

Relational Algebra Exercises

1. Consider a database with the following schema:

<i>Person</i> (name, age, gender)	name is a key
<i>Frequents</i> (name, pizzeria)	(name, pizzeria) is a key
<i>Eats</i> (name, pizza)	(name, pizza) is a key
<i>Serves</i> (pizzeria, pizza, price)	(pizzeria, pizza) is a key

Write relational algebra expressions for the following nine queries. (Warning: some of the later queries are a bit challenging.)

If you know SQL, you can try running SQL queries to match your relational algebra expressions. We've created a [file for download](#) with schema declarations and sample data. (See our [quick guide](#) for SQL system instructions.) To check your queries, the correct results are found in the answers section below.

- Find all pizzerias frequented by at least one person under the age of 18.
- Find the names of all females who eat either mushroom or pepperoni pizza (or both).
- Find the names of all females who eat both mushroom and pepperoni pizza.
- Find all pizzerias that serve at least one pizza that Amy eats for less than \$10.00.
- Find all pizzerias that are frequented by only females or only males.
- For each person, find all pizzas the person eats that are not served by any pizzeria the person frequents. Return all such person (name) / pizza pairs.
- Find the names of all people who frequent only pizzerias serving at least one pizza they eat.
- Find the names of all people who frequent every pizzeria serving at least one pizza they eat.
- Find the pizzeria serving the cheapest pepperoni pizza. In the case of ties, return all of the cheapest-pepperoni pizzerias.

2. Consider a schema with two relations, $R(A, B)$ and $S(B, C)$, where all values are integers. Make no assumptions about keys. Consider the following three relational algebra expressions:

- $\pi_{A,C}(R \bowtie \sigma_{B=1} S)$
- $\pi_A(\sigma_{B=1} R) \times \pi_C(\sigma_{B=1} S)$
- $\pi_{A,C}(\pi_A R \times \sigma_{B=1} S)$

Two of the three expressions are equivalent (i.e., produce the same answer on all databases), while one of them can produce a different answer. Which query can produce a different answer? Give the simplest database instance you can think of where a different answer is produced.

3. Consider a relation $R(A, B)$ that contains r tuples, and a relation $S(B, C)$ that contains s tuples; assume $r > 0$ and $s > 0$. Make no assumptions about keys. For each of the following relational algebra expressions, state in terms of r and s the minimum and maximum number of tuples that could be in the result of the expression.

- a. $R \cup \rho_{S(A,B)} S$
- b. $\pi_{A,C}(R \bowtie S)$
- c. $\pi_B R - (\pi_B R - \pi_B S)$
- d. $(R \bowtie R) \bowtie R$
- e. $\sigma_{A>B} R \cup \sigma_{A<B} R$

4. Two more exotic relational algebra operators we didn't cover are the *semijoin* and *antijoin*. Semijoin is the same as natural join, except only attributes of the first relation are returned in the result. For example, if we have relations *Student*(ID, name) and *Enrolled*(ID, course), and not all students are enrolled in courses, then the query "*Student* \bowtie *Enrolled*" returns the ID and name of all students who are enrolled in at least one course. In the general case, $E_1 \bowtie E_2$ returns all tuples in the result of expression E_1 such that there is at least one tuple in the result of E_2 with matching values for the shared attributes. Antijoin is the converse: $E_1 \triangleright E_2$ returns all tuples in the result of expression E_1 such that there are no tuples in the result of E_2 with matching values for the shared attributes. For example, the query "*Student* \triangleright *Enrolled*" returns the ID and name of all students who are not enrolled in any courses.

Like some other relational operators (e.g., intersection, natural join), semijoin and antijoin are abbreviations - they can be defined in terms of other relational operators. Define $E_1 \bowtie E_2$ in terms of other relational operators. That is, give an equation " $E_1 \bowtie E_2 = ??$ ", where ?? on the right-hand side is a relational algebra expression that doesn't use semijoin. Similarly, give an equation " $E_1 \triangleright E_2 = ??$ ", where ?? on the right-hand side is a relational algebra expression that doesn't use antijoin.

5. Consider a relation *Temp*(regionID, name, high, low) that records historical high and low temperatures for various regions. Regions have names, but they are identified by regionID, which is a key. Consider the following query, which uses the linear notation introduced at the end of the relational algebra videos.

$$\begin{aligned}
 T1(rID, h) &= \pi_{regionID, high} Temp \\
 T2(rID, h) &= \pi_{regionID, low} Temp \\
 T3(regionID) &= \pi_{rID}(T1 \bowtie_{h < high} Temp) \\
 T4(regionID) &= \pi_{rID}(T2 \bowtie_{l > low} Temp) \\
 T5(regionID) &= \pi_{regionID} Temp - T3 \\
 T6(regionID) &= \pi_{regionID} Temp - T4 \\
 Result(n) &= \pi_{name}(Temp \bowtie (T5 \cup T6))
 \end{aligned}$$

State in English what is computed as the final Result. The answer can be articulated in a single phrase.

Show Answers