CptS 451- Introduction to Database Systems

Relational Algebra

(ch-4.1 and ch-4.2)

Instructor: Sakire Arslan Ay

Washington State University

World Class. Face to Face.

WSU CptS 451

Outline

- Relational Algebra
  - Basic operations
- Extended Relational Algebra
  - Outerjoins, Grouping/Aggregation

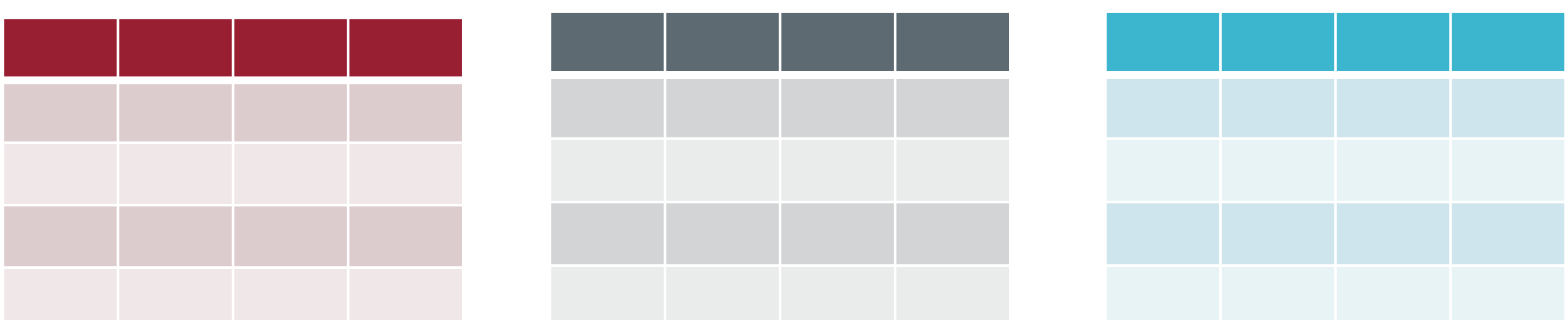
What is “Algebra”?

- 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.
  - Example: (x-5)/y*z-3

What is Relational Algebra?

- An algebra whose operands are relations or variables that represent relations.
- Operators are designed to do the most common things that we need to do with relations in a database.
  - Relational algebra is used as a *query language* for relations.
  - Provides a theoretical foundation for SQL
    - Operations are implemented by the SQL query language

- Queries return relations
  - Closure
  - Compositionality



Core Relational Algebra

WSU CptS 451

- Union, intersection, and difference.
  - Usual set operations, but ***both operands must have the same relation schema***.
- Selection (σ): picking certain rows.
- Projection(Π): picking certain columns.
- Products (X) and joins($\bowtie_C$): compositions of relations.
- Renaming of relations and attributes.

- We will use the RelaX - relational algebra calculator to test relational algebra expressions:
  - https://dbis-uibk.github.io/relax/calc.htm?data=gist:93fc0fcb193d2ca528edbd2470fce7a1

Union ∪, Intersection ∩, Difference –

Set operators. Relations must have the same schema.

R(name, dept)

Name	Dept
John	Physics
Tom	EECS

S(name, dept)

Name	Dept
John	Physics
Shira	Math

R∪S

Name	Dept
John	Physics
Tom	EECS
Shira	Math

R ∩ S

Name	Dept
John	Physics

R-S

Name	Dept
Tom	EECS

Selection σ

σ_c **(R):** return tuples in **R** that satisfy condition ***C***.

- **C** is a condition (as in "if" statements) that refers to attributes of **R**.
- *Schema* of result is same as that of the input relation **R.**

Emp (name, dept, salary)

Name	Dept	Salary
Jane	EECS	30K
Jack	Physics	30K
Tom	EECS	75K
Joe	Math	40K
Jack	Math	50K

"Employees who earn more than 35K"

$\sigma_{\text{salary>35K}}$ (Emp)

Name	Dept	Salary
Tom	EECS	75K
Joe	Math	40K
Jack	Math	50K

"EECS Employees who earn less than 40K"

$\sigma_{\text{dept='eecs' and salary<40K}}$ (Emp)

Name	Dept	Salary
Jane	EECS	30K

Projection Π

$\Pi_{A1,\ldots,Ak}(R)$: pick columns of attributes A1,…,Ak of R. eliminate duplicate tuples, if any.

