

Computer Science COMP2400, Lab 3

August 21, 2007

1 Introduction

This exercise is an introduction to Relational Algebra. The notation used to express relational Algebra is:

π_A	Projection onto the set of attributes A
σ_p	Selection (restriction) by predicate p
$\rho_A(R)$	Rename a relation and/or attributes
\cup	Union of two relations
\cap	Intersection of two relations
\times	Cartesian product of two relations
\bowtie_C	Relational join of two relations with join condition C .
$\mathcal{R}\mathcal{F}_L$	Aggregation of unique values of R with group functions L

Please review the lecture notes and/or text to refresh your memory of the meaning of these operators.

We will be looking at relational queries against the company schema from the E&N and Lab 1. Lab3.zip on the course web site includes a full definition of this schema including the tables you were asked to create in the lab exercise, plus some more sample data.

2 Exercises

Work *in pairs* to complete the following exercises.

2.1 SQL to Relational Algebra

Translate the following SQL statements into relational algebra.

1. `SELECT * FROM employee`
2. `SELECT fname ,lname FROM employee`
3. `SELECT fname ,lname FROM employee WHERE enumber=1234`
4. `SELECT * FROM employee , department`

5. `SELECT * FROM employee, department
WHERE employee.dno = department.dnumber`
6. `SELECT pname, sum(hours) FROM project p, works_on w
WHERE p.pnumber = w.pno
GROUP BY pname`
7. `SELECT pname, dname, sum(hours)
FROM project p, works_on w, employee e, department d
WHERE p.pnumber = w.pno
AND e.ename = w.eno
AND d.dnumber = e.dno
GROUP BY pname, dname`

2.2 Relational Algebra to SQL

Translate the following relational algebra expressions into SQL statements.

1. $\pi_{enumber, salary}(\text{employee})$
2. $\pi_{dname, fname, lname}(\sigma_{enumber=mgr_empid}(\text{employee} \times \text{department}))$
- 3.

<code>mgr_names</code>	<code>←</code>	$\pi_{fname, lname, enumber}(\text{employee} \bowtie_{enumber=mgr_empid} \text{department})$
<code>work_on_projects</code>	<code>←</code>	$\pi_{eno, hours, pname}(\text{works_on} \bowtie_{pno=pnumber} \text{project})$
<code>mgr_works_on</code>	<code>←</code>	$\text{department} \bowtie_{mgr_empid=eno} \text{works_on_projects}$
<code>mgr_hours_worked</code>	<code>←</code>	$\pi_{mgr_empid, pname, hours}(\text{mgr_works_on})$
<code>named_hours</code>	<code>←</code>	$\pi_{fname, lname, hours}(\text{mgr_names} \bowtie_{enumber=eno} \text{mgr_hours_worked})$

$fname, lname \mathcal{F}_{\text{sum}(hours)} \text{named_hours}$

2.3 Query Costs

Relational Algebra is an ‘operational’ language for defining queries, in that it specifies a precise sequence of operations for evaluating a query. For this reason, it can be used as an intermediate representation in real database systems, particularly when alternative execution strategies are evaluated. This exercise takes a brief look at how a database might do this.

In this exercise, we will look at alternative execution strategies for a moderately complex query, and decide which one will execute most quickly. To do this, we need to assign a cost to a relational algebra expression, by defining a real-valued function $\mathcal{C}[\mathcal{E}]$ that gives the cost of the expression \mathcal{E} . The definition we will use is:

$$\begin{aligned} \mathcal{C}[\sigma_p \mathcal{E}] &= \log_2 |E| \\ \mathcal{C}[E \bowtie F] &= \log_2 |E| + \log_2 |F| \end{aligned}$$

All other operators are “free”.

1. Translate the following query into Relational Algebra, using project and join operations. Write the expression using intermediate values for the result of each ‘select’ and ‘join’ operation.

```

SELECT enumber , hours
FROM employee e , works_on w , project p , department d
WHERE e . enumber = w . eno
AND    w . pno = p . pnumber
AND    p . dnum = d . dnumber
AND    d . dname = 'Information Technology '

```

Begin by joining the employee relation with the works_on relation.

2. For each step in your RA expression, calculate the number of rows in the intermediate result.
3. Apply the cost function given above to your sequence of steps.
4. Rearrange the query (ie write down a new sequence of steps) in a way that will produce a cheaper query. Calculate the cost of the new query.

If you have time, try using the same analysis on query 3 in Section 2.2.

2.4 Query Equivalence using Relational Algebra—*Optional*

This exercise requires some inventiveness and mathematical creativity, and not everyone is expected to complete this. The attempt is worthwhile however, since expressing the first query in relational algebra is a good test of your understanding of SQL.

The SQL statements

```

SELECT fname , lname , dno FROM employee
WHERE enumber = (SELECT mgr_empid FROM department
                 WHERE dnumber = dno)

```

and

```

SELECT fname , lname , dno
FROM employee , department
WHERE enumber = mgr_empid

```

return the same result. Translate both queries into relational algebra, and you should be able to establish mathematically that these queries are identical.

2.5 More SQL queries—*Optional*

Like most languages, SQL becomes easier with practice. Here are some more interesting SQL queries on the same schema.

1. Which employee has the highest salary ?
2. Which employees have not worked on the 'Red tape is Fun' project ?
3. Assuming that employees are paid for a 35 hour week, 48 weeks/year, what is the cost of each project at current salary levels ?
4. Which employees work on the most time consuming project ?
5. Which employee has contributed the most hours to projects run by departments they do not belong to.