Relational Algebra

CS 4400

Relational Algebra

Relational model specifies stuctures and constraints, relational algebra provides retrieval operations

- Formal foundation for relational model operations
- Basis for internal query optimization in RDBMS
- Parts of relational algebra found in SQL

Basic Rules

- Relational algebra expressions operate on relations and produce relations as results
- Relational algebra expressions can be chained

`SELECT`

$\sigma_{<condition>}(R)$

- R is the name of a relation
- $< condition >$ is a boolean condition on the values of attributes in the tuples of R

The select operation returns all the tuples from R for which $< condition >$ is true.

`SELECT` Example

Given the following data for `pet`:

```
+------------+----+--------+-----------+
| shelter_id | id | name   | breed     |
+------------+----+--------+-----------+
```

```
|          1 |  1 | Chloe  | Mix       |
|          1 |  2 | Dante  | GSD       |
|          1 |  3 | Heidi  | Dachshund |
|          2 |  1 | Bailey | Mix       |
|          2 |  2 | Sophie | Lab       |
|          2 |  3 | Heidi  | Dachshund |
+------------+----+--------+-----------+
```

$\sigma_{breed='mix'}(pet)$ returns:

```
+------------+----+--------+-----------+
| shelter_id | id | name   | breed     |
+------------+----+--------+-----------+
|          1 |  1 | Chloe  | Mix       |
|          2 |  1 | Bailey | Mix       |
+------------+----+--------+-----------+
```

Properties of `SELECT`

- Result of $\sigma_{<condition>}(R)$ has same schema as R, i.e., same attributes
- SELECT is commutative, e.g.,

  $\sigma_{<c1>}(\sigma_{<c2>}(R)) = \sigma_{<c2>}(\sigma_{<c1>}(R))$
- Cascaded SELECTs can be replaced by single SELECT with conjuction of conditions, e.g.

  $\sigma_{<c1>}(\sigma_{<c2>}(R)) = \sigma_{<c1>AND<c2>}(R)$
- Result of $\sigma_{<condition>}(R)$ has equal or fewer tuples than R

`PROJECT`

$\pi_{<attributelist>}(R)$

- R is the name of a relation
- $< attributelist >$ is a subset of the attributes of relation R

The project operation returns all the tuples in R but with only the attributes in $< attribute - list >$

`PROJECT` Example

```
+------------+----+--------+-----------+
| shelter_id | id | name   | breed     |
+------------+----+--------+-----------+
```

1	1	Chloe	Mix
1	2	Dante	GSD
1	3	Heidi	Dachshund
2	1	Bailey	Mix
2	2	Sophie	Lab
2	3	Heidi	Dachshund

$\pi_{name,breed}(pet) =$

name	breed
Chloe	Mix
Dante	GSD
Heidi	Dachshund
Bailey	Mix
Sophie	Lab

Notice that the duplicate tuple <Heidi, Dachshund> was removed. Results of relational algebra operations are sets.

Properties of PROJECT

- Number of tuples returned by PROJECT is less than or equal to the number of tuples in the input relation because result is a set, i.e., $|\pi_{<attrs>}(R)| \leq |R|$
  - What if $< attrs >$ includes a key of R?
- PROJECT is *not* commutative. In fact $\pi_{<attrs1>}(\pi_{<attrs2>}(R))$ is only a correct expression if $< attrs2 >$ contains the attributes in $< attrs1 >$. In this case the result is simply $\pi_{<attrs1>}(R)$.

Combining PROJECT and SELECT

shelter_id	id	name	breed
1	1	Chloe	Mix
1	2	Dante	GSD
1	3	Heidi	Dachshund
2	1	Bailey	Mix
2	2	Sophie	Lab
2	3	Heidi	Dachshund

```
+------------+----+--------+-----------+
```

$\pi_{name}(\sigma_{breed='Mix'}(pet))$ produces:

```
+--------+
| name   |
+--------+
| Chloe  |
| Bailey |
+--------+
```

Intermediate Results

Previous *in-line expression* could be split up into multiple steps with named intermediate results.

