Chapter 6: Formal Relational Query Languages

Database System Concepts, 6th Ed.

Outline

- Relational Algebra
- Tuple Relational Calculus
- Domain Relational Calculus

Relational Algebra

- Queries in relational algebra are composed using a collection of operators, and each query describes a step-by-step procedure for computing the desired answer; that is, queries are specified in an operational manner.
- The inputs and outputs of a query are relations.
- Every operator in the algebra accepts (one or two) relation instances as arguments and returns a relation instance as the result.
- Each relational query describes a step-by-step procedure for computing the desired answer, based on the order in which operators are applied in the query.

Relational Algebra

- Procedural language
- Six basic operators
  - select: σ
  - project: Π
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ
- The operators take one or two relations as inputs and produce a new relation as a result.

SELECTION AND PROJECTION OPERATORS

- Relational algebra includes operators to select rows from a relation (σ) and to project columns (π). These operations allow us to manipulate data in a single relation. Consider the instance of the Sailors relation below, denoted as S2.

sid	*sname*	*rating*	*age*
28	yuppy	9	35.0
31	Lubber	8	55.5
44	guppy	5	35.0
58	Rusty	10	35.0

Instance s2 of sailors

Select Operation

- Notation: $\sigma_p(r)$
- p is called the **selection predicate**
- Defined as:

$$\sigma_p(\boldsymbol{r}) = \{t \mid t \in r \textbf{ and } p(t)\}$$

Where p is a formula in propositional calculus consisting of **terms** connected by : $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
Each **term** is one of:

<attribute> *op* <attribute> or <constant>

where *op* is one of: $=, \neq, >, \geq, <, \leq$

- Example of selection: $\sigma_{rating>8}(Sailors)$

- The **selection operator σ** specifies the tuples to retain through a selection condition. In general, the selection condition is a Boolean combination (i.e., an expression using the logical connectives ∧ and ∨) of terms that have the form ***attribute op constant*** or ***attribute1 op attribute2***, where ***op*** is one of the comparison operators <,<=,=,!=,>= or >.

- $\sigma p(r) = \{ t \mid t \in r \text{ and } p(t) \}$

- p is called the selection predicate
- Select all tuples from r that satisfies the predicate p
- Does not change the schema

We can retrieve rows corresponding to expert sailors by using the σ operator.

The expression $\sigma_{rating>8}(S2)$ evaluates to the relation shown below.

The subscript rating>8 specifies the selection criterion to be applied while retrieving tuples.

sid	*sname*	*rating*	*age*
28	yuppy	9	35.0
58	Rusty	10	35.0

Project Operation

- Notation:

$$\Pi_{A_1, A_2, \ldots, A_k}(r)$$

  where A_1, A_2 are attribute names and r is a relation name.

- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- Example: to eliminate sid, rating from sailors we use the below one

$$\pi_{\text{sname,rating}}(\text{S2})$$

- The **projection operator π** allows us to extract columns from a relation;
- $\pi_{A1,\ldots,Ak}(r)$
  - Ai , etc. are attributes of r
  - Select only the specified attributes $A_1,\ldots,A_k$ from all tuples of r
  - Duplicate rows are removed, since relations are sets
  - Changes the schema
- for example, we can find out all sailor names and ratings by using π. The expression $\pi_{sname,rating}(S2)$ evaluates to the relation shown below. The subscript sname, rating specifies the fields to be retained.

sname	*rating*
yuppy	9
Lubber	8
guppy	5
Rusty	10

- We can compute the names and ratings of highly rated sailors by combining two of the preceding queries. The expression $\pi_{sname,rating}(\sigma_{rating>8}(S2))$ produces the result shown below

sname	*rating*
yuppy	9
Rusty	10

- **Note: Projection operation removes duplicates by default (which is not the case in SELECT statement)**

- **SET OPERATIONS**
- The following standard operations on sets are available in relational algebra:
- union (∪),
- intersection (∩),
- set-difference (-),
- and cross-product(X).

