David	21

Volunteers

Name
Cathy
David
Eddie
Fred

Results

Name
Cathy
David

→ Query: "Find people who are both students & volunteers"

→ (Π_{Name} Students) ∩ Volunteer

The two sides of an intersection must have the same schema (same set of attributes)

Difference −

Students

Name	Age
Alice	20
Bob	21
Cathy	22
David	21

Volunteers

Name	Age
Cathy	22
David	21
Eddie	43
Fred	35

results

Name	Age
Alice	20
Bob	21

→ Query: "Find people who are students but not volunteers"

→ Students − Volunteers

Duplicate tuples are automatically removed

results

Name	Age
Eddie	43
Fred	35

→ Query: "Find people who are volunteers but not students"

→ Volunteers − Students

Students

Name	Age
Alice	20
Bob	21
Cathy	22
David	21

Volunteers

Name
Cathy
David
Eddie
Fred

Results

Name
Alice
Bob

→ Query: "Find people who are students but not volunteers"

→ $(\Pi_{Name}$ Students$)$ − Volunteers

The two sides of a difference must have the same schema (same set of attributes)

Natural Join ⋈

Students

Name	School
Alice	SCSE
Bob	EEE
Cathy	CEE
David	SCSE

Donations

Name	Amount
Cathy	100
David	200
Eddie	300
Fred	400

Results

Name	School	Amount
David	SCSE	200

→ Query: "For those SCSE students who have made a donation, find their names, schools, & amounts of their donations"

→ $(\sigma_{School = 'SCSE'}$ Students$) \bowtie$ Donations

1. The join is performed based on the common attributes of the 2 relations
2. Each common attribute appear only once in the result

Theta Join $\bowtie_{condition}$

Quiz 1

Name	Score
Alice	70
Bob	90
Cathy	80
David	100

Quiz 2

Name	Score
Alice	80
Bob	90
Cathy	90
David	70

Results

Name	Score	Name	Score
Alice	70	Alice	80
Cathy	80	Cathy	90

→ Query: "Find students who scored higher in Quiz 2 than Quiz 1"

→ Quiz 1 $\bowtie_{Quiz1.Name = Quiz2.Name \text{ AND } Quiz1.Score < Quiz2.Score}$ Quiz 2

→ Note: In the join condition, whenever there are ambiguous attribute names (e.g. Score), need to prefix table name to eliminate ambiguity (e.g. use Quiz1.Score instead of Score)

1. Difference from natural join: Duplicate attributes will NOT be removed from the results
2. In general, join condition in a theta join can also be inequalities

Cartesian Product ×

Students

Name	Age
Alice	19
Bob	22

Courses

ID	Name
C1	DB
C2	Algo

results

Name	Age	ID	Name
Alice	19	C1	DB
Alice	19	C2	Algo
Bob	22	C1	DB
Bob	22	C2	Algo

→ Query: "Create a table that provides all possible student-course combinations"

→ Students × Donations

→ Effect: Theta join without a condition

Assignment :=

→ Useful for breaking down steps → solution easier to write & easier for others to understand

→ Example: $(\Pi_{name}\ Students) \cup (\Pi_{Name}\ Volunteer)$

→ Equivalent Representation

- $R1 := \Pi_{Name}\ Students$
- $R2 := \Pi_{Name}\ Volunteer$
- $R1 \cup R2$ → Note: All attribute names are retained

Rename ρ

Quiz 1

Name	Score
Alice	70
Bob	80
Cathy	90
David	100

→ Similar to assignment, but allows change of attribute names

Evaluation 1

Name	Score
Alice	70
Bob	80
Cathy	90
David	100

→ $\rho_{Evaluation1}\ Quiz1$

Eval 1

SName	QScore
Alice	70
Bob	80
Cathy	90
David	100

→ $\rho_{Eval1(SName, QScore)}\ Quiz1$

Duplicate Elimination δ

Name	Product	Date
Alice	iPhone	2017.01.01
Bob	Xbox	2017.01.01
Cathy	iPhone	2017.01.01
David	Xbox	2017.02.17

