

Chapter - 6

Query Languages

Relational Query Languages

- A **query language** is a language in which user requests information from the database.
- **SQL:-** Languages for describing queries on a relational database
- ***Relational Algebra***
 - We begin with the notations used in RA and that are unique to RA and are procedural, requiring a sequence of operations to express a query transaction

What is an Algebra?

- A language based on operators and a domain of values.
- Operators map values taken from the domain into other domain values.
- Hence, an expression involving operators and arguments produces a value in the domain.
- When the domain is a set of all relations (and the operators are as described later), we get the *relational algebra*,
- We refer to the expression as a *query* and the value produced as the *query result*.

Relational Algebra

- **Domain:** set of relations

Fundamental Operations

- SELECT
 - PROJECT
 - UNION
 - SET DIFFERENCE
 - CARTESIAN PRODUCT
- **Select** and **project** operations are unary operation as they operate on a single relation. **Union, set difference, Cartesian product** and operations are binary operations as they operate on pairs of relations.

Relational Algebra

- Basic operations:

- Selection σ) Selects a subset of rows from relation.
- Projection π) Deletes unwanted columns from relation.
- Cross-product $\times$) Allows us to combine two relations.
- Set-difference $-$) Tuples in reln 1, but not in reln 2.
- Union $\cup$) Tuples in reln 1 and in reln 2.

- Additional operations:

- Intersection, join, division, renaming: Not essential, but (very!) useful.

- Since each operation returns a relation, operations can be composed!

Selection Condition

- Selects rows that satisfy *selection condition*.
- *Schema* of result identical to schema of (only) input relation.
- *Result* relation can be the *input* for another relational algebra operation.
- Operators: $<$, $\leq$, $\geq$, $>$, $=$, $\neq$
- Simple selection condition:
 - $<condition>$ AND $<condition>$
 - $<condition>$ OR $<condition>$
 - NOT $<condition>$

Selection Condition - Examples

nr	name	salary
1	John	100
5	Sarah	300
7	Tom	100

- $\sigma_{salary > 3000 \text{ Or } name = 'Tom'}(\text{Employee})$
- $\sigma_{nr > 5 \text{ AND } nr < 8}(\text{Employee})$
- $\sigma_{\text{NOT}(name = 'John')}(\text{Employee})$
- $\sigma_{nr \neq 7}(\text{Employee})$

Table1:

Employee

SQL	Result	Relational algebra									
<pre>select * from E where salary < 200</pre>	<table> <tr><th>nr</th><th>name</th><th>salary</th></tr> <tr><td>1</td><td>John</td><td>100</td></tr> <tr><td>7</td><td>Tom</td><td>100</td></tr> </table>	nr	name	salary	1	John	100	7	Tom	100	$\text{SELECT}_{salary < 200}(E)$
nr	name	salary									
1	John	100									
7	Tom	100									
<pre>select * from E where salary < 200 and nr >= 7</pre>	<table> <tr><th>nr</th><th>name</th><th>salary</th></tr> <tr><td>7</td><td>Tom</td><td>100</td></tr> </table>	nr	name	salary	7	Tom	100	$\text{SELECT}_{salary < 200 \text{ and } nr \geq 7}(E)$			
nr	name	salary									
7	Tom	100									

Note that the select operation in relational algebra has nothing to do with the SQL keyword select. **Selection in relational algebra returns those tuples in a relation that fulfil a condition, while the SQL keyword select means "here comes an SQL statement".**

BRANCH_NAME	LOAN_NO	AMOUNT
Downtown	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

σ BRANCH_NAME="perryride" (LOAN)

BRANCH_NAME	LOAN_NO	AMOUNT
Perryride	L-15	1500
Perryride	L-16	1300

Projection

- **Deletes attributes that are not in *projection list*.**
- *Schema* of result contains exactly the fields in the projection list, with the same names that they had in the (only) input relation.
- **Projection operator has to eliminate *duplicates*!** (Why??, what are the consequences?)
 - Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it. (Why not?)
 - where ' π ' (pi) is the symbol used to represent the PROJECT operation, and $\langle \text{attributelist} \rangle$ is the desired list of attributes from the attributes of relation R
- **Produces table containing subset of columns of argument table**

$$\pi_{\text{attribute list}}(\text{relation})$$

Example:

Person					$\pi_{\text{Name,Hobby}}(\text{Person})$	
<i>d</i>	Name	Address	Hobby		Name	Hobby
	1123	John 123 Main	stamps		John	stamps
	1123	John 123 Main	coins		John	coins
	5556	Mary 7 Lake Dr	hiking		Mary	hiking
	9876	Bart 5 Pine St	stamps		Bart	stamps

Project Operator

- Example:

Person

$\Pi_{Name, Address}(Person)$

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

<i>Name</i>	<i>Address</i>
John	123 Main
Mary	7 Lake Dr
Bart	5 Pine St

Result is a table (no duplicates)

NAME	STREET	CITY
Jones	Main	Harrison
Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

□ NAME, CITY (CUSTOMER)

NAME	CITY
Jones	Harrison
Smith	Rye
Hays	Harrison
Curry	Rye
Johnson	Brooklyn
Brooks	Brooklyn

<u>sid</u>	<u>bid</u>	<u>day</u>
22	101	10/10/96
58	103	11/12/96

R1

<u>sid</u>	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

S2

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Union, Intersection, Set-Difference

- union: The result of this operation, denoted by $\mathbf{R \cup S}$, is a relation that includes all tuples that are either in R or in S or in both R and S. Duplicate tuples are eliminated.
- intersection: The result of this operation, denoted by $\mathbf{R \cap S}$, is a relation that includes all tuples that are in both R and S.
- set difference(or MINUS): The result of this operation, denoted by $\mathbf{R - S}$, is a relation that includes all tuples that are in R but not in S.

Union, Intersection, Set-Difference

- All of these operations take two input relations, which must be *union-compatible*:
 - Same number of fields.
 - `Corresponding' fields have the same type.

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

$S1 \cup S2$

- What is the *schema* of result?

sid	sname	rating	age
22	dustin	7	45.0

$S1 - S2$

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

$S1 \cap S2$

Cross-Product

- **Each row of S1 is paired with each row of R1.**
- *Result schema* has one field per field of S1 and R1, with field names 'inherited' if possible.

- C =

(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

Cont..

- **Notice that both UNION and INTERSECTION are commutative operations**; that is,

$$R \cup S = S \cup R, \text{ and } R \cap S = S \cap R$$

- **Both UNION and INTERSECTION can be treated as **n-ary** operations applicable to any number of relations because both are associative operations**; that is,

$$R \cup (S \cap T) = (R \cup S) \cap T, \text{ and } (R \cap S) \cap T = R \cap (S \cap T)$$

- The **MINUS operation** is not commutative; that is, in general,

$$R - S \neq S - R$$

- Rename Operation

- The rename operation is used to rename the output relation. It is denoted by **rho** (ρ).
- **Example:** We can use the rename operator to rename EMPLOYEE relation to STAFF
- $\rho(\text{STAFF}, \text{EMPLOYEE})$

- Join Operations:

- A Join operation combines related tuples from different relations, if and only if a given join condition is satisfied. It is denoted by $\bowtie$.

- Natural Join:

- A natural join is the set of tuples of all combinations in R and S that are equal on their common attribute names.
- It is denoted by $\bowtie$.

- $\pi_{\text{EMP_NAME, SALARY}} (\text{EMPLOYEE} \bowtie \text{SALARY})$

EMP_CODE	EMP_NAME
101	Stephan
102	Jack
103	Harry

EMP_CODE	SALARY
101	50000
102	30000
103	25000

(EMPLOYEE ⋈ SALARY)

EMP_CODE	EMP_NAME	SALARY
101	Stephan	50000
102	Jack	30000
103	Harry	25000

- The outer join operation is an extension of the join operation.
- It is used to deal with missing information.

EMP_NAME	STREET	CITY
Ram	Civil line	Mumbai
Shyam	Park street	Kolkata
Ravi	M.G. Street	Delhi
Hari	Nehru nagar	Hyderabad

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Shyam	Wipro	20000
Kuber	HCL	30000
Hari	TCS	50000

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru nagar	Hyderabad	TCS	50000

(EMPLOYEE ⋈ FACT_WORKERS)

An outer join is basically of three types:

Left outer join

Right outer join

Full outer join

A. Left outer join:

Left outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.

In the left outer join, tuples in R have no matching tuples in S.

It is denoted by $\bowtie_{\text{left}}$.

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL

EMPLOYEE $\bowtie$ FACT_WORKERS

B. Right outer join:

Right outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names. In right outer join, tuples in S have no matching tuples in R. It is denoted by $\bowtie\rightarrow$.

EMP_NAME	BRANCH	SALARY	STREET	CITY
Ram	Infosys	10000	Civil line	Mumbai
Shyam	Wipro	20000	Park street	Kolkata
Hari	TCS	50000	Nehru street	Hyderabad
Kuber	HCL	30000	NULL	NULL

EMPLOYEE $\bowtie$ FACT_WORKERS

C. Full outer join:

- ❑ Full outer join is like a left or right join except that it contains all rows from both tables.
- ❑ In full outer join, tuples in R that have no matching tuples in S and tuples in S that have no matching tuples in R in their common attribute name.
- ❑ It is denoted by $\bowtie$.