Union Operation

- **Union:** R ∪ S returns a relation instance containing all tuples that occur in either relation instance R or relation instance S (or both)
- R and S should be union compatible and the result is defined to be identical to the schema of R.
- Two relation instances are said to be union-compatible if the following conditions hold:
  - They have the same number of fields, and
  - Corresponding fields, taken in order from left to right, have the same domains.
- The field names are not used in defining union-compatibility. For convenience, we will assume that the fields of R ∪ S inherit names from R.
- $r \cup s = \{ t \mid t \in r \text{ or } t \in S \}$
- Does not change the schema

Intersection

- **Intersection**: R ∩ S returns a relation instance containing all tuples that occur in both R and S. R and S must be union compatible. Intersection can be replaced by a pair of set-differences.
- R ∩ S = R - (R - S)
- $r \cap s = \{ t \mid t \in r \text{ and } t \in S \}$
- Does not change the schema

Set-Difference

- **Set-Difference**: R – S returns a relation instance containing all tuples that occur in R, but not in S, R and S should be union compatible.
- $r - s = \{ t \mid t \in r \text{ and } t \notin S \}$
- Does not change the schema

Cross product

- **Cross product**: R X S returns a relation instance whose schema contains all fields of R followed by all the fields of S. The result of R X S contains one tuple(r, s) for each pair of tuples r ∈ R, s ∈ S. This operation is sometimes called **Cartesian product.**
- $r \times s = \{ t\, q \mid t \in r \text{ and } q \in S \}$
- Changes the Schema

Considers the below instances:

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

Instance S1 of Sailors

sid	*sname*	*rating*	*age*
28	yuppy	9	35.0
31	Lubber	8	55.5
44	guppy	5	35.0
58	Rusty	10	35.0

Instance S2 of Sailors

sid	*bid*	*day*
22	101	10/10/96
58	103	11/12/96

Instance R1 of Reserves

The application of set operations on them would result in the following:

S1 ∪ S2

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0
28	yuppy	9	35.0
44	guppy	5	35.0

S1 ∩ S2

sid	*sname*	*rating*	*age*
31	Lubber	8	55.5
58	Rusty	10	35.0

S1 – S2

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0

Considers the below instances:

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

Instance S1 of Sailors

sid	*sname*	*rating*	*age*
28	yuppy	9	35.0
31	Lubber	8	55.5
44	guppy	5	35.0
58	Rusty	10	35.0

Instance S2 of Sailors

sid	*bid*	*day*
22	101	10/10/96
58	103	11/12/96

Instance R1 of Reserves

S1 X R1

(*sid*)	*sname*	*rating*	*age*	(*sid*)	*bid*	*day*
22	Dustin	7	45.0	22	101	10/10/96
22	Dustin	7	45.0	58	103	11/12/96
31	Lubber	8	55.5	22	101	10/10/96
31	Lubber	8	55.5	58	103	11/12/96
58	Rusty	10	35.0	22	101	10/10/96
58	Rusty	10	35.0	58	103	11/12/96

Rename Operation

- 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

$$\rho_{x(A_1, A_2, \ldots, A_n)}(E)$$

returns the result of expression *E* under the name *X*, and with the attributes renamed to $A_1, A_2, \ldots, A_n$.

Joins

- The join operation is one of the most useful operations in the relational algebra and the most commonly used way to combine information from two or more relations.
- A join can be defined as a cross-product followed by selections and projections; joins arise much more frequently in practice than plain cross-products.
- The result of a cross-product is typically much larger than the result of a join. There are several variants of the join operation:
- 1. Conditional Join
- 2. Natural Join
- 3. Equi Join

Inner Join

- 4. **Outer Join**
- a) Left outer join

  b) Right outer join

  c) Full outer join

Conditional join/ Theta join

- **Condition Joins**: Accepts a join condition C and the pair of relation instances as arguments and returns a relation instance. The join condition is identical to a selection condition in form. The operation is defined as follows:

$$R \bowtie_c S = \sigma_c(R \times S)$$

- Thus is defined to be a cross-product followed by a selection. Note that the condition c refers to attributes of both R and S.
- Example: S1 S1.sid<R1.sid R1, because sid appears both in S1 and R1, the corresponding fields in the result are unnamed; domains are inherited from the corresponding fields of S1 and R1.

