CS143: Relational Algebra

Book Chapters
(4th) Chapters 3.2
(5th) Chapters 2.2-3
(6th) Chapter 2.6
(7th) Chapter 2.6

Things to Learn

- Relational algebra
  - Select, Project, Join, ...

Database query language

- Data Manipulation Language (DML): A language to query and update relations

What is a query?

- Oxford English Dictionary: A question, especially one addressed to an official or organization
- Database jargon for question (complex word for simple concept)
- Questions to get answers from a database
  - Example: Get the students who are taking all CS classes but no Physics class

- Some queries are easy to pose, some are not
- Some queries are easy for DBMS to answer, some are not

Relational query languages

- Relational algebra (formal), SQL (practical)

- Relational Query:
  - A query is executed against input relations and produces an output relation
    \[
    \text{Input relations} \rightarrow \text{Query} \rightarrow \text{Output relation}
    \]
  - Very useful: “Piping” is possible
Relational Algebra

*Input relations (set) → **query** → Output relation (set)*

- Set semantics. no duplicate tuples. duplicates are eliminated
- In contrast, multiset semantics for SQL (performance reason)

Examples to Use

- School information
  - Student(sid, name, addr, age, GPA)

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>addr</th>
<th>age</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>301</td>
<td>John</td>
<td>183 Westwood</td>
<td>19</td>
<td>2.1</td>
</tr>
<tr>
<td>303</td>
<td>Elaine</td>
<td>301 Wilshire</td>
<td>17</td>
<td>3.9</td>
</tr>
<tr>
<td>401</td>
<td>James</td>
<td>183 Westwood</td>
<td>17</td>
<td>3.5</td>
</tr>
<tr>
<td>208</td>
<td>Esther</td>
<td>421 Wilshire</td>
<td>20</td>
<td>3.1</td>
</tr>
</tbody>
</table>

- Class(dept, cnum, sec, unit, title, instructor)

<table>
<thead>
<tr>
<th>dept</th>
<th>cnum</th>
<th>sec</th>
<th>unit</th>
<th>title</th>
<th>instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>112</td>
<td>01</td>
<td>03</td>
<td>Modeling</td>
<td>Dick Muntz</td>
</tr>
<tr>
<td>CS</td>
<td>143</td>
<td>01</td>
<td>04</td>
<td>DB Systems</td>
<td>John Cho</td>
</tr>
<tr>
<td>EE</td>
<td>143</td>
<td>01</td>
<td>03</td>
<td>Signal</td>
<td>Dick Muntz</td>
</tr>
<tr>
<td>ME</td>
<td>183</td>
<td>02</td>
<td>05</td>
<td>Mechanics</td>
<td>Susan Tracey</td>
</tr>
</tbody>
</table>

- Enroll(sid, dept, cnum, sec)

<table>
<thead>
<tr>
<th>sid</th>
<th>dept</th>
<th>cnum</th>
<th>sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>301</td>
<td>CS</td>
<td>112</td>
<td>01</td>
</tr>
<tr>
<td>301</td>
<td>CS</td>
<td>143</td>
<td>01</td>
</tr>
<tr>
<td>303</td>
<td>EE</td>
<td>143</td>
<td>01</td>
</tr>
<tr>
<td>303</td>
<td>CS</td>
<td>112</td>
<td>01</td>
</tr>
<tr>
<td>401</td>
<td>CS</td>
<td>112</td>
<td>01</td>
</tr>
</tbody>
</table>

Simplest query: relation name

- **Query 1:** All students

SELECT operator

Select all tuples satisfying a condition

- **Query 2:** Students with age < 18
- **Query 3:** Students with GPA > 3.7 and age < 18

- **Notation:** $\sigma_C(R)$
  - Filters out rows in a relation
  - $C$: A boolean expression with attribute names, constants, comparisons ($>, \leq, \neq, \ldots$) and connectives ($\land, \lor, \neg$)
  - $R$ can be either a relation or a result from another operator

**PROJECT operator**
- **Query 4:** sid and GPA of all students

- **Query 5:** All departments offering classes
  - Relational algebra removes duplicates (set semantics)
  - SQL does not (multiset or bag semantics)

- **Notation:** $\pi_A(R)$
  - Filters out columns in a relation
  - $A$: a set of attributes to keep

- **Query 6:** sid and GPA of all students with age < 18
  - We can “compose” multiple operators

- **Q:** Is it ever useful to compose two projection operators next to each other?