R1

Product
iPhone
Xbox
iPhone

R2

Product
iPhone
Xbox

→ Effect: Eliminate duplicate tuples

→ Query: Find the list of products sold on 2017.01.01

→ $R1 := \Pi_{Product}(\sigma_{Date='2017.01.01'}\ Purchase)$

→ $R2 := \delta(R1)$

Extended Projection Π

Scores

Name	Quiz1	Quiz2
Alice	70	90
Bob	90	80
Cathy	80	100
David	100	90

Results

Name	Total
Alice	160
Bob	170
Cathy	180
David	190

→ Similar to ordinary projection, but allows the creation of new attributes via arithmetic

→ Query: "For each student, find his/her total score in Quiz 1 & 2"

→ $\Pi_{Name,\ Quiz1+Quiz2 \to Total}\ Scores$ → if find avg: (Quiz1+Quiz2)/2 → Avg

→ LHS gives the arithmetic performed

→ RHS gives an attribute name to the result

Aggregate Functions

→ MAX(...), MIN(...), AVG(...), SUM(...), COUNT(...)

Grouping & Aggregation γ

Quiz 1

Name	School	Score
Alice	SCSE	90
Bob	EEE	80
Cathy	EEE	100
David	SCSE	90

Results

Max Score
100

→ Query: "Find the highest score in Quiz 1"

→ $\gamma_{MAX(Score) \rightarrow MaxScore}$ Quiz 1

Results

Min Score
80

→ Query: "Find the lowest score in Quiz 1"

→ $\gamma_{MIN(Score) \rightarrow MinScore}$ Quiz 1

Results

Avg Score
90

→ Query: "Find the average score in Quiz 1"

→ $\gamma_{AVG(Score) \rightarrow AvgScore}$ Quiz 1

Results

Sum Score
360

→ Query: "Find the sum of scores in Quiz 1"

→ $\gamma_{SUM(Score) \rightarrow SumScore}$ Quiz 1

Results

NumStu
4

→ Query: "Find the number of students who took Quiz 1"

→ $\gamma_{COUNT(Name) \rightarrow NumStu}$ Quiz 1 OR $\gamma_{COUNT(School) \rightarrow NumStu}$ Quiz 1

OR $\gamma_{COUNT(Score) \rightarrow NumStu}$ Quiz 1

Quiz 1

Name	School	Year	GPA
Alice	SCSE	3	4
Bob	EEE	1	3
Cathy	EEE	2	3.4
David	SCSE	3	3.6

Results

School	Year	GPA
SCSE	3	3.8
EEE	1	3
EEE	2	3.4

→ Query: "Find the avg GPA in each school & year"

→ $\gamma_{School, Year, AVG(GPA) \rightarrow AvgGPA}$ Quiz 1

→ Effect: Divide tuples into separate groups based on their "school, year" value combination, and then compute the average GPA in each group

Example:

Quiz 1

Name	Score
Alice	70
Bob	90
Cathy	80
David	100

R1

Max Score
100

→ Query: "Find the student that scores the highest in Quiz 1"

→ R1: $\gamma_{MAX(Score) \rightarrow MaxScore}$ Quiz 1

→ R2: Quiz 1 $\bowtie_{Score = MaxScore}$ R1

→ R3: $\pi_{Name}(R2)$

R2

Name	Score	MaxScore
David	100	100

$\sigma_{Score = MAX(Score)}$ Quiz 1 is WRONG.

→ Aggregate functions can only be used with the aggregate operation γ

R3

Name
David

Division ÷

Owns

Name	Product
Alice	iPad
Alice	iPhone
Bob	iPhone
Cathy	iPad

AppleP

Product
iPhone
iPad

Results

Name
Alice

Find Name that Owns all the Products in AppleP

→ Query: "Find each person that owns all Apple products"

→ Owns ÷ AppleP

→ $R_1(A,B) \div R_2(B)$ returns a table that contains only A

→ The table contains each A value in R_1 that is associated with every B value in R_2

→ Intuitive interpretation: "Find the A that R_1 all the B in R_2"

Owns

Name	Product
Alice	iPad
Alice	iPhone
Bob	iPhone
Cathy	iPad

AppleP

Product	Price
iPhone	999
iPad	699

Results

Name
Alice

→ Query: "Find each person that owns all Apple products"