(sid)	*sname*	*rating*	*age*	*(sid)*	*bid*	*day*
22	Dustin	7	45.0	58	103	11/12/96
31	Lubber	8	55.5	58	103	11/12/96

S1 $\bowtie_{S1.sid<R1.sid}$ R1

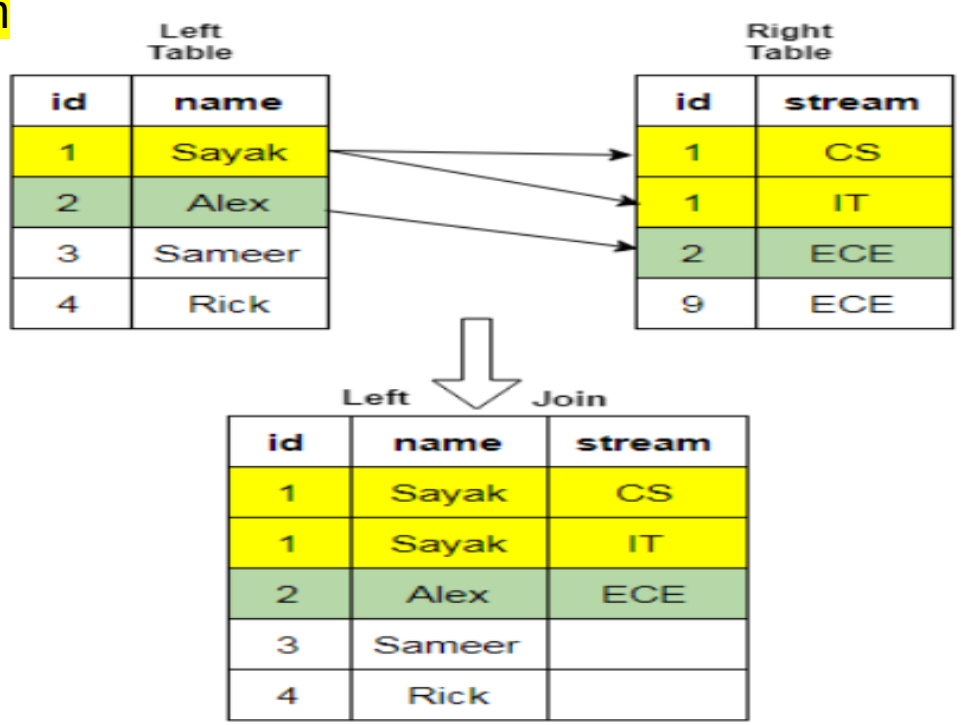
Natural join

- A further special case of the join operation R S is an equijoin in which equalities are specified on all fields having the same name in R and S
- In this case we can simply omit the join condition;
- the default is that the join condition is a collection of equalities on all common fields.
- We call this case a natural join, and it has the property that the result is guaranteed not to have two fields with the same name.
- Natural join is a type of equi-join which occurs implicitly by comparing all the same named columns in both tables.

Left Outer Join

- It takes all the tuples in the left relation that did not match with any tuple in the right relation, pads the tuples with null values for all other attributes from the right relation, and adds them to the result of the natural join.

