CHAPTER 6

Formal Relational Query Languages

Practice Exercises

6.1 Write the following queries in relational algebra, using the university schema.

   a. Find the titles of courses in the Comp. Sci. department that have 3 credits.

   b. Find the IDs of all students who were taught by an instructor named Einstein; make sure there are no duplicates in the result.

   c. Find the highest salary of any instructor.

   d. Find all instructors earning the highest salary (there may be more than one with the same salary).

   e. Find the enrollment of each section that was offered in Autumn 2009.

   f. Find the maximum enrollment, across all sections, in Autumn 2009.

   g. Find the sections that had the maximum enrollment in Autumn 2009

Answer:

   a. \( \Pi_{title} (\sigma_{dept\_name = 'Comp. Sci' \land credits=3} (course)) \)

   b. \( \Pi_{ID} (\sigma_{IID = 'Einstein' \land takes \times \Pi_{IID,course\_id,section\_id,semester,year} (teaches)}) \)

   Assuming the set version of the relational algebra is used, there is no need to explicitly remove duplicates. If the multiset version is used, the grouping operator can be used without any aggregation to remove duplicates. For example given relation \( r(A,B) \) possibly containing duplicates, \( A_B \mathcal{G}(r) \) would return a duplicate free version of the relation.

   c. \( \mathcal{G}_{max(salary)}(instructor) \)
Chapter 6  Formal Relational Query Languages

d.  $instructor \bowtie G_{\text{max}(\text{salary})} as \text{salary}(instructor)$
Note that the above query renames the maximum salary as salary, so
the subsequent natural join outputs only instructors with that salary.

e.  $course_id, section_id as G_{\text{count(s)}} as enrollment(\sigma_{\text{year}=2009 \land \text{semester}=\text{Autumn}(\text{takes}))}

f.  $t1 \leftarrow course_id, section_id as G_{\text{count(s)}} as enrollment(\sigma_{\text{year}=2009 \land \text{semester}=\text{Autumn}(\text{takes}))$
result = $G_{\text{max}(\text{enrollment})}(t1)$

g.  $t2 \leftarrow G_{\text{max}(\text{enrollment}) as enrollment}(t1)$
where $t1$ is as defined in the previous part of the question.
result = $tI \bowtimes t2$

6.2 Consider the relational database of Figure 6.22, where the primary keys are
underlined. Give an expression in the relational algebra to express each of
the following queries:

a. Find the names of all employees who live in the same city and on the
same street as do their managers.

b. Find the names of all employees in this database who do not work
for “First Bank Corporation”.

c. Find the names of all employees who earn more than every employee
of “Small Bank Corporation”.

Answer:

a.  $\Pi_{\text{person.name}}((\text{employee } \bowtie \text{manages})$
$\bowtie\{\text{manager.name} = \text{employee2.person.name} \land \text{employee.street} = \text{employee2.street}$
$\land \text{employee.city} = \text{employee2.city}(p_{\text{employee2}}(\text{employee})))$

b. The following solutions assume that all people work for exactly one
company. If one allows people to appear in the database (e.g. in
employee) but not appear in works, the problem is more complicated.
We give solutions for this more realistic case later.

$c.  \Pi_{\text{person.name}}(\sigma_{\text{company.name} \neq \text{“First Bank Corporation”}}(\text{works}))$
If people may not work for any company:

$c.  \Pi_{\text{person.name}}(\text{employee}) - \Pi_{\text{person.name}}(\sigma_{\text{company.name} = \text{“First Bank Corporation”}}(\text{works}))$

c.  $\Pi_{\text{person.name}}(\text{works}) - (\Pi_{\text{works.person.name}}(\text{works}$
$\bowtie\{\text{works.salary} \leq \text{works2.salary} \land \text{works2.company.name} = \text{“Small Bank Corporation”}$
$p_{\text{works2}}(\text{works}))$

6.3 The natural outer-join operations extend the natural-join operation so that
tuples from the participating relations are not lost in the result of the join.
Describe how the theta-join operation can be extended so that tuples from the left, right, or both relations are not lost from the result of a theta join. 