$\pi_{name}(\sigma_{breed='Mix'}(pet))$

becomes:

$MIXES \leftarrow \sigma_{breed='Mix'}(pet)$

$RESULT \leftarrow \pi_{name}(MIXES)$

`RENAME`

- Rename relation R to S:

  $\rho_S(R)$
- Rename attributes of R to $B_1, ...B_n$:

  $\rho_{(B_1,...,B_n)}(R)$
- Rename R to S and attributes to $B_1, ...B_n$:

  $\rho_{S(B_1,...,B_n)}(R)$

Binary Operators

- UNION, $R \cup S$, is set of all tuples in either R or S
- INTERSECTION, $R \cap S$, is set of all tuples in both R and S
- SET DIFFERENCE, $R - S$, is set of all tuples in R but not in S

Operands must be *union compatible*, or *type compatible*. For R and S to be union compatible:

- Degree of R bust be same as degree of S
- For each attribute A_i in R and B_i in S, $dom(A_i) = dom(B_i)$

Cartesian Product

$R \times S$ Creates "super-tuples" by concatenating every tuple in R with every tuple in S.

$R(A_1, ..., A_n) \times S(B_1, ..., B_m) = Q(A_1, ..., A_n, B_1, ..., B_m)$

Notice that

- Q has degree $n + m$
- $|q(Q)| = |r(R)| \times |s(S)|$

Note that the book abusses notation a bit and writes that last bullet as $|Q| = |R| \times |S|$

Cartesian Product Example

shelter

```
+----+---------+
| id | name    |
+----+---------+
|  1 | Howell  |
|  2 | Mansell |
+----+---------+
```

pet

```
+------------+----+--------+-----------+
| shelter_id | id | name   | breed     |
+------------+----+--------+-----------+
|          1 |  1 | Chloe  | Mix       |
|          1 |  2 | Dante  | GSD       |
|          1 |  3 | Heidi  | Dachshund |
|          2 |  1 | Bailey | Mix       |
|          2 |  2 | Sophie | Lab       |
|          2 |  3 | Heidi  | Dachshund |
+------------+----+--------+-----------+
```

Cross Product Example

```
+-----+---------+------------+-----+--------+-----------+
| sid | sname   | shelter_id | pid | pname  | breed     |
+-----+---------+------------+-----+--------+-----------+
|   1 | Howell  |          1 |   1 | Chloe  | Mix       |
|   2 | Mansell |          1 |   1 | Chloe  | Mix       |
|   1 | Howell  |          1 |   2 | Dante  | GSD       |
```

2	Mansell	1	2	Dante	GSD
1	Howell	1	3	Heidi	Dachshund
2	Mansell	1	3	Heidi	Dachshund
1	Howell	2	1	Bailey	Mix
2	Mansell	2	1	Bailey	Mix
1	Howell	2	2	Sophie	Lab
2	Mansell	2	2	Sophie	Lab
1	Howell	2	3	Heidi	Dachshund
2	Mansell	2	3	Heidi	Dachshund

Note that we've also done a `RENAME` to disambiguate `names` and `ids`:

$\rho_{(sid,sname,shelter\_id,pid,pname,breed)}(shelter \times pet)$

Cross Product and Select

Cross product meaningful when combined with `SELECT`.

$\sigma_{sid=shelter\_id}(\rho_{(sid,sname,shelter\_id,pid,pname,breed)}(shelter \times pet))$

sid	sname	shelter_id	pid	pname	breed
1	Howell	1	1	Chloe	Mix
1	Howell	1	2	Dante	GSD
1	Howell	1	3	Heidi	Dachshund
2	Mansell	2	1	Bailey	Mix
2	Mansell	2	2	Sophie	Lab
2	Mansell	2	3	Heidi	Dachshund

$CROSSED \leftarrow shelter \times pet$

$RENAMED \leftarrow \rho_{(sid,sname,shelter\_id,pid,pname,breed)}(CROSSED)$

$RESULT \leftarrow \sigma_{sid=shelter\_id}(RENAMED)$

Join

`JOIN` is a `CARTESIAN PRODUCT` followed by `SELECT`

$R \bowtie_{<joincondition>} S$

Where

- R and S are relations
- $< joincondition >$ is a boolean condition on values of tuples from R and S

$R \quad \bowtie_{<joincondition>} \quad S$ returns the tuples in $R \times S$ that satisfy the $< joincondition >$

Join Conditions

$< joincondition >$ is of the form $A_i \theta B_j$

- A_i is an attribute of R, B_j is an attribute of S
- $dom(A_i) = dom(B_j)$

A $< joincondition >$ can be a conjunction of simple conditions, e.g.:

$< c_1 > AND < c_2 > ...AND < c_n >$

Join Example

worker

```
+----+--------+---------------+------------+
| id | name   | supervisor_id | shelter_id |
+----+--------+---------------+------------+
|  1 | Tom    |          NULL |          1 |
|  2 | Jie    |             1 |          1 |
|  3 | Ravi   |             2 |          1 |
|  4 | Alice  |             2 |          1 |
|  5 | Aparna |          NULL |          2 |
|  6 | Bob    |             5 |          2 |
|  7 | Xaoxi  |             6 |          2 |
|  8 | Rohan  |             6 |          2 |
+----+--------+---------------+------------+
```

shelter

```
+----+---------+
| id | name    |
+----+---------+
|  1 | Howell  |
|  2 | Mansell |
+----+---------+
```

Join Example

$worker \bowtie_{shelter\_id=sid} \rho_{(sid,sname)}(shelter)$

id	name	supervisor_id	shelter_id	sid	sname
1	Tom	NULL	1	1	Howell
2	Jie	1	1	1	Howell
3	Ravi	2	1	1	Howell
4	Alice	2	1	1	Howell
5	Aparna	NULL	2	2	Mansell
6	Bob	5	2	2	Mansell
7	Xaoxi	6	2	2	Mansell
8	Rohan	6	2	2	Mansell

Notice that we had to use renaming of attributes in *shelter*.

A join operation in which the comparison operator θ is $=$ is called an *equijoin*.

Natural Join

Notice that the `shelter_id` attribute was repeated in the previous equijoin result. A `NATURAL JOIN` is a equijoin in which the redundant attribute has been removed.

$R * S$

Where

- R and S have an attribute with the same name and same domain which is automatically chosen as the equijoin attribute

Natural Join Example

Recall the first join example. If we rename the `id` attribute to `shelter_id` we can use a natural join:

$\rho_{(shelter\_id, sname)}(shelter) * worker$

shelter_id	sname	id	name	supervisor_id
1	Howell	1	Tom	NULL
1	Howell	2	Jie	1
1	Howell	3	Ravi	2
1	Howell	4	Alice	2
2	Mansell	5	Aparna	NULL
2	Mansell	6	Bob	5
2	Mansell	7	Xaoxi	6

```
|          2 | Mansell |  8 | Rohan  |             6 |
+------------+---------+----+--------+---------------+
```

Outer Joins

The joins we've discussed so far have been inner joins. Result relations of inner joins include only tuples from the joined tables that match the join condition.

Outer join results inlude tuples that matched, and tuples that didn't match the join condition.

Left Outer Join

$R_{<joincondition>}S$

Where

- R and S are relations
- $< joincondition >$ is a boolean condition on values of tuples from R and S

$R_{<joincondition>}S$ returns the tuples in $R \times S$ that satisfy the $< joincondition >$ as well as the tuples from R that don't match the join condition. In the result relation the unmatched tuples from R are null-padded to give them the correct degree in the result.

Left Outer Join Example

author

```
+-----------+------------+-----------+
| author_id | first_name | last_name |
+-----------+------------+-----------+
|         1 | John       | McCarthy  |
|         2 | Dennis     | Ritchie   |
|         3 | Ken        | Thompson  |
|         4 | Claude     | Shannon   |
|         5 | Alan       | Turing    |
|         6 | Alonzo     | Church    |
|         7 | Perry      | White     |
|         8 | Moshe      | Vardi     |
|         9 | Roy        | Batty     |
+-----------+------------+-----------+
```

book

book_id	book_title	month	year	editor
1	CACM	April	1960	8
2	CACM	July	1974	8
3	BST	July	1948	2
4	LMS	November	1936	7
5	Mind	October	1950	NULL
6	AMS	Month	1941	NULL
7	AAAI	July	2012	9
8	NIPS	July	2012	9

Return all the authors. For all the authors who are editors, show their books.

Authors and Edited Books

Show all the authors. For all the authors who are editors, show their books.