- **Q:** Is it ever useful to compose two selection operators next to each other?
CROSS PRODUCT (CARTESIAN PRODUCT) operator

- Example: $R \times S$

\[
\begin{array}{c}
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_1</td>
<td>b_1</td>
</tr>
<tr>
<td>a_1</td>
<td>b_2</td>
</tr>
<tr>
<td>a_1</td>
<td>b_3</td>
</tr>
<tr>
<td>a_2</td>
<td>b_1</td>
</tr>
<tr>
<td>a_2</td>
<td>b_2</td>
</tr>
<tr>
<td>a_2</td>
<td>b_3</td>
</tr>
</tbody>
</table>
\end{array}
\]

- Concatenation of tuples from both relations
- One result tuple for each pair of tuples in $R$ and $S$
- If column names conflict, prefix with the table name

- **Notation:** $R_1 \times R_2$
  - $R_1 \times R_2 = \{ t \mid t = \langle t_1, t_2 \rangle \text{ for } t_1 \in R_1 \text{ and } t_2 \in R_2 \}$

- **Q:** Looks odd to concatenate unrelated tuples. Why use $\times$?

- **Query 7:** Names of students who take CS courses
  - Explanation: start with the query requiring sid, not name

- **Q:** Can we write it differently?
  - Benefit of RDBMS. It figures out the best way to compute.

- **Q:** If $|R| = r$ and $|S| = s$, what is $|R \times S|$?

NATURAL JOIN operator

- Example: Student $\bowtie$ Enroll
  - Shorthand for $\sigma_{\text{Student.sid} = \text{Enroll.sid}}$ (Student $\times$ Enroll)

- **Notation:** $R_1 \bowtie R_2$
– Concatenate tuples horizontally
– Enforce equality on common attributes
– We may assume only one copy of the common attributes are kept

• **Query 8:** Names of students who take CS classes (Same as before)

• **Query 9:** Names of students taking classes offered by “Dick Muntz”

• Natural join: The most natural way to join two tables

**RENAME operator**

• **Query 10:** Find the pairs of student names who live in the same address.
• What about $\pi_{\text{name}, \text{name}}(\sigma_{\text{addr} = \text{addr}}(\text{Student} \times \text{Student}))$?

• **Notation:** $\rho_S(R)$ – rename $R$ to $S$
• **Notation:** $\rho_{S(A_1', A_2')}(R)$ for $R(A_1, A_2)$ – rename $R(A_1, A_2)$ to $S(A_1', A_2')$
• **Q:** Is $\pi_{\text{Student}.\text{name}, S.\text{name}}(\sigma_{\text{Student}.\text{addr} = S.\text{addr}}(\text{Student} \times \rho_S(\text{Student})))$ really correct?
  – How many times (John, James) returned?

**UNION operator**

• **Query 11:** Find all student and instructor names.
  – **Q:** Can we do it with cross product or join?

• **Notation:** $R \cup S$
  – Union of tuples from $R$ and $S$
  – The schemas of $R$ and $S$ should be the same
  – No duplicate tuples in the result
DIFFERENCE operator

- **Query 12:** Find the courses (dept, cnum, sec) that no student is taking
  - How can we find the courses that at least one student is taking?

- **Notation:** $R - S$
  - Schemas of $R$ and $S$ must match exactly

- **Query 13:** What if we want to get the titles of the courses?
  - Very common. To match schemas, we lose information. We have to join back.

INTERSECT operator

- **Query 14:** Find the instructors who teach both CS and EE courses
  - Q: Can we answer this using only selection and projection?

- **Notation:** $R \cap S = R - (R - S)$
  - Draw Venn Diagram to verify

More questions

- **Q:** sids of students who did not take any CS courses?
  - Q: Is $\pi_{\text{sid}}(\sigma_{\text{title} \neq \text{CS}}(\text{Enroll}))$ correct?

  - Q: What is its complement?

- **General advice:** When a query is difficult to write, think in terms of its complement.
Relational algebra: things to remember

- Data manipulation language (query language)
  - Relation $\rightarrow$ algebra $\rightarrow$ relation
- Relational algebra: set semantics, SQL: bag semantics
- Operators: $\sigma$, $\times$, $\bowtie$, $\rho$, $\cup$, $\setminus$, $\cap$
- General suggestion: If difficult to write, consider its complement