Relational Algebra

Relational Algebra

Relational Algebra is a collection of operations that are used to manipulate entire relations.

The result of each operation is a new relation, which can be further manipulated.

Relational Algebra Operators are divided into two groups:

- Set Operations like UNION, INTERSECTION, DIFFERENCE, and CARTESIAN PRODUCT.
- Relational Database Specific Operations like SELECT, PROJECT, and JOIN.

SELECT OPERATION (σ)

Selects the tuples (rows) from a relation R that satisfy a certain selection condition c

- **Form of the operation $\sigma_c(R)$**
- **The condition c is an arbitrary Boolean expression on the attributes of R**
- **Resulting relation has the same set of attributes as R**
- **Resulting relation includes each tuple in r(R) whose attribute values satisfy the condition c**

Examples

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

$\sigma_{SALARY>3000}(\text{Employee})$

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

PROJECT Operation (Π)

Keeps only certain attributes (columns) from a relation R specified in an attribute list L

Form of operation: $\Pi_L(R)$

Resulting relation has only those attributes of R specified in L

Example:

$\Pi_{FNAME,\ LNAME,\ SALARY}$ (EMPLOYEE)

The project operation eliminates duplicate tuples in the resulting relation so that the result remains a mathematical set.

$\Pi_{SEX,SALARY}$(EMPLOYEE)

If several male employees have salary 3000, only a single tuple <M,3000> is kept in the resulting relation.

Sequences of operations

Several operations can be combined to form a relational algebra expression (query)

Example: Retrieve the names and salaries of employees who work in department 4:

a) $\Pi_{FNAME,\ LNAME,\ SALARY}(\sigma_{DNO=4}(Employee))$

Alternatively, intermediate relations for each step are specified

b) $Dept4\_Emps \leftarrow \sigma_{DNO=4}(Employee)$

$Result \leftarrow \Pi_{FNAME,\ LNAME,\ SALARY}(Dept4\_Emps)$

Attributes can be renamed in the resulting relation:

$Dept4\_Emps \leftarrow \sigma_{DNO=4}(Employee)$

$R(FN,LN,Sal) \leftarrow \Pi_{FNAME,LNAME,SALARY}(Dept4\_Emps)$

Set Operations

Binary operations from mathematical set theory

Union: $R \cup S$

Intersection: $R \cap S$

Set Difference: $R - S$

Cartesian Product: $R \times S$

The operand relations $R(A_1, A_2, ..., A_n)$ and $S(B_1, B_2, ..., B_n)$ must have the same number of attributes for operations $\cup$, $\cap$, and -. And the domains of corresponding attributes must be compatible; that is $dom(A_i) = dom(B_i)$ for $i = 1,2, ..., n$. This condition is called union compatibility.

The resulting relation for $\cup$, $\cap$, and - has the same attribute names as the first operand relation R (by convention).

Examples

STUDENT

FN	LN
Amy	**Wong**
Jim	**Lai**
Cindy	**Chan**
Micheal	**Chang**

INSTRUCTOR

FN	LN
Amy	Wong
Jim	Lai
Abhay	Joshi
John	Right

STUDENT $\cup$ INSTRUCTOR

FN	LN
Amy	Wong
Jim	Lai
Cindy	Chan
Micheal	Chang
Abhay	Joshi
John	Right

INSTRUCTOR $\cap$ STUDENT

FN	LN
Amy	Wong
Jim	Lai

INSTRUCTOR - STUDENT

FN	LN
Abhay	Joshi
John	Right

STUDENT - INSTRUCTOR

FN	LN
Cindy	Chan
Micheal	Chang

Cartesian Product

$R(A_1, A_2, ..., A_m, B_1, B_2, ..., B_n) \leftarrow R_1(A_1, A_2, ..., A_m) \text{ X } R_2(B_1, B_2, ..., B_n)$

A tuple t exists in R for each combination of tuples t_1 from R_1 and t_2 from R_2 such that: $t[A_1, A_2, ..., A_m] = t_1$ and $t[B_1, B_2, ..., B_n] = t_2$

If R_1 has n_1 tuples and R_2 has n_2 tuples, then R will have $n_1 * n_2$ tuples.

Cartesian Product is meaningless operation on its own. It can combine related tuples from two relations if followed by an appropriate Select operation.

Example: Combine each Employee tuple with the Department tuple of the manager.