Emp (name, dept, salary)

Name	Dept	Salary
Jane	EECS	30K
Jack	Physics	30K
Tom	EECS	75K
Joe	Math	40K
Jack	Math	50K

"Names and departments of employees"

$\Pi_{name,dept}$ (Emp)

Name	Dept
Jane	EECS
Jack	Physics
Tom	EECS
Joe	Math
Jack	Math

"Names of employees"

Π_{name} (Emp)

Name
Jane
Jack
Tom
Joe

Duplicates ("Jack") eliminated.

Extended Projection

$\Pi_{A1,A2+A3\text{->}B}(R)$: the attribute list may contain arbitrary expressions involving attributes:

1. Operations on attributes, e.g., *A1+A2->B*.
2. Duplicate occurrences of the same attribute.

Grades (ID, Exam1, Exam2)

ID	Exam1	Exam2
1254	80	70
1260	100	90
1275	65	70
1279	71	81

"IDs and average exam grades of students."

$\Pi_{ID,(Exam1+Exam2)/2\text{->}Avg}(Grades)$

ID	Avg
1254	75
1260	95
1275	65
1279	71

The calculated attribute is named 'Avg'

Cartesian Product: ×

R × **S**: pair each tuple **r** in **R** with each tuple **s** in **S**.

- *Schema* of result is the attributes of R and then S, in order.
- If attribute *A* exists both in R and S then use R.*A* and S.*A*.

Emp (name, dept)

Name	Dept
Jack	Physics
Tom	EECS

Contact(name, addr)

Name	Addr
Jack	Pullman
Tom	Moscow
Mary	Colfax

Emp × Contact

Emp.Name	Dept	Contact.Name	Addr
Jack	Physics	Jack	Pullman
Jack	Physics	Tom	Moscow
Jack	Physics	Mary	Colfax
Tom	EECS	Jack	Pullman
Tom	EECS	Tom	Moscow
Tom	EECS	Mary	Colfax

Join

$$R \bowtie_{C} S = \sigma_{c}(R \times S)$$

- Take the product **R** X **S**, then apply σ_c to the result.
- Join condition *C* is of the form:

  <*cond_1*> AND <*cond_2*> AND … AND <*cond_k*>

  Each "*cond_i*" is of the form **A *op* B**, where:
  - **A** is an attribute of **R**, **B** is an attribute of **S**
  - ***op*** is a comparison operator: =, <, >, $\geq$, $\leq$, or $\neq$.
- Different types:
  - Theta-join
  - Equi-join
  - Natural join

Theta-Join

$$R \underset{R.A>S.C}{\bowtie} S$$

R(A,B)

A	B
3	4
5	7

S(C,D)

C	D
2	7
6	8

$R \times S$

A	B	C	D
3	4	2	7
~~3~~	~~4~~	~~6~~	~~8~~
5	7	2	7
~~5~~	~~7~~	~~6~~	~~8~~

Result

A	B	C	D
3	4	2	7
5	7	2	7

Theta-Join

$$R \underset{R.A>S.C \text{ and } R.B \neq S.D}{\bowtie} S$$

R(A,B)

A	B
3	4
5	7

S(C,D)

C	D
2	7
6	8

$R \times S$

A	B	C	D
3	4	2	7
~~3~~	~~4~~	~~6~~	~~8~~
~~5~~	~~7~~	~~2~~	~~7~~
~~5~~	~~7~~	~~6~~	~~8~~

Result

A	B	C	D
3	4	2	7

Equi-Join

- Special kind of theta-join: ***C*** only uses the equality operator.

R(A,B)

A	B
3	4
5	7

S(C,D)

C	D
2	7
6	8

$$R \bowtie_{R.B=S.D} S$$

R.A	R.B	R.C	R.D
5	7	2	7

Natural-Join $\bowtie$

- Connects two relations by:
  - Equating attributes of the same name (Equi-Join), and
  - Projecting out one copy of each pair of equated attributes.
- Relations R and S. Let *L* be the union of their attributes. And let A1,…,Ak be their common attributes.

$$R \bowtie S = \Pi_{L} (R \bowtie_{R.A1=S.A1 \text{ and} \ldots, R.Ak=S.Ak} S)$$

Emp(name,dept)

Dept	Name
Physics	Jack
EECS	Tom

Contact(name, addr)

Name	Addr
Jack	Pullman
Tom	Moscow
Mary	Colfax

Emp $\bowtie$ Contact: all employee names, depts, and addresses.

