CS143 Notes: Relational Algebra

Book Chapters

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

Things to Learn

• Relational algebra
  – Select, Project, Join, …

Steps in Database Construction

1. Domain Analysis
2. Database design
3. Table creation: DDL
4. Load: bulk-load
5. Query and update: DML

Database query language

What is a query?

• 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

- Formal: Relational algebra, relational calculus, datalog
- Practical: SQL (← relational algebra), Quel (← relational calculus), QBE (← datalog)
- Relational Query:
  - Data sits in a disk
  - Submit a query
  - Get an answer

\[ \text{Input relations} \rightarrow \text{query} \rightarrow \text{Output relation} \]

Executed against a set of relations and produces a relation

* Important to know
* Very useful: “Piping” is possible

Relational Algebra

\[ \text{Input relations (set)} \rightarrow \text{query} \rightarrow \text{Output relation (set)} \]

- 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)`
    - `sid`  `name`  `addr`  `age`  `GPA`
    - 301 John  183 Westwood 19  2.1
    - 303 Elaine 301 Wilshire 17  3.9
    - 401 James  183 Westwood 17  3.5
    - 208 Esther 421 Wilshire 20  3.1
  - `Class(dept, cnum, sec, unit, title, instructor)`
    - `dept`  `cnum`  `sec`  `unit`  `title`  `instructor`
    - CS  112 01 03 Modeling  Dick Muntz
    - CS  143 01 04 DB Systems  Carlo Zaniolo
    - EE  143 01 03 Signal  Dick Muntz
    - ME  183 02 05 Mechanics  Susan Tracey
  - `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 ($>$, $\leq$, $\neq$, ...)
    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: $R \times S$

  \[
  \begin{array}{c}
  A \\
  a_1 \\
  a_2 \\
  \end{array} \times \begin{array}{c}
  B \\
  b_1 \\
  b_2 \\
  b_3 \\
  \end{array} = \begin{array}{cc}
  A & B \\
  a_1 & b_1 \\
  a_1 & b_2 \\
  a_1 & b_3 \\
  a_2 & b_1 \\
  a_2 & b_2 \\
  a_2 & b_3 \\
  \end{array}
  \]

  – 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 and addresses of students who take CS courses with GPA < 3
• 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 ⊲◁ Enroll
  
  – Shorthand for $\sigma_{\text{Student.sid} = \text{Enroll.sid}}$ (Student × 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 taking CS classes
  
  – Explanation: start with the query requiring sid, not name
• Query 9: Names of students taking classes offered by “Carlo Zaniolo”

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

THETA JOIN operator
• Example: Student ⊲◁ Student.sid=Enroll.sid∧GPA>3.7 Enroll

• Notation: $R_1 \bowtie_C R_2 = \sigma_C(R_1 \times R_2)$
• Generalization of natural join
- Often implemented as the basic operation in DBMS

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

- **Notation:** $\rho_S(R)$ – rename $R$ to $S$
- **Notation:** $\rho_{S(A1', A2')}(R)$ for $R(A1, A2)$ – rename $R(A1, A2)$ to $S(A1', A2')$
- **Q:** Is $\pi_{\text{Student}.name, \text{S}.name}(\sigma_{\text{Student}.addr=S.addr}(\text{Student} \times \rho_S(\text{Student})))$ 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

**DIVISION operator**

Use the boards very carefully keeping all examples.

• Division operator is not used directly by any one
• But how we compute the answer for division is very important to learn
• Learn how we computed the answer, not the operator

• **Query 15:** Find the student sids who take every CS class available
  – Q: What will be the answer?

  – Q: How can we compute it?
  
  *Give time to think about*

  * **Step 1:** We need to know which student is taking which class. Where do we get the student sid and the courses they take(sid, dept, cnum, sec)?
Step 2: We also need to know all CS classes. How do we get the set of all CS courses (dept, cnum, sec)?

Step 3: What does the relation from Step 1 look like if all students take all CS courses?

How can we compute this? Let us call this relation $R$

Step 4: What does $(R - Enroll)$ look like? What’s its meaning?

Step 5: What is the meaning of the projection of Step 4?

Step 6: How can we get the student sids who take all CS courses?

Notation: $R/S$.

Query 16: Find All $A$ values in $R$ such that the values appear with all $B$ values in $S$.

\[
\begin{array}{|c|c|}
\hline
R & S \\
\hline
A & B \\
\hline
a_1 & b_1 & b_1 \\
a_1 & b_2 & b_2 \\
a_2 & b_1 & b_1 \\
a_3 & b_2 & \ \\
\hline
\end{array}
\]

$R \approx$ students and classes they take

$S \approx$ all CS classes

$R/S \approx$ All students who take all CS classes

Formal definition:

* We assume $R(A,B)$ and $S(B)$
* The set of all $a \in R.A$ such that $\langle a, b \rangle \in R$ for every $b \in S$
* $R/S = \{a \mid a \in R.A$ and $\langle a, b \rangle \in R$ for all $b \in S\}$

Result for the example:

\[
\begin{array}{|c|}
\hline
A \\
a_1 \\
\hline
\end{array}
\]

Q: How to compute it?
Ans: $\frac{R}{S} = \pi_A(R) - \pi_A((\pi_A(R) \times S) - R)$

All $(R.A, S.B)$ pairs
* $\pi_A(R)$: All $A$’s
* $S$: All $B$’s
* $\pi_A(R) \times S$: all $R.A$ and $S.B$ pairs

$\pi_A(R) \times S - R$: $(A, B)$ pairs that are missing in $R$
$\pi_A(\pi_A(R) \times S - R)$: All $R.A$’s that do not have some $S.B$

- Analogy with integer division
  * In integer: $\frac{R}{S}$ is the largest integer $T$ such that $T \times S \leq R$
  * In relational algebra: $\frac{R}{S}$ is the largest relation $T$ such that $T \times S \subseteq R$

- The division operator is not used often, but how to compute it is important

More questions
- **Q**: sids of students who did not take any CS courses?
  - **Q**: Is $\pi_{sid}(\sigma_{title\neq‘CS’}(Enroll))$ 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: $\sigma, \times, \bowtie, \rho, \cup, -, \cap, /$
- General suggestion: If difficult to write, consider its complement