5. Relational Algebra

Relational Algebra

- Relation database 에 대해 질의(query) 처리 연산자들의 모임을 relational algebra 라 함.

- Query는 relation에서 원하는 정보를 검색(retrieve)하는 조건을 명시하는 명령문

- (1 개 혹은 여러 개의) 입력 relation에 대해 질의 연산의 결과인 (1 개의) 출력 relation 이 생성됨.

- 기본적으로 8 개의 검색 연산자들로 구성됨.

- 이 연산자들은 Query language인 SQL의 검색 연산 과정을 실현하는데 매우 중요함.

Relational Algebra

- Basic Operations
  - SELECT
  - PROJECT
  - UNION
  - DIFFERENCE
  - INTERSECTION
  - CARTESIAN PRODUCT
  - JOIN
  - DIVISION

- Additional Operations
  - Aggregate Functions
  - OUTER JOIN
  - OUTER UNION

SELECT (σ)

- SELECT(σ)는 단일 relation에서 주어진 조건을 만족하는 tuple들만을 선택하여 출력.

- 표기 방법 : $\sigma_{<condition>}(R)$

  - R 은 relation 이름, <condition>은 조건식
  - <condition>은 term들을 AND, OR, NOT 등으로 연결함.
  - 각 term은 다음과 같이 표현.

    attribute *op* value

    or

    attribute *op* attribute

    단, *op* 는 {=, ≠, >, ≥, <, ≤} 의 비교 연산자

SELECT (σ)

R :

A	B	C	D
1	2	2	9
2	2	5	7
3	3	8	3
5	5	9	8

$\sigma_{(A = B) \text{ AND } (D >= 7)}(R)$:

A	B	C	D
2	2	5	7
5	5	9	8

SELECT (σ)

EMP

SSN	name	age	salary
11111	bob	30	30000
33333	jim	28	25000
44444	jim	40	35000
77777	abe	40	18000
88888	sam	55	35000

Q: Get employees whose age is greater than 30

σ age >30 (EMP)

Show query result :

Q: Get employees whose age is 40 and salary is less than $35,000

σ age=40 AND salary<35000 (EMP)

Show query result :

SELECT (σ)

- Relation을 **수평 분할**(Horizontal Partition) 함.

  - 즉 relation의 구조는 그대로 동일; tuple들만을 선택 분할 함
  - 예 : employee들을 age별 (30 이하, 31 – 50, 51 이상)로 분할;

- $|\sigma_{<condition>}(R)| \leq |R|$

- 교환(commutative) 법칙 성립.

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

- 여러 개의 조건들을 AND로 표현할 수 있음.

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

PROJECT (π)

- PROJECT(π)는 단일 relation에서 원하는 attribute들만을 출력

- 표기 방법 : $\pi_{<list>}(R)$
  - <list>는 relation R에서 원하는 attribute들 리스트.
  - 이 리스트에 없는 attribute들은 소거되어서 결과에 나옴.

- Relation을 수직 분할(vertical partition) 함.
  - 즉 relation의 attribute들이 여러 개의 그룹으로 분할 됨.
  - 예 : employee들을 {SSN, name, age}, {SSN, name, salary}
    2 개의 그룹들로 분할;

PROJECT (π)

- PROJECT는 연산 결과에서 중복(duplicated) tuple들이 발생되면, 이들을 하나만 생성되게 소거함.

  - 그 이유는 relation은 "set of tuples" 로 정의되기 때문 : "All tuples in a relation must be distinct"

  - Question : 실제 응용에서 중복 tuple들을 꼭 소거할 필요가 있을 까? SQL 에서는 어떨까?

- $|\pi_{<list>}(R)| \leq |R|$

- If <list> includes a key of R, then $|\pi_{<list>}(R)| = |R|$

  예: $\pi_{<SSN,\ age>}$ (EMPLOYEE)

PROJECT (π)

R :

A	B	C
3	1	1
3	2	1
7	3	1
7	4	2

$\pi_{A, C}(R)$:

A	C
3	1
3	1
7	1
7	2

------>

중복 tuple들 소거

A	C
3	1
7	1
7	2

PROJECT (π)

EMP

SSN	name	age	salary
11111	bob	30	30000
33333	jim	28	25000
44444	jim	40	35000
77777	abe	40	18000
88888	sam	28	35000

Q: Get each employee's ages

π age (EMP)

Show query result :

Q: Get employee's SSN and ages

π ssn, age (EMP)

Show query result :

How to Express?

- Relational algebra의 2 가지 표현 방식

  (1) 연산들을 nesting하여 한꺼번에 표현.

  (2) 한번에 하나씩 연산들을 표현하면서 단계별로 표현.
  (단, 이 경우 중간 결과를 저장하기 위한 relation 이름 지정 필요)

- 예:

  Get the salary of employees who work in department # 5.

  방식 (1) : $\pi_{salary}(\sigma_{DNO=5}(EMPLOYEE))$

  방식 (2) : $DEP5\_EMPS \leftarrow \sigma_{DNO=5}(EMPLOYEE)$

  $RESULT \leftarrow \pi_{salary}(DEP5\_EMPS)$

UNION (∪)

- R ∪ S includes all tuples that are either in relation R or in S.

$$R \cup S = \{t \mid t \in R \text{ or } t \in S\}$$

- Duplicate tuples are eliminated.