EMP-DEPT <- Department X Employee

MGR-EMP <- $\sigma_{ENO = MgrENO}$(EMP-DEPT)

Example

EMPLOYEE

ENO	Name	EDNO
1	Cindy	10
2	Alice	20
3	Micheal	20

DEPARTMENT

DNO	DName	MgrENO
10	Sales	1
20	Accounts	3

EMP-DEPT = EMPLOYEE X DEPARTMENT

ENO	Name	EDNO	DNO	DName	MgrENO
1	Cindy	10	10	Sales	1
1	Cindy	10	20	Accounts	3
2	Alice	20	10	Sales	1
2	Alice	20	20	Accounts	3
3	Micheal	20	10	Sales	1
3	Micheal	20	20	Accounts	3

MGR-EMP = $\sigma_{(ENO = MgrENO)}$(EMPLOYEE X DEPARTMENT)

ENO	Name	EDNO	DNO	DName	MgrENO
1	Cindy	10	10	Sales	1
3	Micheal	20	20	Accounts	3

Join Operations

Theta Join: Similar to a cartesian product followed by a select. The condition **c** is called a join condition. $R_1(A_1, A_2, .., A_m)$ and $R_2(B_1, B_2,..., B_n)$, $R_1 \bowtie_{A\theta B} R_2$, where θ in {=,<,>,<=,>=,<>}.

Example: Given relations Emp(Name, Sal); MGR(Mname, Bonus)

$Emp \bowtie_{Sal < Bonus} MGR$; lists Emps with Salary<Bonus of MGRs

Equijoin: The join condition c includes one or more equality comparisons involving join attributes from R_1 and R_2.

That is, c is $(A_i = B_j)$ AND … AND $(A_k = B_l)$

where A's and B's are distinct attributes of relations R_1 and R_2.

Example: Retrieve each Department's name and its Manager's name: $T \leftarrow Department \bowtie_{MGRSSN=SSN} Employee$

$Result \leftarrow \Pi_{DNAME,\ NAME}(T)$

Join Operations (Cont.)

Natural Join (*): In Equijoin R <- R1 $\bowtie_c$ R2, the join attribute of R2 appear redundantly in the result relation R. In a natural join, the redundant join attributes of R2 are eliminated from R. The equality condition is implied and need not be specified.

R <- R $*_{\text{(join attributes of R), (join attributes of S)}}$ S

Note: In the original definition of natural join, the join attributes were required to have same names in both relations.

Example: Retrieve each Employee(ENO, Name, SuperENO) name and the name of his/her Supervisor:

Supervisor(SENO, SName) <- $\Pi_{\text{ENO, Name}}$(Employee)

T<- Employee * Supervisor

Result <- $\Pi_{\text{Name,Sname}}$ (T)

Join Operations (Cont.)

There can be a more than one set of join attributes with a different meaning between the same two relations. For example:

Employee.SSN= Department.MgrSSN meaning Employee manages Department, and Employee.Dno = Department.Dnumber meaning Employee works for the Department

Example: Retrieve each Employee's name and the name of the Department he/she works for:

T<- Employee $*_{\text{EDNO=DNO}}$ Department; Result <- $\Pi_{\text{Name, DName}}$ (T)

A relation can have a set of join attributes to join it with itself.

Employee(1).SuperSSN=Employee(2).SSN meaning Employee(2) supervises Employee(1)

One can think of this as joining two distinct copies of the relation, although only one relation actually exists.

Example Database

STUDENT			
Name	Student Number	Class	Major
Smith	17	1	COSC
Brown	18	2	COSC

GRADE-REPORT		
Student Number	Section-Identifier	Grade
17	85	A
18	102	B+

PREREQUISITE	
Course Number	Prerequisite Number
COSC3380	COSC3320
COSC3320	COSC1310

COURSE			
Course Name	Course Number	Credit Hours	Department
Intro to CS	COSC1310	4	COSC
Data Structures	COSC3320	4	COSC
Discrete Mathematics	MATH2410	3	MATH
Data Base	COSC3380	3	COSC

SECTION				
Section-Identifier	Course Number	Semester	Year	Instructor
85	MATH2410	Fall	91	King
92	COSC1310	Fall	91	Anderson
102	COSC3320	Spring	92	Knuth
135	COSC3380	Fall	92	Stone

QUIZ

For those students who received Grade 'A' in a course, get the student name and course name (see example database in slide 24)

Answer:

Student name is in relation STUDENT, Course name is in relation COURSE, while grade is in relation GRADE-REPORT. For a student to retrieve the Course name, one has to join STUDENT with GRADE-REPORT to get Section number, and then with SECTION to get the course number, and then from COURSE the Course name. Since we want only those students who got grade 'A', a selection operation on relation GRADE-REPORT is needed. Since we want only student Name and Course Name (and not all attributes), a project operation is required.

$$\Pi_{\text{Name, Course Name}}(\sigma_{\text{Grade = `A'}}(\text{STUDENT*GRADE-REPORT *SECTION* COURSE}))$$

The result is:

Name	Course Name
SMITH	DISCRETE MATHEMATICS

Divide operation

Divide operation is useful for special queries, such as, retrieve the names of employees who work on all the projects that Lee works on.

ID_PNOS

ID	PNOS
1	10
1	20
2	10
2	20
3	5
3	10
4	20
5	10

Lee_PNOS

PNOS
10
20

ID_PNOS/Lee_PNOS

ID
1
2

$R(Z) / S(X)$, where $X \subseteq Z$; let $Y = Z - X$.

The result of DIVISION is a relation $T(Y)$ that includes a tuple t in $T(Y)$ if a tuple t_R in $R(Z)$ whose $t_R[Y] = t$ appears in R, with $t_R[X] = t_S$ for every tuple t_S in $S(X)$

Relational Algebra expression for Division:

$T_1 \leftarrow \Pi_Y(R)$; $T_2 \leftarrow \Pi_Y((S \times T_1) - R)$; $T \leftarrow T_1 - T_2$

QUIZ

Find the names of employees who work on *all* projects controlled by department number 5

get all projects controlled by department number 5

$$DEPT5\_PROJS(PNO) \leftarrow \Pi_{PNUMBER}(\sigma_{DNUM=5}(PROJECT))$$

get all employees and projects they work on.

$$EMP\_PROJ(SNN,PNO) \leftarrow \Pi_{ESSN,\ PNUMBER}\ (WORKS\_ON)$$

get all ESSNs that work on all projects controlled by department 5

$$RESULT\_EMP\_SSNS \leftarrow EMP\_PROJ\ /\ DEPT5\_PROJS$$

project the employee name

$$RESULT \leftarrow \Pi_{LNAME,FNAME}(RESULT\_EMP\_SSNS * EMPLOYEE)$$

See text book Pages 170-172 for more examples.

Relational Algebra: Completeness

All the operations discussed so far can be described as a sequence of only the operations SELECT, PROJECT, UNION, SET DIFFERENCE, and CARTESIAN PRODUCT.

Hence, the set {Π, σ, U, -, X} is called a complete set of relational algebra operations. Any query language equivalent to these operations is called relationally complete.

For database applications, additional operations are needed that were not part of the original relational algebra. These include:

Aggregate functions and grouping

OUTER JOIN and OUTER UNION.

Aggregate Relational Operations

Aggregate Functions - functions such as SUM, COUNT, AVERAGE, MIN, MAX are often applied to sets of values or sets of tuples in database applications

<grouping attributes> $\mathfrak{I}$ <function list> (R)

The grouping attributes are optional

Example1: Retrieve the average salary of all employees (no grouping needed) R(AVGSAL) <- $\mathfrak{I}_{\text{Average salary}}$(Employee)

Example2: For each department, retrieve the department number, the number of employees, and the average salary (in the department):

R(DNo, NumEmps, AVGSAL) <- $_{\text{DNo}}\mathfrak{I}_{\text{Count SSN, Average salary}}$(Employee)

DNo is called the grouping attribute in the above example.

OUTER JOIN & OUTER UNION

OUTER JOIN

- In a regular equijoin or natural join operation, tuples in R_1 or R_2 that do not have matching tuples in the other relation do not appear in the result
- Some queries require all tuples in R_1 (or R_2 or both to appear in the result)
- When no matching tuples are found, nulls are placed for the missing attributes

OUTER JOIN & OUTER UNION

Left Outer Join: $R_1 ⟕ R_2$ lets every tuple in R_1 appear in the result

Right Outer Join: $R_1 ⟖ R_2$ lets every tuple in R_2 appear in the result

Full Outer Join: $R_1 ⟗ R_2$ lets every tuple in R_1 and R_2 appear in the result.

Example: List of all employee names and and also names of the departments they manage if they happen to manage

R<- $\Pi_{Fname,Minit,LName,Dname}$(Employee ⟕ Department)

Outer Union operation was developed to take the union of two relations that are not union compatible.