Introduction to Database Systems
CSE 414

Lecture 11: More Relational Algebra

Announcements

• WQ4/HW4 released
  – Both due next Tuesday
• Please make sure you get your AWS set up!
  – Will need for HW6
• Do not use seaquill for data storage
  – Machine gets wiped out periodically

Relational Algebra Operators

• Union \( \cup \), intersection \( \cap \), difference
• Selection \( \sigma \)
• Projection \( \pi \)
• Cartesian product \( \times \), join \( \bowtie \)
• (Rename \( \rho \))
• Duplicate elimination \( \delta \)
• Grouping and aggregation \( \gamma \)
• Sorting \( \# \)

All operators take in 1 or more relations as inputs and return another relation

Composing RA Operators

Patient

<table>
<thead>
<tr>
<th>no</th>
<th>name</th>
<th>zip</th>
<th>disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>p1</td>
<td>98125</td>
<td>flu</td>
</tr>
<tr>
<td>2</td>
<td>p2</td>
<td>98125</td>
<td>heart</td>
</tr>
<tr>
<td>3</td>
<td>p3</td>
<td>98120</td>
<td>lung</td>
</tr>
<tr>
<td>4</td>
<td>p4</td>
<td>98120</td>
<td>heart</td>
</tr>
</tbody>
</table>

\( \sigma_{\text{disease}=\text{heart}}(\text{Patient}) \)

\( \pi_{\text{zip, disease}}(\sigma_{\text{disease}=\text{heart}}(\text{Patient})) \)

Natural Join

\( R1 \bowtie R2 \)

• Meaning: \( R1 \bowtie R2 = \Pi_A(\sigma_\theta(R1 \times R2)) \)

• Where:
  – Selection \( \sigma_\theta \) checks equality of all common attributes (i.e., attributes with same names)
  – Projection \( \Pi_A \) eliminates duplicate common attributes

Join Summary

• Theta-join: \( R \bowtie_\theta S = \sigma_\theta (R \times S) \)
  – Join of R and S with a join condition \( \theta \)
  – Cross-product followed by selection \( \theta \)
  – No projection
• Equijoin: \( R \bowtie_\theta S = \sigma_\theta (R \times S) \)
  – Join condition \( \theta \) consists only of equalities
  – No projection
• Natural join: \( R \bowtie S = \Pi_A (\sigma_\theta (R \times S)) \)
  – Equality on all fields with same name in R and in S
  – Projection \( \Pi_A \) drops all redundant attributes
Some Examples

Supplier(sno, sname, scity, sstate)
Part(pno, pname, psize, pcolor)
Supply(sno, pno, qty, price)

Name of supplier of parts with size greater than 10
\( \pi_{\text{sname}}(\text{Supplier} \bowtie (\text{Supply} \bowtie (\sigma_{\text{psize}>10}(\text{Part})))) \)

Name of supplier of red parts or parts with size greater than 10
\( \pi_{\text{sname}}(\text{Supplier} \bowtie (\text{Supply} \bowtie (\sigma_{\text{psize}>10}(\text{Part}))) \cup \sigma_{\text{pcolor} = \text{red}}(\text{Part}))) \)

Can be represented as trees as well

Representing RA Queries as Trees

Relational Algebra Operators
- Union (\( \cup \)), intersection (\( \cap \)), difference (\( - \))
- Selection (\( \sigma \))
- Projection (\( \pi \))
- Cartesian product (\( \times \)), join (\( \bowtie \))
- Rename (\( \rho \))
- Duplicate elimination (\( \delta \))
- Grouping and aggregation (\( \gamma \))
- Sorting (\( \tau \))

All operators take in 1 or more relations as inputs and return another relation

Extended RA: Operators on Bags
- Duplicate elimination (\( \delta \))
- Grouping (\( \gamma \))
  - Takes in relation and a list of grouping operations (e.g., aggregates). Returns a new relation.
- Sorting (\( \tau \))
  - Takes in a relation, a list of attributes to sort on, and an order. Returns a new relation.

