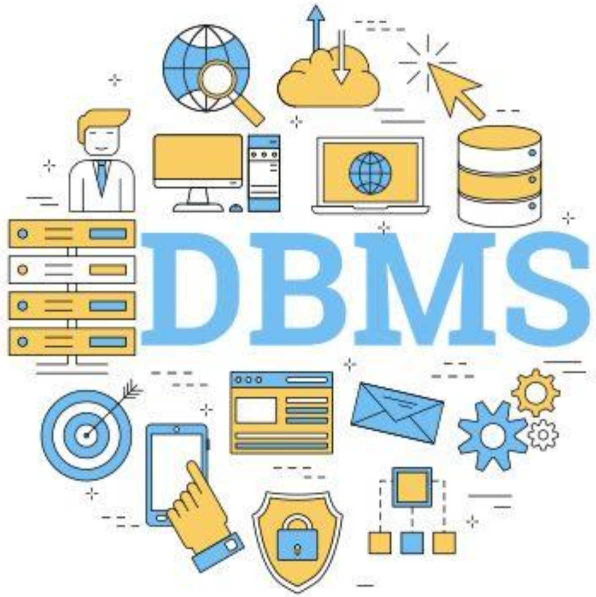
Database Systems Concepts and Design

CSC201S2/G2



Chapter 6: Relational Algebra

Outline

- Introduction for Relational Algebra
- Unary Relational Operations
- Relational Algebra Operations from Set Theory

Relational Algebra

- The basic set of operations for the relational model
- Enable the specification of basic retrievals
- Algebra operations thus produce new relations, which can be further manipulated the same algebra.
- A sequence of relational algebra operations forms a relational algebra expression,
- The result will also be a relation that represents the result of a database query

Definitions Relational Algebra

Domain: set of relations

Basic operators: select, project, union, set difference, Cartesian (cross) product

Derived operators: set intersection, division, join

Procedural: Relational expression specifies query by describing an algorithm for determining the result of an expression

Unary Relational Operation

SELECT Operation: select a subset of the tuples from a relation that satisfy a selection condition.

Examples:

$$\sigma_{DNO = 4}(\text{EMPLOYEE})$$

$$\sigma_{SALARY > 30,000}(\text{EMPLOYEE})$$

denoted by $\sigma_{\text{<selection condition>}}(R)$ where the symbol σ (sigma) is used to denote the select operator, and the selection condition is a Boolean expression specified on the attributes of relation R

Unary Relational Operation

The SELECT operation $\sigma_{<\text{selection condition}>}(R)$ produces a relation S that has the same schema as R

The SELECT operation σ is commutative;

$\sigma_{<\text{condition1}>}(\sigma_{<\text{condition2}>}(R)) = \sigma_{<\text{condition2}>}(\sigma_{<\text{condition1}>}(R))$

A cascaded SELECT operation may be applied in any order

$\sigma_{<\text{condition1}>}(\sigma_{<\text{condition2}>}(\sigma_{<\text{condition3}>}(R)) = \sigma_{<\text{condition2}>}(\sigma_{<\text{condition3}>}(\sigma_{<\text{condition1}>}(R)))$

A cascaded SELECT operation may be replaced by a single selection with a conjunction of all the conditions

$\sigma_{<\text{condition1}>}(\sigma_{<\text{condition2}>}(\sigma_{<\text{condition3}>}(R)) = \sigma_{<\text{condition1}>\ \text{AND}\ <\text{condition2}>\ \text{AND}\ <\text{condition3}>}(R)))$

Selection Condition

Operators: $<, \leq, \geq, >, =, \neq$

Simple selection condition:

- <attribute> operator <constant>
- <attribute> operator <attribute>
- <condition> AND <condition>
- <condition> OR <condition>
- NOT <condition>

Examples

Person

Id	*Name*	*Address*	*Hobby*
1123	John	123 Main	stamps
1123	John	123 Main	coins
5556	Mary	7 Lake Dr	hiking
9876	Bart	5 Pine St	stamps

$\sigma_{\,Id>3000 \text{ OR } Hobby=\text{'hiking'}}$ (Person)

$\sigma_{\,Id>3000 \text{ AND } Id<3999}$ (Person)

$\sigma_{\,\text{NOT}(Hobby=\text{'hiking'})}$ (Person)

$\sigma_{\,Hobby\neq\text{'hiking'}}$ (Person)

Unary Relational Operation

- **PROJECT Operation:** selects certain *columns* from the table and discards the others.

  Example: $\pi_{\text{LNAME, FNAME,SALARY}}(\text{EMPLOYEE})$

  The general form of the project operation is: $\pi_{<\text{attribute list}>}(R)$ where π is the symbol used to represent the project operation and <attribute list> is the desired list of attributes.

  PROJECT *removes duplicate tuples*, so the result is a set of tuples and hence a valid relation.