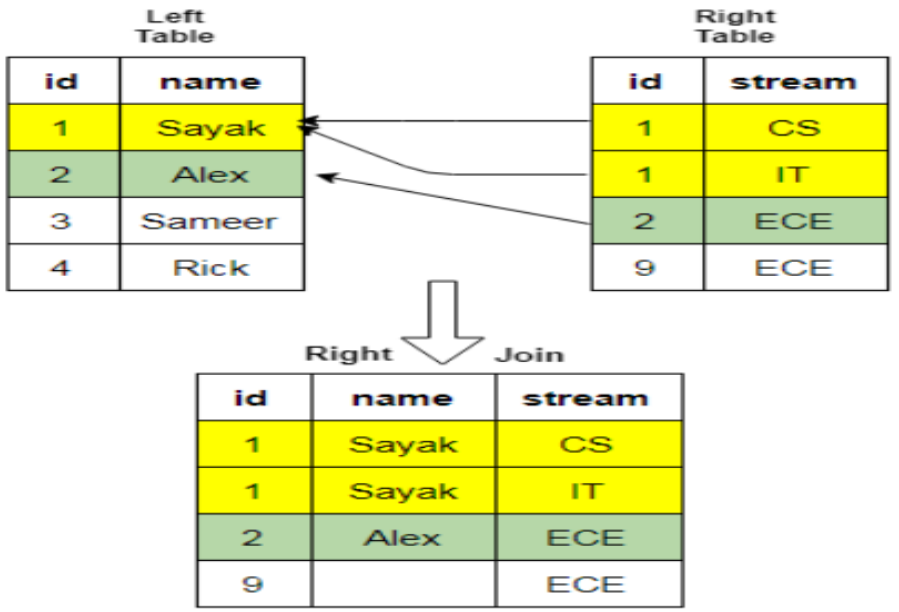
```
SELECT s1.id, s1.name, s2.stream
 FROM student_name AS s1
LEFT JOIN student_stream AS s2
ON s1.id = s2.id;
```



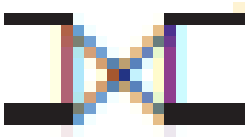
Right outer join

- It takes all the tuples in the right relation that did not match with any tuple in the left relation, pads the tuples with null values for all other attributes from the left relation, and adds them to the result of the natural join.

```
SELECT s1.id, s1.name, s2.stream
FROM student_name AS s1
RIGHT JOIN student_stream AS s2
ON s1.id = s2.id;
```



Full Outer join



It does both the operations, padding tuples from the left relation that did not match any from the right relation, as well as tuples from the right relation that did not match any from the left relation, and adding them to the result of the join.

```
SELECT s1.id, s1.name,
s2.stream
FROM student_name AS s1 FULL
JOIN student_stream AS s2 ON
s1.id = s2.id;
```

Left Table

id	name
1	Sayak
2	Alex
3	Sameer
4	Rick

Right Table

id	stream
1	CS
1	IT
2	ECE
9	ECE

Full Join

id	name	stream
1	Sayak	CS
1	Sayak	IT
2	Alex	ECE
3	Sameer	
4	Rick	
9		ECE

Division operation

- Consider two relation instances A and B in which A has (exactly) two fields x and y and B has just one field y, with the same domain as in A. A/B can be defined as the set of all x values (in the form of unary tuples) such that for every y value in (a tuple of) B, there exists a tuple (x, y) in A. A/B can also be defined as:

$$\pi_x(A) - \pi_x((\pi_x(A) \times B) - A)$$

$$r \div s = \{t \mid t \in \Pi_{R-S}(r) \text{ and } \forall u \in s(tu \in r)\}$$

- Changes the schema to R-S

Example:

A

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B1

pno
p2

B2

pno
p2
p4

B3

pno
p1
p2
p4

A/B1

sno
s1
s2
s3
s4

A/B2

sno
s1
s4

A/B3

sno
s1

Assignment operation

- It is convenient at times to write a relational-algebra expression by assigning parts of it to temporary relation variables. The assignment operation, denoted by , works like the assignment in programming.