Using Extended RA Operators

Extended RA
Typical Plan for a Query (1/2)

Answer
\[ \pi_{\text{fields}} \]
\[ \sigma_{\text{selection condition}} \]
\[ \text{SELECT PROJECT JOIN Query} \]
\[ \sigma_{\text{join condition}} \]
\[ … \]
\[ R \]
\[ S \]

Typical Plan for a Query (1/2)

\[ \sigma_{\text{having condition}} \]
\[ \gamma_{\text{fields, sum/count/min/max(fields)}} \]
\[ \text{SELECT fields} \]
\[ \text{FROM R, S, … WHERE condition} \]
\[ \pi_{\text{fields}} \]
\[ \sigma_{\text{where condition}} \]
\[ \text{JOIN condition} \]
\[ \text{GROUP BY fields} \]
\[ \text{HAVING condition} \]

How about Subqueries?

\[ \text{SELECT Q.sno} \]
\[ \text{FROM Supplier AS Q} \]
\[ \text{WHERE Q.sstate = 'WA'} \]
\[ \text{and not exists} \]
\[ \text{(SELECT *)} \]
\[ \text{FROM Supply AS P} \]
\[ \text{WHERE P.sno = Q.sno and P.price > 100)} \]

How about Subqueries?

Option 1: create nested plans

\[ \text{SELECT Q.sno} \]
\[ \text{FROM Supplier AS Q} \]
\[ \text{WHERE Q.sstate = 'WA'} \]
\[ \text{and not exists} \]
\[ \text{(SELECT *)} \]
\[ \text{FROM Supply AS P} \]
\[ \text{WHERE P.sno = Q.sno and P.price > 100)} \]

How about Subqueries?

\[ \text{SELECT Q.sno} \]
\[ \text{FROM Supplier AS Q} \]
\[ \text{WHERE Q.sstate = 'WA'} \]
\[ \text{and not exists} \]
\[ \text{(SELECT *)} \]
\[ \text{FROM Supply AS P} \]
\[ \text{WHERE P.sno = Q.sno and P.price > 100)} \]

How about Subqueries?

\[ \text{SELECT Q.sno} \]
\[ \text{FROM Supplier AS Q} \]
\[ \text{WHERE Q.sstate = 'WA'} \]
\[ \text{and not exists} \]
\[ \text{(SELECT *)} \]
\[ \text{FROM Supply AS P} \]
\[ \text{WHERE P.sno = Q.sno and P.price > 100)} \]
How about Subqueries?

```
(SELECT Q.sno
 FROM Supplier AS Q
 WHERE Q.sstate = 'WA')
EXCEPT
(SELECT P.sno
 FROM Supply AS P
 WHERE P.price > 100)
```

Finally...

```
(SELECT Q.sno
 FROM Supplier AS Q
 WHERE Q.sstate = 'WA')
EXCEPT
(SELECT P.sno
 FROM Supply AS P
 WHERE P.price > 100)
```

Summary of RA and SQL

- SQL = a declarative language where we say what data we want to retrieve
- RA = an algebra where we say how we want to retrieve the data
- Theorem: SQL and RA can express exactly the same class of queries

```
RDBMS translate SQL → RA, then optimize RA
```

Datalog v.s. RA (and SQL)

- Datalog without recursion, but with negation and aggregates expresses the same queries as RA: next slides

```
RA to Datalog by Examples
```

```
Union:
R(A,B,C) U S(A,B,C)

U(x,y,z) :- R(x,y,z).
U(x,y,z) :- S(x,y,z).
```
RA to Datalog by Examples

Intersection:
\( R(A,B,C) \cap S(A,B,C) \)

\[ I(x,y,z) : \neg R(x,y,z), S(x,y,z). \]

Selection: \( \sigma_{A>100 \text{ and } B='foo'} \) (R)
\( L(x,y,z) : R(x,y,z), x > 100, y='foo'. \)

Selection: \( \sigma_{A>100 \text{ or } B='foo'} \) (R)
\( L(x,y,z) : R(x,y,z), x > 100. \)
\( L(x,y,z) : R(x,y,z), y='foo'. \)

Selection: \( \sigma_{A>100} \) (R)
\( L(x,y,z) : R(x,y,z), x > 100. \)

Equi-join: \( R \bowtie S \)(A=S.D and B=S.E)
\( J(x,y,z,q) : R(x,y,z), S(x,y,q). \)

Projection: \( \Pi_A(R) \)
\( P(x) : R(x,y,z). \)

To express difference, we add negation
\( R - S \)
\( D(x,y,z) : R(x,y,z), \neg S(x,y,z). \)

Examples

Translate: \( \Pi_A(\sigma_{B=3} (R)) \)
\( A(a) : R(a,3,\_). \)

Underscore used to denote an "anonymous variable"
Each such variable is unique
Examples

Translate: $\Pi_{A,B,C}(\sigma_{B=3}(R) \bowtie_{R.A=5.D} \sigma_{E=5}(S))$

$A(a) : R(a,3,\_) S(a,5,\_).$

These are different \_\_'s.