Dept	Name	Addr
Physics	Jack	Pullman
EECS	Tom	Moscow

Natural Join – Exercise1

WSU CptS 451

- *Names and GPAs of students with SAT>1200 who applied to CS and were rejected*

$$\Pi_{sName,GPA}(\sigma_{(SAT>1000)\ AND\ (major='CS')\ AND\ (decision='R')}(Student \bowtie Apply))$$

College

cName	state	enrolment

Student

sID	sName	GPA	SAT

Apply

sID	cName	major	decision

Student ⋈ Apply

sID	sName	GPA	SAT	cName	major	decision

https://dbis-uibk.github.io/relax/calc.htm?data=gist:b58990c762b4e2e168a57f22a4d0bb51

Natural Join – Exercise2

– *Names and GPAs of students with GPA>3.5 who applied to CS at colleges with enrollment>20,000 and were rejected*

$$\Pi_{\text{sName,GPA}}\left(\sigma_{\text{GPA>3.5 AND major='CS' AND enrollment>20000 AND decision='R'}}((\text{Student} \bowtie \text{Apply}) \bowtie \text{College})\right)$$

College

cName	state	enrolment

Student

sID	sName	GPA	SAT

Apply

sID	cName	major	decision

Student $\bowtie$ Apply $\bowtie$ College

sID	sName	GPA	SAT	cName	major	decision	state	enrollment

Renaming ρ

- <u>Motivation</u>: disambiguate relation and attribute names.
  – e.g., in R × R, how to differentiate the attributes from the two instances?

$$\rho_{\mathbf{S}(B1,\ldots,Bn)}(R)$$

S is a relation identical to **R**, with new attributes B1,…,Bn.

$$\rho_{emp1(name1,dept1)}(Emp)$$

Emp(name,dept)

Name	Dept
Jack	Physics
Tom	EECS

Emp1(name1, addr1)

Name1	Dept1
Jack	Physics
Tom	EECS

Renaming ρ (cont)

Emp (name, dept)

Name	Dept
Jack	Physics
Tom	EECS
Mary	EECS

"List all employees who work in the same department as Tom."

$$\Pi_{\text{emp1.name}}\left(\rho_{\text{emp1(name1,dept1)}}(\text{emp}) \bowtie_{\text{emp1.dept1=emp.dept}} \sigma_{\text{name='Tom'}}(\text{emp})\right)$$

Name
Tom
Mary

Schemas of Results - Summary

- **Schema of each operation**:
  - $\cup$, $\cap$, $-$: same as the schema of the two relations (remember that the schema of the operands should be same)
  - Selection σ: same as the relation's schema
  - Projection Π: attributes in the projection
  - Cartesian product **X** : attributes in two relations, use prefix to avoid confusion
  - Theta Join $\bowtie_C$: same as **X**
  - Natural Join $\bowtie$: union of relations' attributes, merge common attributes
  - Renaming ρ : new renamed attributes

Building Complex Expressions

- Three notations (similar to arithmetic):
    1. Expressions with several operators
        - Combine operators with parentheses and precedence rules
    2. Sequences of assignment statements
    3. Expression trees

1. Expressions with Several Operators

- Examples:

  $$R \bowtie S = \Pi_L (R \bowtie_{R.A1=S.A1,\ldots,R.Ak=S.Ak} S)$$

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

- Precedence of relational operators:
  1. () parenthesis
  2. [σ, π, ρ] (highest).
  3. [X, ⋈].
  4. ∩.
  5. [∪, —]

2. Sequences of Assignments

- Motivation: expressions can be complicated
- Introduce names for intermediate relations, using the assignment operator "="
- Then a query can be written as a sequential program consisting of a series of assignments

- Example:

$$\prod_{balance}(\sigma_{custssn=ssn}(account \times (\sigma_{name=tom} customer)))$$

R1 = $\sigma_{name=tom}$ (customer)

R2 = $\sigma_{custssn=ssn}$ (account$\times$R1)

Result = $\Pi_{balance}$ (R2)