Answer:

a. The left outer theta join of \( r(R) \) and \( s(S) \) can be defined as:
\[
(r \bowtie_0 s) \cup ((r - \Pi_R(r \bowtie_0 s)) \times (null, null, \ldots, null))
\]
The tuple of nulls is of size equal to the number of attributes in \( S \).

b. The right outer theta join of \( r(R) \) and \( s(S) \) can be defined as:
\[
(r \bowtie_0 s) \cup ((null, null, \ldots, null) \times (s - \Pi_S(r \bowtie_0 s)))
\]
The tuple of nulls is of size equal to the number of attributes in \( R \).

c. The full outer theta join of \( r(R) \) and \( s(S) \) can be defined as:
\[
(r \bowtie_0 s) \cup ((null, null, \ldots, null) \times (s - \Pi_S(r \bowtie_0 s))) \cup ((r - \Pi_R(r \bowtie_0 s)) \times (null, null, \ldots, null))
\]
The first tuple of nulls is of size equal to the number of attributes in \( R \), and the second one is of size equal to the number of attributes in \( S \).

6.4 (Division operation): The division operator of relational algebra, “\( \div \)”, is defined as follows. Let \( r(R) \) and \( s(S) \) be relations, and let \( S \subseteq R \); that is, every attribute of schema \( S \) is also in schema \( R \). Then \( r \div s \) is a relation on schema \( R - S \) (that is, on the schema containing all attributes of schema \( R \) that are not in schema \( S \)). A tuple \( t \) is in \( r \div s \) if and only if both of two conditions hold:

- \( t \) is in \( \Pi_{R-S}(r) \)
- For every tuple \( t_s \) in \( s \), there is a tuple \( t_r \) in \( r \) satisfying both of the following:
  a. \( t_r[S] = t_s[S] \)
  b. \( t_r[R - S] = t \)

Given the above definition:

a. Write a relational algebra expression using the division operator to find the IDs of all students who have taken all Comp. Sci. courses. (Hint: project \( \text{takes} \) to just ID and \( \text{course id} \), and generate the set of all Comp. Sci. \( \text{course ids} \) using a select expression, before doing the division.)

b. Show how to write the above query in relational algebra, without using division. (By doing so, you would have shown how to define the division operation using the other relational algebra operations.)

Answer:

a. \( \Pi_D(\Pi_{ID,CourseId}(\text{takes}) \div \Pi_{CourseId}(\sigma_{\text{department}='\text{Comp. Sci.'}}(\text{course}))) \)

b. The required expression is as follows:
\[
r \leftarrow \Pi_{ID,CourseId}(\text{takes})
\]
Let the following relation schemas be given:

\[ R = (A, B, C) \]
\[ S = (D, E, F) \]

Let relations \( r(R) \) and \( s(S) \) be given. Give an expression in the tuple relational calculus that is equivalent to each of the following:

a. \( \Pi_A(r) \)

b. \( \sigma_{B = 17}(r) \)

c. \( r \times s \)

d. \( \Pi_{A,F}(\sigma_{C = D}(r \times s)) \)

**Answer:**

a. \( \{ t \mid \forall q \in r \ (q[A] = t[A]) \} \)

b. \( \{ t \mid t \in r \land t[B] = 17 \} \)

c. \( \{ t \mid \exists p \in r \ \exists q \in s \ (t[A] = p[A] \land t[B] = p[B] \land t[C] = p[C] \land t[D] = q[D] \land t[E] = q[E] \land t[F] = q[F]) \} \)
Practice Exercises 5

6.6 Let \( R = (A, B, C) \), and let \( r_1 \) and \( r_2 \) both be relations on schema \( R \). Give an expression in the domain relational calculus that is equivalent to each of the following:

a. \( \Pi_A(r_1) \)

b. \( \sigma_{B=17}(r_1) \)

c. \( r_1 \cup r_2 \)

d. \( r_1 \cap r_2 \)

e. \( r_1 - r_2 \)

f. \( \Pi_{A,B}(r_1) \bowtie \Pi_{B,C}(r_2) \)

Answer:

a. \( \{< t > \mid \exists p, q ( < t, p, q > \in r_1) \} \)

b. \( \{< a, b, c > \mid < a, b, c > \in r_1 \land b = 17 \} \)

c. \( \{< a, b, c > \mid < a, b, c > \in r_1 \lor < a, b, c > \in r_2 \} \)

d. \( \{< a, b, c > \mid < a, b, c > \in r_1 \land < a, b, c > \in r_2 \} \)

e. \( \{< a, b, c > \mid < a, b, c > \in r_1 \land < a, b, c > \not\in r_2 \} \)

f. \( \{< a, b, c > \mid \exists p, q ( < a, b, p > \in r_1 \land < q, b, c > \in r_2) \} \)

6.7 Let \( R = (A, B) \) and \( S = (A, C) \), and let \( r(R) \) and \( s(S) \) be relations. Write expressions in relational algebra for each of the following queries:

a. \( \{< a > \mid \exists b ( < a, b > \in r \land b = 7) \} \)

b. \( \{< a, b, c > \mid < a, b > \in r \land < a, c > \in s \} \)

c. \( \{< a > \mid \exists c ( < a, c > \in s \land \exists b_1, b_2 ( < a, b_1 > \in r \land < c, b_2 > \in r \land b_1 > b_2)) \} \)