Example

(a) $\sigma_{\text{(DNO=4 AND SALARY>25000) OR (DNO=5 AND SALARY>30000)}}\text{(EMPLOYEE)}$

(b) $\pi_{\text{LNAME, FNAME, SALARY}}\text{(EMPLOYEE)}$

(c) $\pi_{\text{SEX, SALARY}}\text{(EMPLOYEE)}$

(a)

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
Franklin	T	Wong	333445555	1955-12-08	638 Voss,Houston,TX	M	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry,Bellaire,TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 FireOak,Humble,TX	M	38000	333445555	5

(b)

LNAME	FNAME	SALARY
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000
Jabbar	Ahmad	25000
Borg	James	55000

(c)

SEX	SALARY
M	30000
M	40000
F	25000
F	43000
M	38000
M	25000
M	55000

Exercise

- Consider the following relations

DEPOSIT

Branch name	Account number	Customer name	Balance
Downtown	101	Johnson	500
Mianus	215	Smith	700
Perry ridge	102	Hayes	400
Round Hill	305	Turner	350
Perry ridge	201	Williams	900
Redwood	222	Lindsay	700
Brighton	217	Green	750
Downtown	105	Green	850

BORROW

Branch name	Loan number	Customer name	Amount
Downtown	17	Jones	1000
Redwood	23	Smith	2000
Perry ridge	15	Hayes	1500
Downtown	14	Jackson	1500
Mianus	93	Curry	500
Round Hill	11	Turner	900
Pownal	29	Williams	1200
North Town	16	Adams	1300
Downtown	18	Johnson	2000
Perry ridge	25	Glenn	2500
Brighton	10	Brooks	2200

CLIENT

Customer name	Banker name
Turner	Johnson
Hayes	Jones
Johnson	Johnson

Exercise

- Consider the following relations

CUSTOMER

Customer name	Street	Customer city
Jones	Main	Harrison
Smith	North	Rye
Hayes	Main	Harrison
Curry	North	Rye
Lindsay	Park	Pittsfield
Turner	Putnam	Stamford
Williams	Nissan	Princeton
Adams	Spring	Pittsfield
Johnson	Alma	Palo Alto
Glenn	Sand Hill	Woodside
Brooks	Senator	Brooklyn
Green	Walnut	Stamford

BRANCH

Branch name	Assets	Branch city
Downtown	900000	Brooklyn
Redwood	2100000	Palo Alto
Perry ridge	1700000	Horse neck
Mainus	400000	Horse neck
Round Hill	800000	Horse neck
Pownal	300000	Bennington
North Town	3700000	Rye
Brighton	7100000	Brooklyn

Exercise

Write relational algebra statements to do following:

1. Find the **names** of all **branches** in the **deposit relation**?
2. Find all tuples **having an account at the Perry ridge branch**?
3. Find all tuples **having a loan from the Perry ridge branch**?
4. Find all tuples in which the **amount borrowed is more than $1200**?
5. Find those tuples pertaining loans of **more than $1200 made by the Perry ridge branch**?
6. Find all those customers who have the **same name as their personal banker**?

Exercise

7.

a. Find **all customers with a loan at the Perry ridge branch**?

b. Find **all customers with an account at the Perry ridge**?

c. Find **everyone who has a loan, an account, or both**?

8. Find all customers of the Perry ridge branch who **have an account there but not a loan**?

9. Find all customers with **both a loan and an account at the Perry ridge branch**?

Answers

1. $\pi_{Branch\ Name}$(Deposit)
2. $\sigma_{Branch\ Name='Perry\ ridge'}$(Deposit)
3. $\sigma_{Branch\ Name='Perry\ ridge'}$(Borrow)
4. $\sigma_{amount>\$1200}$(Borrow)
5. $\sigma_{Branch\ Name='Perry\ ridge'\ AND\ amount>\$1200}$ (Borrow)
6. $\sigma_{Customer\ name=Banker\ Name}$(Client)
7. $\pi_{Customer\ Name}(\sigma_{Branch\ Name='Perry\ ridge'}$(Borrow))
8. $\pi_{Customer\ Name}(\sigma_{Branch\ Name='Perry\ ridge'}$(Deposit))

Answers

9. $\pi_{Customer\ Name}(\text{Deposit}) \cup \pi_{Customer\ Name}(\text{Borrow})$

10. $\pi_{Customer\ Name}(\sigma_{Branch\ Name='Perry\ ridge'}(\text{Deposit})) -$

$\pi_{Customer\ Name}(\sigma_{Branch\ Name='Perry\ ridge'}(\text{Borrow}))$

11. $\pi_{Customer\ Name}(\sigma_{Branch\ Name='Perry\ ridge'}(\text{Deposit})) \cap$

$\pi_{Customer\ Name}(\sigma_{Branch\ Name='Perry\ ridge'}(\text{Borrow}))$

Relational Algebra Operations from Set Theory

- The UNION, INTERSECTION, and MINUS Operations
- The CARTESIAN PRODUCT (or CROSS PRODUCT) Operation

Relational Algebra Operations from Set Theory

- A relation is a set of tuples, so set operations apply: ∩, ∪, - (set difference)
- Result of combining two relations with a set operator is a relation => all elements are tuples with the same structure

Cartesian product (R×S)

- Defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S.

