RA/SQL Division, SQL JOINs

CS430/630 Lecture 9

Slides based on “Database Management Systems” 3rd ed. Ramakrishnan and Gehrke

Division: / operator in Relational Algebra

- Division is used to answer queries such as:
  - Find sailors who have reserved all boats. (or all blue boats)
  - Find suppliers that catalog all green parts.
- Let $A$ have 2 fields, $x$ and $y$; $B$ have only field $y$:
  - $A/B = \{x \mid \exists y \in A \forall y \in B\}$
    - $A/B$ contains all $x$ tuples (sailors) such that for every $y$ tuple (boat) in $B$, there is an $xy$ tuple in $A$ (reservation by $x$ for $y$)
    - Or, if the set of $y$ values (boats) associated with an $x$ value (sailor) in $A$ contains all $y$ values in $B$, the $x$ value is in $A/B$.
- In general, $x$ and $y$ can be any sets of fields (not just singletons, but we won’t cover those cases)

Visualizing Relationships

Let $A = \text{sid}$ and bid columns of reserves (as table)

Let $B = \text{bid}$ column of boats (as table)

Then $A/B = \text{table containing only sid sid2}$, the answer to the query about which sailors reserved all boats.

If $A = B \times C$, it’s simple...

- If $A = B \times C$, then $A/B = C$ and $A/C = B$
- This means $A$ has all possible $(x, y)$ pairs
- Example: $A = B = C = \begin{bmatrix}
  \text{sid} & \text{bid} \\
  s1 & b1 \\
  s1 & b2 \\
  s1 & b3 \\
  s1 & b4 \\
  s2 & b1 \\
  s2 & b2 \\
  s3 & b2 \\
  s4 & b2 \\
  s4 & b4 \\
\end{bmatrix}$

Examples of Division $A/B$

- $A/B$ finds the $x$ values (si) in $A$ that have all the $y$ values (bi) as $(x, y)$ rows in $A$.
- Here the sailors who reserved all four boats in $B$. 

But most $(x, y)$ relationships are more complicated...

Example: $A = \text{sid}$ and bid columns of reserves =

<table>
<thead>
<tr>
<th>sid</th>
<th>bid</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>b1</td>
</tr>
<tr>
<td>s1</td>
<td>b2</td>
</tr>
<tr>
<td>s1</td>
<td>b3</td>
</tr>
<tr>
<td>s1</td>
<td>b4</td>
</tr>
<tr>
<td>s2</td>
<td>b1</td>
</tr>
<tr>
<td>s2</td>
<td>b2</td>
</tr>
<tr>
<td>s2</td>
<td>b2</td>
</tr>
<tr>
<td>s3</td>
<td>b2</td>
</tr>
<tr>
<td>s4</td>
<td>b2</td>
</tr>
<tr>
<td>s4</td>
<td>b4</td>
</tr>
</tbody>
</table>

- Here, sailor s1 reserved all boats (b1-b4), but the others reserved an incomplete set of boats
- So there’s no “factoring” of $A$ into two simple tables, but we still want to find the sailors who reserved all boats, or a certain set of boats.
Examples of Division $A \div B$ with smaller $B$-sets, i.e. smaller “for all” sets

<table>
<thead>
<tr>
<th>sid</th>
<th>bid</th>
<th>bid</th>
<th>bid</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>b1</td>
<td>b2</td>
<td>b1</td>
</tr>
<tr>
<td>s1</td>
<td>b2</td>
<td>B1</td>
<td>b4</td>
</tr>
<tr>
<td>s1</td>
<td>b3</td>
<td></td>
<td>b2</td>
</tr>
<tr>
<td>s1</td>
<td>b4</td>
<td></td>
<td>B3</td>
</tr>
<tr>
<td>s2</td>
<td>b1</td>
<td>s1</td>
<td></td>
</tr>
<tr>
<td>s2</td>
<td>b2</td>
<td>s1</td>
<td>s4</td>
</tr>
<tr>
<td>s3</td>
<td>b2</td>
<td>s3</td>
<td>s1</td>
</tr>
<tr>
<td>s4</td>
<td>b2</td>
<td>s4</td>
<td>s1</td>
</tr>
<tr>
<td>s4</td>
<td>b4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>A/B1</td>
<td>A/B2</td>
<td>A/B3</td>
</tr>
</tbody>
</table>

Division problems we will cover

- Although it is possible to use division with various sized tables, we will only cover cases of $A / B$ for 2-column $A$ and 1-column $B$.
- This will cover the common case of $A$ being the “relationship” table (pared down to the two id columns) and $B$ being the id column of one of the related entities, the “for all” condition.
- For example, the table of sid, bid of the last slide is $A = (s1, s2, s3, s4)$ and table of bids is $B = \pi_{\text{bid}}\text{Boats}$
- So the division is $(\pi_{\text{sid, bid}}\text{Reserves})/\pi_{\text{bid}}\text{Boats}$
- Providing the set of sids of sailors who reserved all the boats.