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL
Kuber	NULL	NULL	HCL	30000

EMPLOYEE □ FACT_WORKERS

3. Equi join:

- It is also known as an inner join. It is the most common join.
- It is based on matched data as per the equality condition.
- The equi join uses the comparison operator(=).

CLASS_ID	NAME
1	John
2	Harry
3	Jackson

PRODUCT_ID	CITY
1	Delhi
2	Mumbai
3	Noida

CUSTOMER ⋈ PRODUCT

CLASS_ID	NAME	PRODUCT_ID	CITY
1	John	1	Delhi
2	Harry	2	Mumbai
3	Harry	3	Noida

Relational calculus

- **Expression to specify a retrieval request, and hence there is no description of how to evaluate a query.**
- A calculus expression specifies what is to be retrieved rather than how to retrieve it.
- Therefore, the **relational calculus** is considered to be a non procedural language. This differs from **relational algebra**, where we must write sequence of operations to specify a retrieval request; hence, it can be considered as a procedural way of stating a query.
- It is possible to nest algebra operations to form a single expression;
- However, a certain order among the operations is always explicitly specified in a relational algebra expression.

Cont..

- A simple tuple relational calculus query is of the form $\{t \mid \text{COND}(t)\}$

Where **t** is a **tuple variable** and **COND(t)** is a **conditional expression involving t**. The result of such a query is the set of all tuples t that satisfy COND(t).

- The AND, OR, and NOT logical operators are also used in the expressions.
- An example of a RC expression is:

$$\{t \mid \text{STUDENT}(t) \text{ AND } t.\text{GPA} > 3.0\}$$

- This query would return all students whose GPA is greater than 3.0.
- Expressions and Formulas in RC as noted by Elmasri et al are in the form:

$$\{t_1.A_j, t_2.A_k, \dots, t_n.A_m \mid \text{COND}(t_1, t_2, \dots, t_n, t_{n+1}, t_{n+2}, \dots,$$

Cont..

- For example2, to find all employees whose salary is above \$50,000, we can write the following tuple calculus expression: $\{r \mid \text{EMPLOYEE}(t) \text{ and } t.\text{SALARY} > 50000\}$
- The condition $\text{EMPLOYEE}(t)$ specifies that the range relation of tuple variable t is EMPLOYEE . Each EMPLOYEE tuple t that satisfies the condition $t.\text{SALARY} > 50000$ will be retrieved.
- Notice that $t.\text{SALARY}$ references attribute SALARY of tuple variable t ; this notation resembles how attribute names are qualified with relation names or aliases in SQL.

Universal and Extensile quantifier

- **Universal quantifier ($\forall$):** Is true if **all tuples** **make** the formula **true**.
- **Existential quantifier ($\exists$):** Is true if there **exists** **some tuple** that makes the formula **true**.
- Read example of them more

Sql language

- **Sql** stands for structured query language.
- standard language for accessing and manipulating database.
- Sql language(DDL(create , drop, alter),DML(select, update, Insert , delete))
- SQL is not case sensitive
- Primary key cant be null

Cont..

- The SQL language may be considered one of the major reasons for the success of relational databases in the commercial world.
- Because it became a standard for relational databases, users were less concerned about migrating their database applications from Other types of database systems for example , network or hierarchical systems-to relational systems

Cont..

- SQL is a comprehensive database language.
- It has statements for DDL query, and update.
Hence, it is both a DML.
- In addition, it has facilities for Defining views on the database, for specifying security and authorization, for defining integrity constraints, and for specifying transaction controls.

Query Sublanguage of SQL

- Some Sql keywords are:
 - Select,
 - where,
 - from,
 - create,
 - delete,
 - drop,
 - update,
 - alter
 - etc.....

Duplicates

- Duplicate rows not allowed in a relation
- However, duplicate elimination from query result is costly and not automatically done; it must be explicitly requested:

```
SELECT DISTINCT .....  
FROM .....
```

Relational Algebra (RA) and Relational Calculus (RC)

- Relational Algebra (RA) and Relational Calculus (RC) are **formal languages** for the database relational model.
- **SQL** is the **practical language** in the database relational model.
- In these **formal languages** a conceptual database model is **expressed in mathematical terms and notations** while in the **practical language** - SQL, the mathematical expressions of the functionality and transaction of the database operations are **implemented physically**.
- Each language, that RA, RC, and SQL have their own notations to express their notations.
- **RA Notations - Procedural Expression Language**
- **RC Notation - Non Procedural Expression Language**
- SQL is what is called a declarative language or non procedural, meaning that you **only need to specify what needs to be accomplished** (such as a query or insert) and **not how to do it**; the DBMS determines and performs internally the step by step operation needed to obtain the result.