Example:

$$temp1 \leftarrow R \times S$$
$$temp2 \leftarrow \sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \ldots \wedge r.A_n = s.A_n}(temp1)$$
$$result = \Pi_{R \cup S}(temp2)$$

SOME EXAMPLES OF ALGEBRA QUERIES

Consider the following relation instances:

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

SAILORS

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

RESERVES

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

BOATS

Formal Definition

- A basic expression in the relational algebra consists of either one of the following:
  - A relation in the database
  - A constant relation
- Let E_1 and E_2 be relational-algebra expressions; the following are all relational-algebra expressions:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p (E_1)$, P is a predicate on attributes in E_1
  - $\Pi_S(E_1)$, S is a list consisting of some of the attributes in E_1
  - $\rho_x (E_1)$, x is the new name for the result of E_1

EXTENDED RELATIONAL-ALGEBRA OPERATIONS

GENERALIZED PROJECTION

The generalized-projection operation extends the projection operation by allowing arithmetic functions to be used in the projection list. The generalized projection has the form:

$$\Pi_{F_1, F_2, \ldots, F_n}(E)$$

Where E is any relational-algebra expression , and each of $F_1, F_{2,\ldots,} F_n$ is an arithmetic expression involving constants and attributes in the schema of E

Example:

$$\Pi_{ID, name, dept\_name, salary \div 12}(instructor)$$

AGGREGATE FUNCTIONS

- Aggregate functions take a collection of values and return a single value as a result. Eg: Avg, Min, Max etc
- The aggregate operation permits the use of aggregate functions on a set of values.
- The general form of the aggregation operation is as follows:

$$G_1, G_2, \ldots, G_n \mathcal{G} F_1(A_1), F_2(A_2), \ldots, F_m(A_m)(E)$$

- Where E is any relational algebra expression, $G_1, G_{2,\ldots,} G_n$ constitute a list of attributes on which to group ; each F_i is an aggregate function; and each A_i is an attribute name.
- The tuples in the result of the expression E are partitioned into groups in such a way that
  1. All tuples in a group have the same values for $G_1, G_2, \ldots, G_n$.
  2. Tuples in different groups have different values for $G_1, G_2, \ldots, G_n$.

Examples:

1. $\mathcal{G}_{\text{sum}(salary)}(instructor)$
2. $\mathcal{G}_{\text{count-distinct}(ID)}(\sigma_{semester=\text{“Spring”} \wedge year\ =2010}(teaches))$
3. $dept\_name\ \mathcal{G}_{\text{average}(salary)}(instructor)$

Null Values

Let us see how various relational-algebra operations deal with NULL values

- **NULL** indicates value **unknown** or **non-existent.**
- Arithmetic expressions involving null evaluate to null
- Aggregate functions ignore null
- All relational operations ignore NULL

Logical operations :

i) and : true **and** unknown = unknown
false **and** unknown = false
unknown **and** unknown = unknown
ii) or : true **or** unknown = true
false **or** unknown = unknown
unknown **or** unknown = unknown
iii) not : **not** unknown = unknown

Exercise : Write how the following operations work on NULL
Select, Join, Projection, Set Operations, Aggregate, Generalized Projection, Outer Join

Modification of the database

- Assignment operator is used to express them

DELETION: R ← R - E (R is a relation, E is a relational-algebra expression)
Only whole tuples can be deleted, not some attributes

Applying $r \leftarrow r - \sigma_{A=1}(r)$ on

A	B	C
1	1	5
1	2	5
2	3	5
2	4	8

returns

A	B	C
2	3	5
2	4	8

INSERTION : R ← R ∪ E (R is a relation, E is a relational-algebra expression)
Only whole tuples can be inserted, not some attributes

Applying $r \leftarrow r \cup \{(1, 2, 5)\}$ on

A	B	C
1	1	5
2	3	5
2	4	8

returns

A	B	C
1	1	5
2	3	5
2	4	8
1	2	5

- UPDATING : We can use the generalized projection to do this task

$$r \leftarrow \Pi_{F_1, F_2, \ldots, F_n}(r)$$

- Where each Fi is eitehr ith attribute of r, if the ith attribute is not updated, or, if the attribute is to be updated, Fi is an expression, involving only constants and the attributes of r, that gives the new value for the attribute.