Query 1

“Find the names of sailors who’ve reserved all boats”

$\rho(\text{Tempsids},(\pi_{\text{sid, bid}}\text{Reserves})/(\pi_{\text{bid}}\text{Boats}))$

$\pi_{\text{name}}(\text{Temp} \bowtie\text{Sailors})$

Expressing $A \div B$ Using Basic Operators

- For $A \div B$, compute all $x$ values that are not disqualified by some $y$ value in $B$
  - $x$ value is disqualified if by attaching $y$ value from $B$, we obtain an $xy$ tuple that is not in $A$.
- All possible $xy$ tuples:
  - $\pi_x(A) \bowtie B$:
  - $\pi_x(A) \bowtie B \Rightarrow A$

Disqualified $x$ values: $\pi_x((\pi_x(A) \bowtie B) \Rightarrow A)$

$A/B$: $\pi_x(A) -$ all disqualified tuples

$\pi_x(A) - \pi_x((\pi_x(A) \bowtie B) \Rightarrow A)$

Not that this is impressive, but not useful in practice. It does show one (inefficient) way to do division in SQL.

Query 2

“Find sailors who’ve reserved all red boats”

$\rho(\text{Temp},(\pi_{\text{sid, bid}}\text{Reserves})/(\pi_{\text{bid}}\sigma_{\text{color}=\text{red}}\text{Boats}))$

$\pi_{\text{name}}(\text{Temp} \bowtie\text{Sailors})$

Find disqualified $x$’s for $A \div B$

Example: $A = (s1, b1, s2, b2, s3, b3, s4, b4)$

- Here, sailor $s1$ reserved all boats (b1-b4), but the others reserved an incomplete set of boats
- Example: sailor $s2$ doesn’t have <s2, b3>, so x=s2 is disqualified from $A \div B$.

$\pi_x(A) = (s1, s2, s3, s4)$

$\pi_x(A) \bowtie B = ((s1, b1), (s1, b3), (s2, b1), (s3, b2), (s4, b2), (s4, b4))$ (16 tuples)

Not here: (s2, b3) (s3, b1) (s3, b3) (s4, b3) (s4, b4)

$\pi_x((\pi_x(A) \bowtie B) \Rightarrow A) = (s2, s3, s4)$

= all disqualified tuples, leaving only $s1$ qualified
Division in SQL

- Not supported as primitive operator
- Need to use nested queries to express division
  - One of the most subtle queries in SQL
  - Need to pay close attention to writing SQL division queries!
- There are two well-known ways of writing division queries
  - Using the set EXCEPT operator (2-level nesting)
  - Without the EXCEPT operator (3-level nesting)

Division: Solution 1 (Q9, pg. 150)

“Find sailors who’ve reserved all boats.”

With `EXCEPT` (use `MINUS` in Oracle):

```sql
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS
    ( 
        -- look for bid(s) unconnected via R to this sid
        (SELECT B.bid FROM Boats B)
        EXCEPT
        -- ones connected via R to S.sid
        (SELECT R.bid FROM Reserves R WHERE R.sid=S.sid)
    )
```

Note: the parentheses around the two inner `SELECT`s are not needed: “subquery EXCEPT subquery” qualifies as a subquery, and the two are surrounded by parentheses needed by `exists` (...)

Division: Solution 2 (also on pg. 150)

“Find sailors who’ve reserved all boats.”

Without `EXCEPT`: so works on mysql

```sql
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS
    ( SELECT B.bid FROM Boats B
    WHERE NOT EXISTS
        (SELECT * FROM Reserves R WHERE R.bid=B.bid AND R.sid=S.sid)
    )
```

Division: Solution 3 (not in text)

“Find sailors who’ve reserved all boats.”

```sql
SELECT r.sid
FROM Reserves r
WHERE r.bid IN (SELECT b.bid from Boats b)
GROUP BY r.sid
HAVING COUNT(distinct bid) = (SELECT COUNT(*) FROM Boats)
```

The `HAVING` group for a certain `sid` qualifies the `sid` if it contains all the bids, shown by proper count. The distinct is needed because there can be multiple Reserves rows for one `sid`-bid combination (R is not a binary relationship table).

Note: if Reserves can have null sids, this solution can report a null as an output, unlike the earlier solutions, because `GROUP BY` groups nulls together as a legitimate group. (But we will only use division on non-null values.)
“Find sailors who’ve reserved all red boats.”

“Find sailors who’ve reserved all red boats.”

With **EXCEPT**:

```sql
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS (  
  (SELECT B.bid FROM Boats B
   WHERE B.color = 'red')
EXCEPT
  (SELECT R.bid FROM Reserves R
   WHERE R.sid=S.sid) )
```

“Find sailors who’ve reserved all red boats.”

