Unary Operations Reminder

- Selection (σ) ↔ WHERE
- Projection (π) ↔ SELECT
- Renaming (ρ) ↔ AS

Set Operations Summary

Set operations are binary operations that can be applied to two relations with the same attribute sets.

- Set union (U)
- Set intersection (∩)
- Set difference (—)

Set Union

Set union takes two relations (with the same attribute sets) and makes a new relation which contains any tuple that was in either of the original relations (without duplicates, of course).

cf. UNION operation in SQL

Letting a and b represent relations, we write:

\[ a \cup b \]

Set Intersection

Set intersection takes two relations (with the same attribute sets) and makes a new relation which contains any tuple that was in both of the original relations.

cf. INTERSECT operation in SQL

Letting a and b represent relations, we write:

\[ a \cap b \]
Set Difference

Set union takes two relations (with the same attribute sets) and makes a new relation which contains all tuples from the first relation which are not in the second relation.

cf. EXCEPT operation in SQL

Letting a and b represent relations, we write:

\[ a - b \]

Set Operations Example

<table>
<thead>
<tr>
<th>a</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>42</td>
</tr>
<tr>
<td>orange</td>
<td>19</td>
</tr>
<tr>
<td>cherry</td>
<td>77</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>b</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>banana</td>
<td>8</td>
</tr>
<tr>
<td>apple</td>
<td>42</td>
</tr>
<tr>
<td>coconut</td>
<td>17</td>
</tr>
</tbody>
</table>

\[ a \cup b \]

\[ a \cap b \]

\[ a - b \]

Set Operation Properties

Set union and intersection are commutative:

\[ a \cup b = b \cup a \]
\[ a \cap b = b \cap a \]

and associative:

\[ a \cup (b \cup c) = (a \cup b) \cup c \]
\[ a \cap (b \cap c) = (a \cap b) \cap c \]

Set difference is neither commutative nor associative!

Intersection can be expressed in terms of difference (and thus is not strictly necessary):

\[ a \cap b = a - (a - b) \]

Cartesian Products & Joins

Cartesian Product

The Cartesian or cross product in RA has the same meaning as a cross product in SQL: every tuple from one operand is paired with every tuple from the other operand.

We write

\[ a \times b \]

Cartesian Product Notes

- Renaming of attributes may be necessary to avoid duplicate attribute names
- If a has m attributes and b has n attributes, then \( a \times b \) has \((m + n)\) attributes
- The size (# of tuples) of \( a \times b \) is

\[ |a| \times |b| = |a||b| \]
Join

As in SQL, cross product has little utility on its own, and is typically followed by a selection to eliminate meaningless rows:

\[ R_1 = \text{mines_courses} \times \text{mines_cs_faculty} \]
\[ R_2 = \sigma_{\text{instructor} = \text{name}}(R_1) \]

Since this is such a common usage, we have join notation to combine the two operations:

\[ \sigma_{\text{condition}}(a \times b) = a \bowtie_{\text{condition}} b \]

Join Notes (1)

- A join in RA corresponds to an inner join in SQL (note it is equivalent to a “WHERE-clause join”).
- Outer joins are not part of the basic relational algebra (they are an extension).

Join Notes (2)

Some terminology – you won’t be tested on this, but you should be aware of it.

When we have

\[ a \bowtie_{\text{condition}} b \]

and condition is of the general form

\[ \text{cond}_1 \text{ AND cond}_2 \text{ AND ...} \]

and \( \text{cond}_i \) is of the form

\[ a_j \Theta b_k \]

where \( a_j \) is some attribute of \( a \) and \( b_k \) is some attribute of \( b \), and \( \Theta \) is one of the usual comparison operators (=, <, etc.), THEN, the join is called a “theta-join”.

Join Notes (3)

More terminology:

When we have a theta-join where the operation \( (\Theta) \) is equality (=), the join is called an “equijoin”.

Note that an equijoin has two attributes that agree on every single value.

Join Notes (4)

More terminology:

If we want to perform an equijoin on two relations, \( a \) and \( b \), such that the join condition will equate attributes in \( a \) and \( b \) with the same names, we don’t / can’t keep both of the same-named attributes.

The operation in which we a) do the equijoin and b) eliminate the duplicate attributes all at once is called a “natural join”.

The course textbook uses the notation

\[ a \ast b \]

for a natural join, but other sources use

\[ a \bowtie b \]

(without a condition), so beware.

MATHY GOODNESS
Division

There is also a division operator:

\[ a \div b \]

Essentially, the inverse of cross product.

It has no direct equivalent in SQL.

We won’t study it further.

Completeness

It can be shown that the set \( \{\sigma, \pi, \rho, U, -, \times\} \) forms a complete set of operators for the (basic) relational algebra (everything we’ve looked at so far).

That is, you can express all the other operators using some combination of the above.

Just thought you should know.

Aggregates and Grouping

- Not expressible in base relational algebra
- Obviously important for DBMS creators!
- Notations vary
- We won’t be dealing with these.

Outer Joins

Left outer join:

\[ a \bowtie b \]

Right outer join:

\[ a \bowtie b \]

Full outer join:

\[ a \bowtie b \]

Up Next

- Next time (10/17):
  Quiz review & help session.
- Friday, 10/19:
  Quiz 2 on ERD