Answer:

a. \( \Pi_A(\sigma_{B=17}(r)) \)

b. \( r \bowtie s \)

c. \( \Pi_A(s \bowtie (\Pi_{c,A}(\sigma_{c,b=17,B}(r \bowtie \rho_{b,c}(r)))))) \)

6.8 Consider the relational database of Figure 6.22 where the primary keys are underlined. Give an expression in tuple relational calculus for each of the following queries:

a. Find all employees who work directly for “Jones.”

b. Find all cities of residence of all employees who work directly for “Jones.”
Chapter 6 Formal Relational Query Languages

c. Find the name of the manager of the manager of “Jones.”

d. Find those employees who earn more than all employees living in the city “Mumbai.”

Answer:

a. 
\[ \{ t \mid \exists m \in \text{manages} (t[\text{person name}] = m[\text{person name}] \land m[\text{manager name}] = 'Jones') \} \]

b.
\[ \{ t \mid \exists m \in \text{manages} \exists e \in \text{employee} (e[\text{person name}] = m[\text{person name}] \land m[\text{manager name}] = 'Jones' \land t[\text{city}] = e[\text{city}]) \} \]

c.
\[ \{ t \mid \exists m1 \in \text{manages} \exists m2 \in \text{manages} (m1[\text{manager name}] = m2[\text{person name}] \land m1[\text{person name}] = 'Jones' \land t[\text{manager name}] = m2[\text{manager name}]) \} \]

d.
\[ \{ t \mid \exists w1 \in \text{works} \neg \exists w2 \in \text{works} (w1[\text{salary}] < w2[\text{salary}] \land \exists e2 \in \text{employee} (w2[\text{person name}] = e2[\text{person name}] \land e2[\text{city}] = 'Mumbai')) \} \]

6.9 Describe how to translate join expressions in SQL to relational algebra.

Answer: A query of the form

\[
\text{select } A1, A2, \ldots, An \\
\text{from } R1, R2, \ldots, Rm \\
\text{where } P
\]

can be translated into relational algebra as follows:

\[ \Pi_{A1, A2, \ldots, An} (\sigma_P(R1 \times R2 \times \ldots \times Rm)) \]

An SQL join expression of the form

\[ R1 \text{ natural join } R2 \]

can be written as \( R1 \bowtie R2 \).

An SQL join expression of the form

\[ R1 \text{ join } R2 \text{ on } (P) \]

can be written as \( R1 \bowtie_P R2 \).
An SQL join expression of the form

\[ R_1 \textbf{join} R_2 \textbf{using} (A_1, A_2, \ldots, A_n) \]

can be written as \( \Pi_S(R_1 \bowtie R_2. A_1 = R_1. A_1 \wedge R_1. A_2 = R_2. A_2 \wedge \ldots \wedge R_1. A_n = R_2. A_n) \)

where \( S \) is \( A_1, A_2, \ldots, A_n \) followed by all attributes of \( R_1 \) other than \( R_1.A_1, R_1.A_2, \ldots, R_1.A_n \), followed by all attributes of \( R_2 \) other than \( R_2.A_1, R_2.A_2, \ldots, R_2.A_n \).

The outer join versions of the SQL join expressions can be similarly written by using \( \bowtie \), \( \bowtie \) and \( \bowtie \) in place of \( \bowtie \).

The most direct way to handle subqueries is to extend the relational algebra. To handle where clause subqueries, we need to allow selection predicates to contain nested relational algebra expressions, which can reference correlation attributes from outer level relations. Scalar subqueries can be similarly translated by allowing nested relational algebra expressions to appear in scalar expressions. An alternative approach to handling such subqueries used in some database systems, such as Microsoft SQL Server, introduces a new relational algebra operator called the Apply operator; see Chapter 30, page 1230-1231 for details. Without such extensions, translating subqueries into standard relational algebra can be rather complicated.

\[ ^1 \text{The case of outer joins with the using clause is a little more complicated; with a right outer join it is possible that } R_1.A_1 \text{ is null, but } R_2.A_1 \text{ is not, and the output should contain the non-null value. The SQL coalesce function can be used, replacing } S \text{ by coalesce}(R_1.A_1, R_2.A_1), \text{ coalesce}(R_1.A_2, R_2.A_2), \ldots, \text{ coalesce}(R_1.A_n, R_2.A_n) \text{, followed by the other attributes of } R_1 \text{ and } R_2. \]