Without **EXCEPT**:

```sql
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS (  
  (SELECT B.bid FROM Boats B
   WHERE B.color = 'red' AND NOT EXISTS (SELECT *
     FROM Reserves R
     WHERE R.bid=B.bid
     AND R.sid=S.sid))
```

**Another Example**

```sql
Movies (movie_id, title, year, studio)
Stars_In (actor_id, movie_id, character)

"Find names of actors who star in ALL movies produced by Universal in year 1990."
SELECT ???
```

```sql
Movies (movie_id, title, year, studio)
Stars_In (actor_id, movie_id, character)

"Find names of actors who star in ALL movies produced by Universal in year 1990."
SELECT A.name FROM Actors A
WHERE NOT EXISTS(  
  SELECT M.movie_id FROM Movies M
  WHERE M.year=1990 AND M.studio='Universal'
EXCEPT
  SELECT S.movie_id FROM Stars_In S
  WHERE S.actor_id=A.actor_id)
```

**Join Expressions**

- SQL keywords for operations we already saw
- Cross Product:
  - Sailors **CROSS JOIN** Reserves
- Condition Join:
  - Sailors **JOIN** Reserves **ON** <condition>
- Natural Join: uses same-named columns for join columns
  - Sailors **NATURAL JOIN** Reserves

**Usage Example:**

```sql
SELECT *
FROM Sailors **JOIN** Reserves **ON** Sailors.sid=Reserves.sid
```
The syntax for an inner join that uses table aliases

```
SELECT select_list
FROM table_1 n1
[INNER] JOIN table_2 n2
ON n1.column_name operator n2.column_name
[INNER] JOIN table_3 n3
ON n2.column_name operator n3.column_name...
```

Note: The INNER keyword is optional and doesn’t change the meaning or action of the join. It just emphasizes that the JOIN is not an OUTER JOIN, coming up soon…

An inner join example from Murach Chap. 4

```
SELECT invoice_number, vendor_name, invoice_due_date,
    (invoice_total - payment_total - credit_total) AS balance_due
FROM vendors v JOIN invoices i
ON v.vendor_id = i.vendor_id
WHERE (invoice_total - payment_total - credit_total) > 0
ORDER BY invoice_due_date DESC
```

The result set

(!40 rows selected!)

Note: uses ORDER BY, not covered in R&G Chap. 5, but should be.

SQL in fig4-02a.sql
(fig4-ap_tables.sql first load the needed tables with)

A join with a table from another schema

```
SELECT vendor_name, customer_last_name, customer_first_name, vendor_state AS state,
    vendor_city AS city
FROM vendors v JOIN om.customers c
ON v.vendor_zip_code = c.customer_zip
ORDER BY state, city
```

The result set

(37 rows)

Oracle: another schema is another user’s schema

Mysql: another schema is another database

The same join with one condition in a WHERE clause

```
SELECT invoice_number, invoice_date, invoice_total, line_item_amt
FROM invoices i JOIN invoice_line_items li
ON i.invoice_id = li.invoice_id
WHERE i.invoice_total > li.line_item_amt
ORDER BY invoice_number
```

The result set

(6 rows selected)

SQL in fig4-04b.sql

A self-join that returns vendors from cities in common with other vendors

```
SELECT DISTINCT v1.vendor_name, v1.vendor_city, v1.vendor_state
FROM vendors v1 JOIN vendors v2
ON (v1.vendor_city = v2.vendor_city) AND
   (v1.vendor_id <> v2.vendor_id)
ORDER BY v1.vendor_state, v1.vendor_city
```

The result set

(84 rows selected)

SQL in fig4-05.sql
A SELECT statement that joins four tables

```
SELECT vendor_name, invoice_number, invoice_date, line_item_amt, account_description
FROM vendors v
JOIN invoices i ON v.vendor_id = i.vendor_id
JOIN invoice_line_items li ON i.invoice_id = li.invoice_id
JOIN general_ledger_accounts gl ON li.account_number = gl.account_number
WHERE (invoice_total - payment_total - credit_total) > 0
ORDER BY vendor_name, line_item_amt DESC
```

The result set

<table>
<thead>
<tr>
<th>Vendor Name</th>
<th>Invoice Number</th>
<th>Invoice Date</th>
<th>Line Item Amount</th>
<th>Account Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC Company</td>
<td>12345</td>
<td>2023-01-01</td>
<td>1000</td>
<td>ABC Sales</td>
</tr>
<tr>
<td>DEF Corp</td>
<td>67890</td>
<td>2023-02-02</td>
<td>2000</td>
<td>DEF Services</td>
</tr>
<tr>
<td>GHI Inc</td>
<td>09876</td>
<td>2023-03-03</td>
<td>3000</td>
<td>GHI Consulting</td>
</tr>
</tbody>
</table>

(44 rows selected)