- Notation ***r* x *s***

Cartesian product (R×S)

- Defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S.

- Notation ***r* x *s***

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

r x s

Cartesian product (R×S)

Build expressions using multiple operations

Example:

- r x s

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

- $\sigma_{A=C}(r \times s)$

A	B	C	D	E
α	1	α	10	a
β	2	β	20	a
β	2	β	20	b

Rename Operation ($\rho_x(E)$)

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.

Example: $\rho_x(E)$ returns the expression *E* under the name *X*

- If a relational-algebra expression *E* has arity *n*, then

$$\rho_{x\ (A1,\ A2,\ \ldots,\ An)}(E)$$

returns the result of expression *E* under the name *X*, and with the attributes renamed to *A1, A2, …., An*.

Union Compatible Relation

- Two relations are *union compatible* if
  - Both have **same number of columns**
  - **Names of attributes are the same in both**
  - Attributes with the same name in both relations have the **same domain**
- Union compatible relations can be combined using *union*, *intersection*, and *set difference*

Set Intersection (∩)

- $R \cap S$
  - Defines a relation consisting of the set of all tuples that are in both R and S.
  - R and S must be **union-compatible**.

- Expressed using basic operations: $R \cap S = R - (R - S)$

Example

- Relation r, s:

r

A	B
a	1
b	2
c	1

s

A	B
b	2
c	3

$r \cap s$

A	B
b	2

List all cities where there is both a branch office and at least one property for rent.

$$\Pi_{city}(\textbf{Branch}) \cap \Pi_{city}(\textbf{PropertyForRent})$$

city
Aberdeen
London
Glasgow

Union operation (∪)

- $R \cup S$
  - Union of two relations R and S defines a relation that contains all the tuples of **R, or S, or both R and S, duplicate tuples being eliminated**.
  - R and S must be **union-compatible.**

- If **R and S have *I* and *J* tuples, respectively, union is obtained by concatenating them into one relation with a maximum of (*I* + *J*) tuples.**

For $r \cup s$ to be valid:

- **r, s must have the same number of attributes**
- **The attribute domains must be compatible**

Example:

Relations *r*, *s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	1
α	2
β	1
β	3

$r \cup s$:

Consider relational schemas:

Depositor(customer_name, account_number)

Borrower(customer_name, loan_number)

Find all customers with either an account or a loan

$\Pi_{customer\text{-}name}$ ***(depositor)*** $\cup\ \Pi_{customer\text{-}name}$ ***(borrower)***

Set Difference (-)

- R – S
  - Defines a relation consisting of the tuples that are in relation R, but not in S.
  - R and S must be union-compatible.

List all cities where there is a branch office but no properties for rent.

$$\pi_{city}(\mathbf{Branch}) - \pi_{city}(\mathbf{PropertyForRent})$$

city
Bristol

Related SQL Operations

```
SELECT column1, column2
FROM table1
UNION/UNION ALL/INTERSECT/EXCEPT
SELECT column1, column2
FROM table2;
```

Exercise

- Perform all set operations for the above relations and state the results
- State related SQL queries for each operations

Company 1

employee_id	employee_name	employee_city
1	Bhim Shekh	Surat
2	Mehul Mohan	Goa
3	Palash Yadav	Ahmedabad
4	Ela Shikha	Delhi
5	Mrinal Thakur	Bangalore
6	Sitara Vani	Bangalore
7	Mohak Jain	Gurugram
8	Adesh Patel	Jaipur
9	Kunal Tandon	Delhi
10	Romit Soni	Mumbai

Company 2

employee_id	employee_name	employee_city
1	Sahdev Ramiah	Raipur
2	Mehul Mohan	Goa
3	Mohak Jain	Gurugram
4	Pragun Sarika	Chennai
5	Pooja Srivastava	Bangalore
6	Vani Shekhawat	Delhi
7	Mohak Jain	Ahmedabad
8	Neera Shah	Ahmedabad
9	Poonam Oberoi	Delhi
10	Abhishek Saini	Raipur

Exercise

- Perform Cartesian operations for the above relations and state the results
- State related SQL queries for each operations

Employee

employee_id	employee_name
1	Bhim Shekh
2	Palash Yadav
3	Ela Shikha
4	Mrinal Thakur