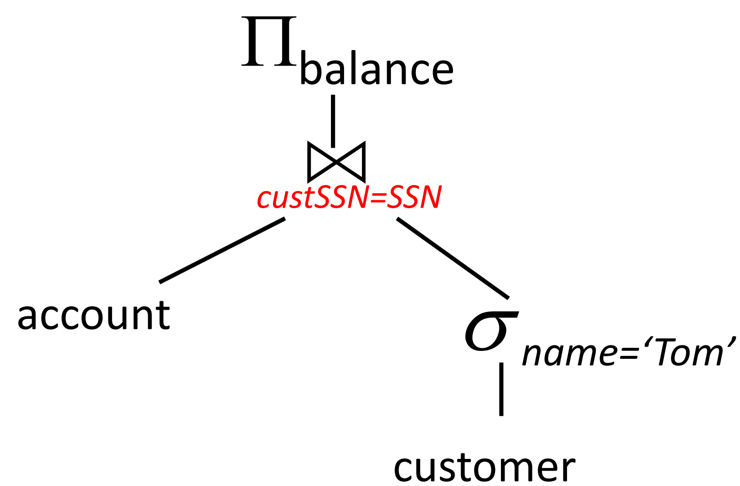
3.Expression Trees

- Leaves are operands --- variables standing for relations.
- Interior nodes are operators, applied to their child or children.

Example: "List account balance of Tom."

customer(SSN, name, city)
account(custSSN, balance)

$$\Pi_{balance}(account \bowtie_{custSSN=SSN} \sigma_{name='Tom'}(customer))$$



Expression Trees – Example1 (Alternative Solution)

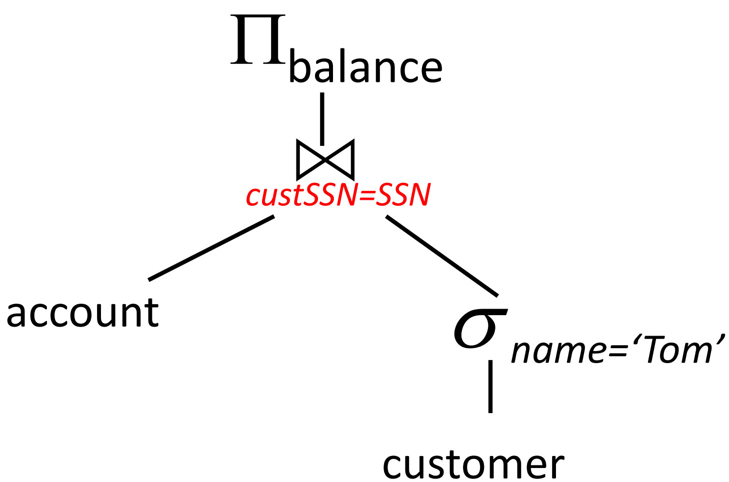
customer(SSN, name, city)

account(custSSN, balance)

- "List account balance of Tom."

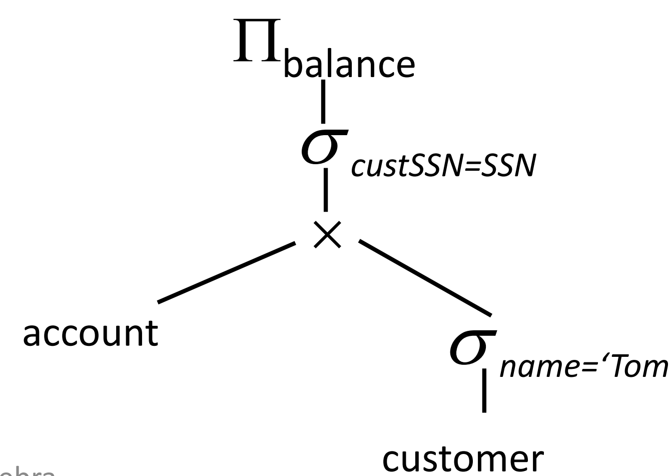
Solution 1:

$$\Pi_{balance}\left(account \bowtie_{custSSN=SSN} \sigma_{name='Tom'}(customer)\right)$$



Solution 2:

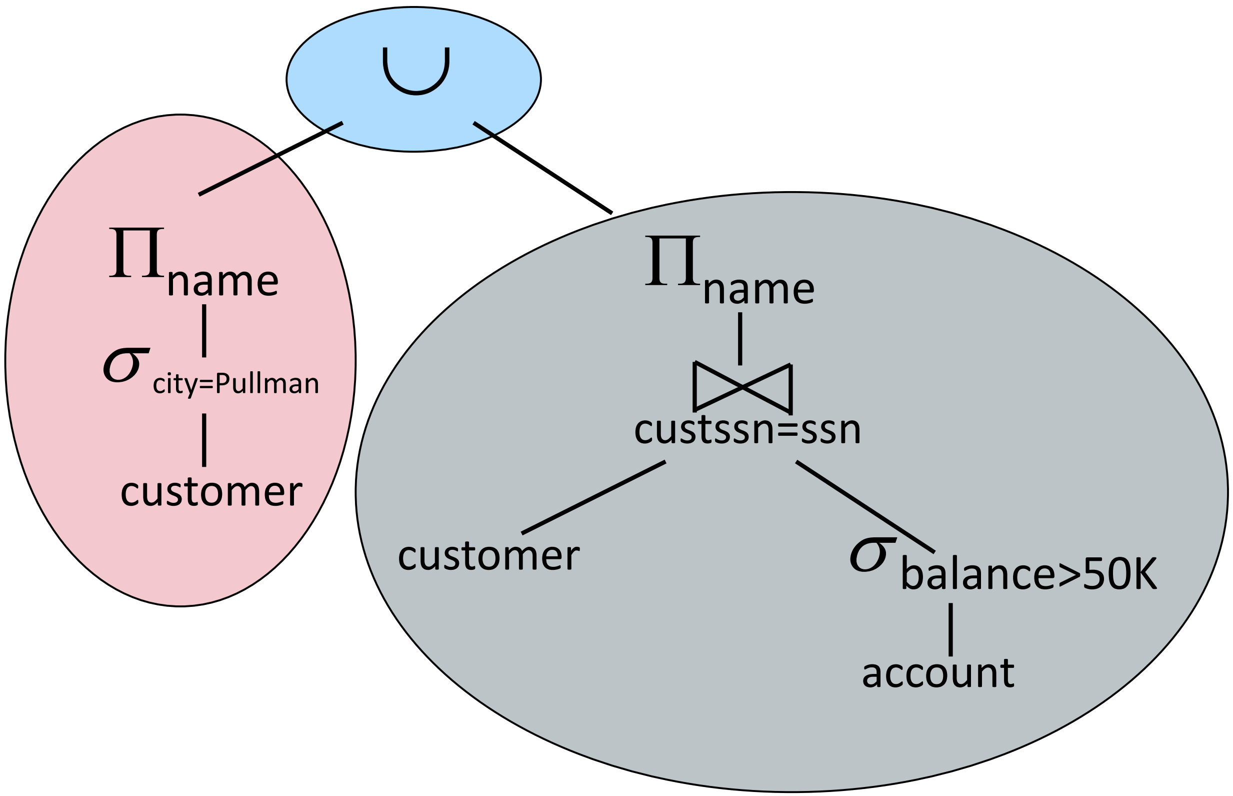
$$\Pi_{balance}\left(\sigma_{custSSN=SSN}\left(account \times \sigma_{name='Tom'}(customer)\right)\right)$$



Expression Trees – Example2

customer(ssn, name, city)
account(custssn, balance)

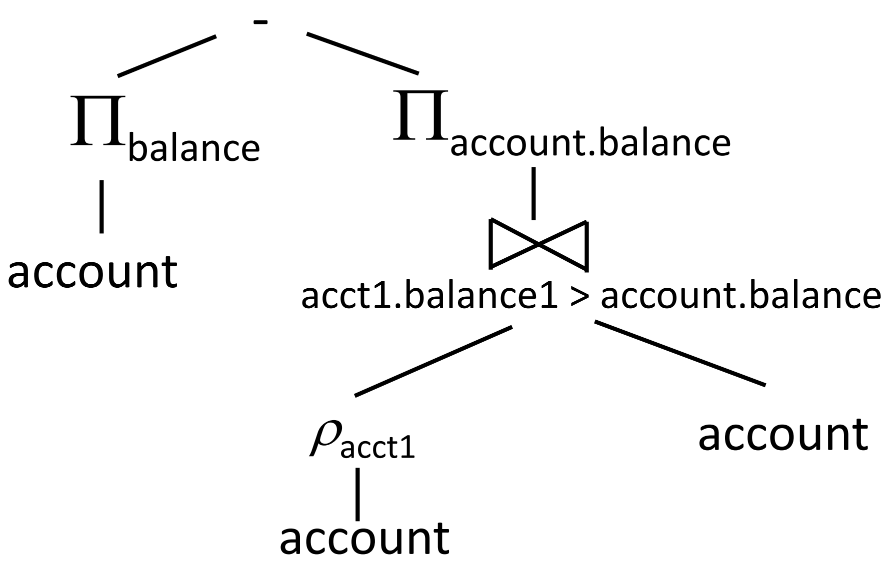
- Find names of customers in Pullman ***or*** having a balance > 50K.



