Relational Algebra 2

Chapter 5.2

V4.0

Relational Algebra - Example

Consider the following SQL to find which departments have had employees on the \`Further Accounting‘ course.

```
SELECT DISTINCT dname
FROM department, course, empcourse, employee
WHERE course.courseno = empcourse.courseno
    AND empcourse.empno = employee.empno
    AND employee.depno = department.depno
     AND cname = 'Further Accounting';
```

The equivalent relational algebra is:

$\text{PROJECT}_{\text{dname}}$ (department JOIN $_{\text{depno = depno}}$ (
$\text{PROJECT}_{\text{depno}}$ (employee JOIN $_{\text{empno = empno}}$ (
$\text{PROJECT}_{\text{empno}}$ (empcourse JOIN $_{\text{courseno = courseno}}$ (
$\text{PROJECT}_{\text{courseno}}$ (SELECT $_{\text{cname = `Further Accounting‘}}$ course)
))))))

Modern Join

```
SELECT DISTINCT dname
FROM department
    JOIN employee ON employee.depno = department.depno
        JOIN empcourse ON empcourse.empno = employee.empno
            JOIN course ON course.courseno = empcourse.courseno
                AND cname = 'Further Accounting';
```

As Subqueries

```
SELECT DISTINCT dname
FROM department
WHERE depno IN
    (SELECT depno FROM employee
    WHERE empno IN
        (SELECT empno FROM empcourse
        WHERE courseno IN
            (SELECT courseno FROM course
            WHERE cname = 'Further Accounting'
          )
      )
  )
```

Symbolic Notation

From the example, one can see that for complicated cases a large amount of the answer is formed from operator names, such as PROJECT and JOIN. It is therefore commonplace to use symbolic notation to represent the operators.

- SELECT -> σ (sigma)
- PROJECT -> π (pi)
- PRODUCT -> x (times)
- JOIN -> |x| (bow-tie)
- UNION -> ∪ (cup)
- INTERSECTION -> ∩ (cap)
- DIFFERENCE -> - (minus)
- RENAME -> ρ (rho)

Usage

The symbolic operators are used as with the verbal ones. So, to find all employees in department 1:

SELECT $_{\text{depno = 1}}$ (employee)

becomes: $\sigma_{\text{depno = 1}}$ (employee)

Conditions can be combined together using

$\wedge$ (AND) and $\vee$ (OR).

For example, all employees in department 1 called `Smith':

SELECT $_{\text{depno = 1} \wedge \text{surname = `Smith'}}$ (employee)

becomes: $\sigma_{\text{depno = 1} \wedge \text{surname = `Smith'}}$ (employee)

Usage Cont…

The use of the symbolic notation can lend itself to brevity. Even better, when the JOIN is a natural join, the JOIN condition may be omitted from |x|. Our first example resulted in:

PROJECT $_{\text{dname}}$ (department JOIN $_{\text{depno = depno}}$ (
 PROJECT $_{\text{depno}}$ (employee JOIN $_{\text{empno = empno}}$ (
  PROJECT $_{\text{empno}}$ (empcourse JOIN $_{\text{courseno = courseno}}$ (
   PROJECT $_{\text{courseno}}$ (SELECT $_{\text{cname = `Further Accounting'}}$ course)))))))

becomes

π_{dname} (department |x| (
 π_{depno} (employee |x| (
  π_{empno} (empcourse |x| (
   π_{courseno} ($\sigma_{\text{cname = `Further Accounting'}}$ course)))))))

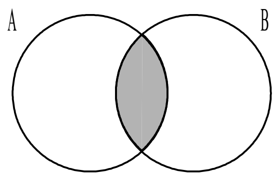
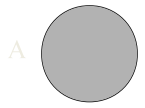
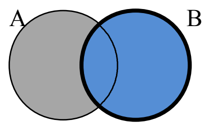
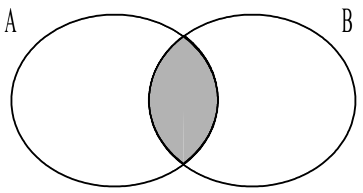
Rename Operator (rho)

The rename operator returns an existing relation under a new name. $\rho_A(B)$ is the relation B with its name changed to A. For example, find all the employees in the same Department as employee 3.

$$\pi_{\text{emp2.surname, emp2.forenames}} (\sigma_{\text{employee.empno} = 3 \wedge \text{employee.depno} = \text{emp2.depno}} (\text{employee} \times (\rho_{\text{emp2}} \text{employee})))$$

Derivable Operators

- Fundamental operators: σ , π , × , ∪ , -, ρ
- Derivable operators: |x|, ∩ e.g. **A ∩ B ⇔ A - (A – B)**

A ∩ B	⇔	A -	(A – B)
			
			

Deriving $|\times|$ from π , σ , and $\times$

- $A |\times|_{c} B \Leftrightarrow \pi_{a1,a2,...aN} (\sigma_{c} (A \times B))$
  - where **c is the join condition** (eg A.a1 = B.a1),
  - and a1,a2,...aN are all the attributes of A and B without repetition.
- c is usually the comparison of primary and foreign key.
- Where there are N tables, there are usually N-1 join-conditions.
- In the case of a natural join, the conditions can be left out, but otherwise leaving out conditions results in a cartesian product (a common mistake to make).

Equivalences

The same relational algebraic expression can be written in many different ways. The order in which tuples appear in relations is never significant.

- $A \times B \Leftrightarrow B \times A$
- $A \cap B \Leftrightarrow B \cap A$
- $A \cup B \Leftrightarrow B \cup A$
- (A - B) is not the same as (B - A)
- $\sigma_{c1} (\sigma_{c2} (A)) \Leftrightarrow \sigma_{c2} (\sigma_{c1} (A)) \Leftrightarrow \sigma_{c1 \wedge c2} (A)$
- $\pi_{a1}(A) \Leftrightarrow \pi_{a1}(\pi_{a1,etc}(A))$, where etc is any other attribute of A.
- ...

While equivalent expressions always give the same result, some may be much easier to evaluate than others.

When any query is submitted to the DBMS, its query optimiser tries to find the most efficient equivalent expression before evaluating it.

Comparing RA and SQL

- Relational algebra:
  - is closed (the result of every expression is a relation)
  - has a rigorous foundation
  - has simple semantics
  - is used for reasoning, query optimisation, etc.
- SQL:
  - is a superset of relational algebra (has “extras”)
  - has convenient formatting features, etc.
  - provides aggregate functions
  - has complicated semantics
  - is an end-user language.

Comparing RA and SQL

Any relational language as powerful as relational algebra is called relationally complete.

A relationally complete language can perform all basic, meaningful operations on relations.

Since SQL is a superset of relational algebra, it is also relationally complete.