- Update allows values of only some attributes to change

- Note that the schema of the expression resulting from the generalized projection expression must match the original schema of r

Applying $r \leftarrow \Pi_{A,2*B,C}(r)$ on

A	B	C
1	2	5
1	1	5
2	4	8

returns

A	B	C
1	4	5
1	2	5
2	8	8

Applying $r \leftarrow \Pi_{A,2*B,C}(\sigma_{A=1}(r))$ on

A	B	C
1	2	5
1	1	5
2	4	8

returns

A	B	C
1	4	5
1	2	5

Operations that result in the schema change :

Select	×
Project	✓
Union	×
Difference	×
Cartesian product	✓
Rename	✓ (not the meaning)

Drawbacks of Relational Algebra

- First-order propositional logic
- Do not support recursive closure operations
- Needs specifying multiple queries, each solving only one level at a time

1) Find the names of sailors who have reserved boat # 103

$\pi_{sname}((\sigma_{bid=103}Reserves) \bowtie Sailors)$

2) Find the names of sailors who reserved a red boat

$\pi_{sname}((\sigma_{color='red'}Boats) \bowtie Reserves \bowtie Sailors)$

3) Find Colors of boats reserved by Lubber

$\pi_{color}((\sigma_{snamer='Lubber'}Sailors) \bowtie Reserves \bowtie Boats)$

4) Find the names of sailors who have reserved at least one boat

$\pi_{sname}(Sailors \bowtie Reserves)$

5) Find the names of sailors who have reserved a red or a green boat

$\rho\ (Tempboats, (\sigma_{color='red'}Boats) \cup (\sigma_{color='green'}Boats))$

$\pi_{sname}(Tempboats \bowtie Reserves \bowtie Sailors)$

6) Find the names of sailors who have reserved a red or a green boat

In this case, writing the same query as above replacing the union operation with the intersection operation may seem correct, but it is not, because in that case it would search for a boat which is both in red as well as green color which always returns a null.

Hence, the correct query would be:

ρ (Tempred, $\pi_{sid}((\sigma_{color='red'}Boats) \bowtie Reserves)$)

ρ (Tempgreen, $\pi_{sid}((\sigma_{color='green'}Boats) \bowtie Reserves)$)

$\pi_{sname}((Tempred \cap Tempgreen) \bowtie Sailors)$

7) Find the names of sailors who reserved at least two boats

ρ (Reservation, $\pi_{sid,sname,bid}(Sailors \bowtie Reserves)$)

ρ (Reservationpairs(1→sid1, 2→sname1, 3→bid1, 4→sid2, 5→sname2, 6→bid2), Reservations X Reservations)

$\pi_{sname1}\ \sigma_{(sid1=sid2)\wedge(bid1!=bid2)}Reservationpairs$

8) Find the sids of the sailors with age over 20 who have not reserved a red boat.

$\pi_{sid}(\sigma_{age>20}Sailors) - \pi_{sid}((\sigma_{color='red'}Boats) \bowtie Reserves \bowtie Sailors)$

9) Find the names of Sailors who have reserved all boats.

ρ (Tempsids,($\pi_{sid,bid}$Reserves) / (π_{bid}Boats))

π_{sname}(Tempsids $\bowtie$ Sailors)

10) Find the names of sailors who have reserved all boats called 'Interlake'

ρ (Tempsids,($\pi_{sid,bid}$Reserves) / (π_{bid}($\sigma_{bname='Interlake'}$Boats))

π_{sname}(Tempsids $\bowtie$ Sailors)