Example 3

customer(ssn, name, city)
account(custssn, balance)

- Find the highest balance among all the customers.



https://dbis-uibk.github.io/relax/calc.htm?data=gist:528fb4e46355f4b8e9911fdd3376f73d

Relational Operations on Bags

Bag Semantics

- Set semantics: no duplicates
  - {1,2,1,3} is a bag.
  - {1,2,3} is also a bag that happens to be a set.
- Bag semantics: duplicates possible, no order
  - SQL (mostly) uses the bag semantics: a table could have duplicated tuples
  - Reason: efficient processing.
    - Some operations, like projection, are more efficient on bags than sets

R(name, dept)

Name	Dept
Jack	Physics
Tom	EECS
Tom	EECS

A bag.

Operations on Bags

- **Bag Union**: Combine elements from two relations
  - E.g., $\{1,3,7\} \cup \{3,5,6\}=\{1,3,3,5,6,7\}$
- **Bag intersection**: take the minimum of the number of occurrences in each bag
  - E.g.: $\{1,1,2,2,3\} \cap \{1,2,2,2,3,4\}=\{1,2,2,3\}$
- **Bag difference**: proper-subtract the number of occurrences in the two bags
  - E.g.: $\{1,1,1,2,2,3\}-\{1,2,3,4\}=\{1,1,2\}$

Operations on Bags

- Selection and projection applies to each tuple independently. Duplicates are not eliminated in the result.

- Products and joins are done on each pair of tuples, regardless of whether it is a duplicate or not. The duplicates are not eliminated in the final result.

Operations on Bags

Example: Bag Selection

R(A,B)

A	B
1	2
5	6
1	2

$\sigma_{A+B<5}$ (R)

A	B
1	2
1	2

Example: Bag Projection

R(A,B)

A	B
1	2
5	6
1	2

π_{A} (R)

A
1
5
1

Operations on Bags

WSU CptS 451

Example: Bag Product

R(A,B)

A	B
1	2
5	6
1	2

S(B,C)

B	C
2	4
7	8

RXS

R.A	R.B	S.B	S.C
1	2	2	4
1	2	7	8
5	6	2	4
5	6	7	8
1	2	2	4
1	2	7	8

Operations on Bags

WSU CptS 451

Example: Bag Natural-Join

R(A,B)

A	B
1	2
5	6
1	2

S(B,C)

B	C
2	4
7	8

$R \bowtie S$

A	B	C
1	2	4
1	2	4

Example: Bag Theta-Join

R(A,B)

A	B
1	2
5	6
1	2

S(B,C)

B	C
2	4
7	8

$R \bowtie_{R.B \leq S.B} S$

R.A	R.B	S.B	S.C

Laws for Bags

- Some laws for sets still hold for bags
  - E.g., union and intersection are still commutative and associative
  - $R \cap S = S \cap R$, $(R \cap S) \cap T = R \cap (S \cap T)$
- Some laws for sets might not hold for bags:
  1. $R \cap (S \cup T) = (R \cap S) \cup (R \cap T)$ is true for sets

     But it is **not true** for bags. E.g.:
     - $R=S=T=\{1\}$
     - $R \cap (S \cup T)=\{1\}$
     - $(R \cap S) \cup (R \cap T)=\{1,1\}$
  2. $S \cup S=S$ is true for sets

     But it is **not true** for bags

Extended Operators of Relational Algebra

Extended Relational Algebra

- Basic relational algebra is using the set semantics
- Extended relational algebra and SQL is using bag semantics.
- More operations needed:
  - Duplicate-elimination operator δ
  - Sorting operator τ
  - Grouping and aggregation operator γ

Duplicate elimination δ

- δ(R) = relation with one copy of each tuple that appears one or more times in R

R

A	B
1	2
3	4
3	4

δ (R)

A	B
1	2
3	4

Sorting τ

- $\tau_L(R)$ = sort the records in R according to attributes on list L (i.e., L= {A1,A2,..Ak}
  - Sort first on the value of A1, then A2, and so on.