- Get the SSNs of all employees who either work in department 5 or whose salary is greater than $30,000.

$$DEP5\_EMPS \leftarrow \pi_{SSN}(\sigma_{DNO=5}(EMPLOYEE))$$

$$HIGH\_SAL\_EMPS \leftarrow \pi_{SSN}(\sigma_{SALARY > 30000}(EMPLOYEE))$$

$$RESULT \leftarrow DEP5\_EMPS \cup HIGH\_SAL\_EMPS$$

UNION (∪)

- R ∪ S includes all tuples that are either in relation R or in S.

$$R \cup S = \{t \mid t \in R \text{ or } t \in S\}$$

- Duplicate tuples are eliminated.

- Get the SSNs of all employees who either work in department 5 or whose salary is greater than $30,000.

$$DEP5\_EMPS \leftarrow \pi_{SSN}(\sigma_{DNO=5}(EMPLOYEE))$$

$$HIGH\_SAL\_EMPS \leftarrow \pi_{SSN}(\sigma_{SALARY > 30000}(EMPLOYEE))$$

$$RESULT \leftarrow DEP5\_EMPS \cup HIGH\_SAL\_EMPS$$

UNION

R :

A	B
3	1
3	2
7	1

S :

A	B
3	2
7	3

R ∪ S :

A	B
3	1
3	2
7	1
7	3

INTERSECTION (∩)

- R ∩ S includes all tuples that are in both R and S.

$$R \cap S = \{t \mid t \in R \text{ and } t \in S\}$$

- Duplicate tuples are eliminated.

- Get the SSNs of all employees who work in department 5 and whose salary is greater than $30,000.

$$DEP5\_EMPS \leftarrow \pi_{SSN}(\sigma_{DNO = 5}(EMPLOYEE))$$

$$HIGH\_SAL\_EMPS \leftarrow \pi_{SSN}(\sigma_{SALARY > 30000}(EMPLOYEE))$$

$$RESULT \leftarrow DEP5\_EMPS \cap HIGH\_SAL\_EMPS$$

INTERSECTION

R :

A	B
3	1
3	2
7	1

S :

A	B
3	2
7	3

R ∩ S :

A	B
3	2

DIFFERENCE (–)

- R – S includes all tuples that are in R, but not in S.

$$R - S = \{t \mid t \in R \text{ and } t \notin S\}$$

- Duplicate tuples are eliminated.

- Get the SSNs of all employees who work in department 5, but whose salary is not greater than $30,000.

$$DEP5\_EMPS \leftarrow \pi_{SSN}(\sigma_{DNO=5}(EMPLOYEE))$$

$$HIGH\_SAL\_EMPS \leftarrow \pi_{SSN}(\sigma_{SALARY > 30000}(EMPLOYEE))$$

$$RESULT \leftarrow DEP5\_EMPS - HIGH\_SAL\_EMPS$$

DIFFERENCE

R :

A	B
3	1
3	2
7	1

S :

A	B
3	2
7	3

R – S :

A	B
3	1
7	1

Type Compatibility:

- ∪, ∩, – 의 3 연산자는 다음 조건을 만족해야 사용가능.

- 즉, relation R과 S는 다음 두 조건을 만족해야 함.

  1) R과 S의 attribute의 개수가 서로 같음.

  2) 같은 위치에 서로 대응하는 attribute의 domain이 같음.

  : domain(Ai) = domain(Bi) for $1 \le i \le n$

- 참고 : attribute의 이름은 서로 달라도 상관 없음.

- 이 경우 (관례상) 결과 relation은 첫 번째 relation R의 attribute 들의 이름들을 가짐.

연산 규칙

- Union과 Intersection : 교환(commutative) 법칙 성립

$$R \cup S = S \cup R$$

$$R \cap S = S \cap R$$

- Union과 Intersection : 결합(associative) 법칙 성립

$$R \cup (S \cup T) = (R \cup S) \cup T$$

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

- Difference : 교환(commutative) 법칙 성립 안 함.

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

CARTESIN PRODUCT (x)

- R x S combines both tuples from relation R and S.

$$R \times S = \{rs \mid r \in R \text{ and } s \in S\}$$

- 즉, R (A_1, A_2, . . ., A_m) x S (B_1, B_2, . . ., B_n) 의 결과는 Q (A_1, A_2, . . ., A_m, B_1, B_2, . . ., B_n)로 정의됨. 단,

  (1) Q는 (m + n) 개의 attribute들을 가짐.

  (2) Q는 R과 S에 속한 각 tuple들을 모두 연결하여 합침

- If R has m tuples (|R| = m 로 표기) and S has n tuples, then, |R x S| = |R| * |S| = m * n .

CARTESIN PRODUCT

R :

A	B
a1	b1
a2	b2

S :

C	D	E
c1	d1	e1
c2	d1	e2
c2	d2	e2
c3	d3	e3

R x S :

A	B	C	D	E
a1	b1	c1	d1	e1
a1	b1	c2	d1	e2
a1	b1	c2	d2	e2
a1	b1	c3	d3	e3
a2	b2	c1	d1	e1
a2	b2	c2	d1	e2
a2	b2	c2	d2	e2
a2	b2	c3	d3	e3

CARTESIN PRODUCT

- Get all combined tuples of both departments and employees.

DEPT

DNO	dname	mgr-ssn
d1	DB	22222
d2	security	33333
d3	network	55555

EMP

ssn	ename	age
11111	abe	28
22222	bob	46
33333	ann	31
44444	jim	50
55555	eve	38

DEPT x EMP

(Total 15 tuples)

DNO	dname	mgr-ssn	ssn	ename	age
d1	DB	22222	11111	abe	28
d1	DB	22222	22222	bob	46
d1	DB	22222	33333	ann	31
d1	DB	22222	44444	jim	50
d1	DB	22222	55555	eve	38
d2	security	33333	11111	abe	28
. . . .	. . . . .	. . . . .	. . . . .	. . . . .	
d3	network	55555	55555	eve	38