City

city_id	dept_id	dept_name
011	1	Finance
022	2	Admin
033	3	Intelligence
044	4	Serices

Join

- A join combines the rows of two or more tables based on related columns.
- Used for **retrieving the data from multiple tables simultaneously using related columns of tables**

JOIN operation denoted as $\mathbf{R3 <- \bowtie (R1)_{<join\_condition>} (R2)}$

$$\textbf{Example: Temp} <- \bowtie^{(\text{Student})}{}_{\text{S.roll=E.roll}}{}^{(\text{Exam})}$$

Example

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	HARSH	DELHI	XXXXXXXXXX	18
2	PRATIK	BIHAR	XXXXXXXXXX	19
3	RIYANKA	SILIGURI	XXXXXXXXXX	20
4	DEEP	RAMNAGAR	XXXXXXXXXX	18
5	SAPTARHI	KOLKATA	XXXXXXXXXX	19
6	DHANRAJ	BARABAJAR	XXXXXXXXXX	20
7	ROHIT	BALURGHAT	XXXXXXXXXX	18
8	NIRAJ	ALIPUR	XXXXXXXXXX	19

Student

COURSE_ID	ROLL_NO
1	1
2	2
2	3
3	4
1	5
4	9
5	10
4	11

StudentCourse

```
SELECT s.roll_no, s.name, s.address, s.phone, s.age,
sc.course_id
FROM Student s
JOIN StudentCourse sc ON s.roll_no = sc.roll_no;
```

Inner Join

- Combines two or more tables based on related columns and returns **only rows that have matching values among tables**.

- Inner join has three types.

  Conditional join

  Equi Join

  Natural Join

Conditional/ Theta Join

- Relations are combined based on the specified condition
- Condition can include <, >, <=, >=, ≠ operators in addition to the '=' operator

R	S
10	5
7	20

$A \bowtie_{S<T} B$

T	U
10	12
17	6

$\rightarrow$

R	S	T	U
10	5	10	12

Equi Join

- Join condition uses the equality operator ('=') between columns.

Column A	Column B
a	a
a	b

$$A \bowtie_{A.Column\ B\ =\ C.Column\ B} (C)$$

Column A	Column B
a	**a**
a	**c**

Column A	Column B
a	a

Natural Join

- Do not need any comparison operators. I
- Columns should have the **same name and domain**.
- **At least one common attribute** between the two tables.

Number	Square
2	4
3	9

$A \bowtie B$

Number	Cube
2	8
3	27

Number	Square	Cube
2	4	8
3	9	27

Exercise

Find all pairs of employees and departments where the department's location matches the employee's city.

Find employees who earn more than $55,000 and are in the HR department

Departments:

DepartmentID	Name	Location
101	HR	New York
102	Marketing	Chicago
103	IT	Seattle

Employees:

EmployeeID	Name	DepartmentID	Salary
1	Alice	101	50000
2	Bob	102	60000
3	Charlie	101	55000
4	David	103	52000

Exercise

Consider the following relation (all Primary keys are underlined).

- `Customer (accountId, lastName, firstName, street, city, state, zipCode,balance).`
- `Videotape (videoId. dateAcquired, movieId, storyId).`
- `Movie (movieId, title, genre, length, rating).`
- `Rental (accountId. videoId, dateRented, dateReturned, dateDue, cost).`
- `PreviousRental (accountId, videoId, dateRented, dateReturned, cost)`
- `Employee (ssn, lastName, firstName)`
- `Store (storeId, street, city, state, zipCode)`
- `Timecard (ssn, date, startTime, endTime, storeId, paid)`
- `HourlyEmployee (ssn, hourlyRate)`
- `WorksIn (ssn, storeId)`

Write relational algebra expression to find each of the following:

- All customers who live in California.
- All employees who work in store number 3.
- The names of all movies whose genre is comedy.
- The names of all customers who have rented a movie since 1st January I999.
- The employee name, date worked and hours for each time card that has more than eight hours worked.
- All videotapes currently rented by customer that live in Florida.
- All hourly employees who work in store number 3.
- All customers whose lastname is also the lastname of an employee.
- All employees who work in all stores.

Exercise

For each of the following queries, give an expression in the relational algebra. Consider the employee database given below:

- `Employee (employee_name, street, city)`
- `Works (employee_name, company_name, salary)`
- `Company (company_name, city)`
- `Manages (employee_name, manager_name)`

- Find the **names** of all **employees** who work for **First Bank Corporation**.
- Find the **names** and **cities** of residence of **all employees** who work for **First Bank Corporation**.
- Find the **names**, **street address** and **cities** of residence of all **employees** who work for **First Bank Corporation** and earn **more than $10,000**.
- Find the **names** of **employees** who live in the **same city as the company for which they work**.
- Find the **names** of all **employees** who **do not work for First Bank Corporation**.


Q&A