R

A	B
5	2
3	7
3	4

$\tau_A(R)$

A	B
3	7
3	4
5	2

$\tau_{A,B}(R)$

A	B
3	4
3	7
5	2

Aggregation

- Aggregation function takes a collection of values and returns a single value as a result.
  - avg: average value
  - min: minimum value
  - max: maximum value
  - sum: sum of values
  - count: number of values

Account

Name	Branch	Balance
Tom	Pullman	10K
Tom	Seattle	20K
Mary	Seattle	5K
Jack	Pullman	20K

COUNT(*)
4

MAX(Balance)
20K

SUM(Balance)
55K

MIN(Balance)
5K

AVG(Balance)
13.75K

* refers to complete list of attributes

Grouping

- Aggregate operation applied to an attribute calculates the value for the entire column.
- Grouping allows to consider the tuples in a relation as groups (corresponding to the value of one or more attributes) and aggregate only within groups.
- Example:

Account

Name	Branch	Balance
Tom	Pullman	10K
Tom	Seattle	20K
Mary	Seattle	5K
Jack	Pullman	40K
Mark	Seattle	20K
James	Moscow	15K
Abby	Pullman	45K
Ashley	Moscow	13K

Group according to branch

Name	Branch	Balance
Tom	Seattle	20K
Mary	Seattle	5K
Mark	Seattle	20K
Tom	Pullman	10K
Jack	Pullman	40K
Abby	Pullman	45K
James	Moscow	15K
Ashley	Moscow	13K

Grouping and Aggregation γ

- $\gamma_{L,\theta(A)}(R)$
- Group R according to attributes on list L
- Within each group, do aggregation $\theta(A)$
- Result has one tuple for each group. It includes:
  1. The grouping attributes and
  2. Their group's aggregations.

First, group tuples in Accunt according to "Branch."

Account

Name	Branch	Balance
Tom	Seattle	20K
Mary	Seattle	5K
Mark	Seattle	20K
Tom	Pullman	10K
Jack	Pullman	40K
Abby	Pullman	45K
James	Moscow	15K
Ashley	Moscow	13K

Then do the aggregation within each group.

$\gamma_{\text{branch,Count(Name)->NumCust}}(\text{Account})$

Branch	NumCust
Pullman	3
Seattle	3
Moscow	2

$\gamma_{\text{branch,Sum(Balance)->Total}}(\text{Account})$

Branch	Total
Pullman	95K
Seattle	45K
Moscow	28K

Another Example: Grouping/Aggregation

R(A,B,C)

A	B	C
1	2	3
4	5	6
1	2	5

$\gamma_{A,B,AVG(C)\text{->}X}$ (R) = ??

First group R by A and B:

A	B	C
1	2	3
1	2	5
4	5	6

Then calculate average for C within groups:

A	B	X
1	2	4
4	5	6

Outer Join

- Motivation:

R(Dept,Name)

Dept	Name
Physics	Jack
EECS	Tom

S(Name,Addr)

Name	Addr
Jack	Pullman
Mike	Seattle
Mary	Spokane

- Variants of the outerjoin:
  - Full outer join (outer join)
  - Left outer join
  - Right outer join

Outer Join

- An extension of the natural join operation that avoids loss of information.
- Computes the join and then adds ("dangling") tuples form one relation that does not match tuples in the other relation to the result of the join.
- Variants of the outerjoin:
  - Full outer join $R \overset{o}{\bowtie} S$ (or just outer join)
  - Left outer join $R \overset{o}{\bowtie}_L S$
  - Right outer join $R \overset{o}{\bowtie}_R S$

Full Outer Join

WSU CptS 451

- Add dangling tuples from both left and right relations.
- Pad null values for the dangling tuples

R(Dept,Name)

Dept	Name
Physics	Jack
EECS	Tom

S(Name,Addr)

Name	Addr
Jack	Pullman
Mike	Seattle
Mary	Spokane

$R \bowtie S$

Dept	Name	Addr
Physics	Jack	Pullman

Natural Join

$R \overset{o}{\bowtie} S$

Dept	Name	Addr
Physics	Jack	Pullman
EECS	Tom	NULL
NULL	Mike	Seattle
NULL	Mary	Spokane

Full Outer Join

Left and Right Outer Join

- Left Outer Join: Add dangling tuples from left relation.
- Right Outer Join: Add dangling tuples from right relation.
- Pad NULL values for the dangling tuples