→ Owns ÷ ($\pi_{Product}$ AppleP)

→ Owns ÷ AppleP is WRONG, "Price" doesn't appear in "Owns"

Example:

→ Owns (Name, Since, Product)
AppleP (Product)

→ Owns ÷ AppleP

- Results (Name, Since)
  - The result is a table with attributes in Owns but not in AppleP, i.e., Name & Since
  - The table contains every {Name, Since} combination that is associated with all Product in AppleP

Left Outer join $\overset{\circ}{\bowtie}_{L\ condition}$

Students

SName	School
Alice	SCSE
Bob	EEE
Cathy	CEE
David	SCSE

Donations

Name	Amount
Cathy	100
David	200
Eddie	300
Fred	400

Results

SName	Name	School	Amount
Alice	NULL	SCSE	NULL
Bob	NULL	EEE	NULL
Cathy	Cathy	CEE	100
David	David	SCSE	200

→ Query: "For each student, find the amount of his/her donation"

→ Students $\overset{\circ}{\bowtie}_{L\ SName = Name}$ Donations → if Students.SName = Students.Name, then there is no need to put a condition

→ All attributes are retained

→ For each student that has not made a donation, a "NULL" value is given as his/her Amount

Right Outer join $\overset{\circ}{\bowtie}_{R\ condition}$

Students

SName	School
Alice	SCSE
Bob	EEE
Cathy	CEE
David	SCSE

Donations

Name	Amount
Cathy	100
David	200
Eddie	300
Fred	400

Results

SName	Name	School	Amount
Cathy	Cathy	SCSE	100
David	David	EEE	200
NULL	Eddie	NULL	300
NULL	Fred	NULL	400

→ Query: "For each donator, find the school he/she is in"

→ Students $\overset{\circ}{\bowtie}_{R\ SName = Name}$ Donations → if Students.SName = Students.Name, then there is no need to put a condition

→ All attributes are retained

→ For each student that has not made a donation, a "NULL" value is given as his/her Amount

Full Outer join $\overset{\circ}{\bowtie}_{condition}$

Students

SName	School
Alice	SCSE
Bib	EEE
Cathy	CEE
David	SCSE

Donations

Name	Amount
Cathy	100
David	200
Eddie	300
Fred	400

Results

SName	Name	School	Amount
Alice	NULL	SCSE	NULL
Bob	NULL	EEE	NULL
Cathy	Cathy	CEE	100
David	David	SCSE	200
NULL	Eddie	NULL	300
NULL	Fred	NULL	400

→ Combination of left & right outer joins

→ Students $\overset{\circ}{\bowtie}_{SName = Name}$ Donations

Example:

CastIn

Name	Movie	Year
John	Batman	2012
Steve	Batman	2012
Meg	The Women	2008

Stars

Name	Gender	Birth
John	Male	1980
Meg	Female	1981
Steve	Male	1990

→ For each movie, count the number of male movie stars that were cast in the movie

→ $R1 := \sigma_{Gender = 'Male'}$ Stars

→ $R2 := CastIn \overset{\circ}{\bowtie}_{L} R1$

Name	Movie	Year	Gender	Birth
John	Batman	2012	MALE	1980
Steve	Batman	2012	MALE	1990
Meg	The Women	2008	NULL	NULL

→ $R3 := \gamma_{movie, COUNT(Gender) \rightarrow MaleStars}(R2)$

Movie	MaleStars
Batman	2
The Women	1

→ WRONG; counting rows instead of actual male stars

R2

Name	Male
John	1
Steve	1

Correct solution:

→ $R1 := \sigma_{Gender = 'Male'}$ Stars

→ $R2 := \gamma_{Name, COUNT(Gender) \rightarrow Male}(R1)$

→ $R3 := CastIn \overset{\circ}{\bowtie}_{L} R2$

→ $R4 := \gamma_{movie, SUM(male) \rightarrow MaleStars}$ R3

R3

Name	Movie	Year	Male
John	Batman	2012	1
Steve	Batman	2012	1
Meg	The Women	2008	NULL

R4

Movie	MaleStars
Batman	2
The Women	0