R(Dept,Name)

Dept	Name
Physics	Jack
EECS	Tom

S(Name,Addr)

Name	Addr
Jack	Pullman
Mike	Seattle
Mary	Spokane

$R \overset{o}{\bowtie}_L S$

Dept	Name	Addr
Physics	Jack	Pullman
EECS	Tom	NULL

Left Outer Join

$R \overset{o}{\bowtie}_R S$

Dept	Name	Addr
Physics	Jack	Pullman
NULL	Mike	Seattle
NULL	Mary	Spokane

Right Outer Join

Limitation of Relational Algebra

- Some queries cannot be represented
- Example, recursive queries:
  - Table `R(Parent,Child)`
  - How to find all the ancestors of “Tom”?
  - Impossible to write this query in relational algebra
- More expressive languages needed:
  - E.g., Datalog

Relational Algebra Example:

- Consider the following relational schema for a library (assume using bag semantics):

Member

member#	name	dob

Books

isbn	title	authors	publisher

Borrowed

member#	isbn	date

https://dbis-uibk.github.io/relax/calc.htm?data=gist:93fc0fcb193d2ca528edbd2470fce7a1

Write the following queries in relational algebra.

a. Find the names of members who have borrowed any book published by "McGrawHill".

Steps:

1. Find the books published by McGraw-Hill.
2. Find the member# s that borrowed the books in step1. Get the names for those members.

$$R1 = \sigma_{\text{publisher='McGraw-Hill'}}(\text{Books})$$

$$\text{Result} = \Pi_{\text{name}}(\text{Member} \bowtie (\text{Borrowed} \bowtie R1))$$

Relational Algebra Example:

Member

member#	name	dob

Books

isbn	title	authors	publisher

Borrowed

member#	isbn	date

b. Find the names of members who have borrowed more than five books published by "McGrawHill". A member can borrow the same book more than once.

$R1 = \sigma_{\text{publisher='McGraw-Hill'}}(\text{Books})$

$R2 = \sigma_{\text{numBooks>5}}\ (\gamma_{\text{member\#,COUNT(isbn)->numBooks}}\ (\text{Borrowed} \bowtie R1))$

$\text{Result} = \Pi_{\text{name}}(\text{Member} \bowtie R2)$

Relational Algebra Example:

Member

member#	name	dob

Books

isbn	title	authors	publisher

Borrowed

member#	isbn	date

c. Find the names and membership numbers of members who have borrowed more than one different books published by "McGrawHill". A member can borrow the same book more than once.

R1 := $\sigma_{\text{publisher='McGraw-Hill'}}(\text{Books})$

R2:=$\sigma_{\text{numBooks>1}}\ (\gamma_{\text{member\#,COUNT(isbn)->numBooks}}\ (\delta(\Pi_{\text{member\#,isbn}}(\text{Borrowed} \bowtie \text{R1}))))$

Result:= $\Pi_{\text{name,member\#}}\ (\text{Member} \bowtie \text{R2})$

Relational Algebra Example:

Member

member#	name	dob

Books

isbn	title	authors	publisher

Borrowed

member#	isbn	date

d. For each publisher, find the name and membership number of members who have borrowed more than five books of that publisher

$$R1 = \sigma_{numBooks>5} (\gamma_{publisher,member\#,COUNT(isbn)\text{->}numBooks} (Borrowed \bowtie Books))$$
$$Result := \Pi_{publisher,name,member\#} (Member \bowtie R1)$$

Relational Algebra Example:

Member

member#	name	dob

Books

isbn	title	authors	publisher

Borrowed

member#	isbn	date

e. Find the average number of books borrowed by members. Take into account that if a member does not borrow any books, then that member does not appear in the borrowed relation.

$$\gamma_{COUNT(isbn)} \text{(Borrowed)} \,/\, \gamma_{COUNT(member\#)}\, \delta(\Pi_{member\#} \text{(Borrowed)})$$

Relational Algebra Example:

Member

member#	name	dob

Books

isbn	title	authors	publisher

Borrowed

member#	isbn	date

f. Find the members who didn't borrow any 'McGrawHill' books. Give the names and memberNo's of those members.

Relational Algebra Example:

WSU CptS 451

Member

member#	name	dob

Books

isbn	title	authors	publisher

Borrowed

member#	isbn	date

g. Find the members who borrowed the most number of books.

h. Find the pair of members who borrowed the same books.