$R_{author_{i}d=editor}S$

author_id	first_name	last_name	book_id	book_title	month	year	editor
8	Moshe	Vardi	1	CACM	April	1960	8
8	Moshe	Vardi	2	CACM	July	1974	8
2	Dennis	Ritchie	3	BST	July	1948	2
7	Perry	White	4	LMS	November	1936	7
9	Roy	Batty	7	AAAI	July	2012	9
9	Roy	Batty	8	NIPS	July	2012	9
1	John	McCarthy	NULL	NULL	NULL	NULL	NULL
3	Ken	Thompson	NULL	NULL	NULL	NULL	NULL
4	Claude	Shannon	NULL	NULL	NULL	NULL	NULL
5	Alan	Turing	NULL	NULL	NULL	NULL	NULL
6	Alonzo	Church	NULL	NULL	NULL	NULL	NULL

Notice how attribute values are padded to the right in a left outer join.

Right Outer Join

$R_{<joincondition>}S$

Where

- R and S are relations

- $< joincondition >$ is a boolean condition on values of tuples from R and S

$R_{<joincondition>}S$ returns the tuples in $R \times S$ that satisfy the $< joincondition >$ as well as the tuples from S that don't match the join condition. In the result relation the unmatched tuples from S are null-padded to give them the correct degree in the result.

Right Outer Join Example

Show all the books. For books with editors, show their editors.

$R_{author_id=editor}S$

author_id	first_name	last_name	book_id	book_title	month	year	editor
8	Moshe	Vardi	1	CACM	April	1960	8
8	Moshe	Vardi	2	CACM	July	1974	8
2	Dennis	Ritchie	3	BST	July	1948	2
7	Perry	White	4	LMS	November	1936	7
NULL	NULL	NULL	5	Mind	October	1950	NULL
NULL	NULL	NULL	6	AMS	Month	1941	NULL
9	Roy	Batty	7	AAAI	July	2012	9
9	Roy	Batty	8	NIPS	July	2012	9

Notice how attribute values are padded to the left in a right outer join.

Full Outer Join

$R_{<joincondition>}S$

Where

- R and S are relations
- $< joincondition >$ is a boolean condition on values of tuples from R and S

$R < joincondition > S$ returns the tuples in $R \times S$ that satisfy the $< joincondition >$ as well as the tuples from both R and S that don't match the join condition. In the result relation the unmatched tuples are null-padded to give them the correct degree in the result.

Full Outer Join Example

Show all authors and books, matching editors with their books.

$R_{author_id=editor} S$

author_id	first_name	last_name	book_id	book_title	month	year	editor
8	Moshe	Vardi	1	CACM	April	1960	8
8	Moshe	Vardi	2	CACM	July	1974	8
2	Dennis	Ritchie	3	BST	July	1948	2
7	Perry	White	4	LMS	November	1936	7
9	Roy	Batty	7	AAAI	July	2012	9
9	Roy	Batty	8	NIPS	July	2012	9
1	John	McCarthy	NULL	NULL	NULL	NULL	NULL
3	Ken	Thompson	NULL	NULL	NULL	NULL	NULL
4	Claude	Shannon	NULL	NULL	NULL	NULL	NULL
5	Alan	Turing	NULL	NULL	NULL	NULL	NULL
6	Alonzo	Church	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	5	Mind	October	1950	NULL
NULL	NULL	NULL	6	AMS	Month	1941	NULL

Review Question 1

Given the $r(book)$:

book_id	book_title	month	year	editor
1	CACM	April	1960	8
2	CACM	July	1974	8
3	BST	July	1948	2
4	LMS	November	1936	7
5	Mind	October	1950	7
6	AMS	Month	1941	7
7	AAAI	July	2012	9
8	NIPS	July	2012	9

How many tuples are in $\pi_{book\_title}(book)$? (Notice I've abused notation here, since that's how you'll see it in the book and on the test.)

Review Question 1 Answer

7:

```
+------------+
| book_title |
+------------+
| CACM       |
| BST        |
| LMS        |
| Mind       |
| AMS        |
| AAAI       |
| NIPS       |
+------------+
```

The *book_title* appears twice in *book* and the result of a relational algebra expression is a set.

Review Question 2

Given the relation $r(book)$:

```
+---------+------------+----------+------+--------+
| book_id | book_title | month    | year | editor |
+---------+------------+----------+------+--------+
|       1 | CACM       | April    | 1960 |      8 |
|       2 | CACM       | July     | 1974 |      8 |
|       3 | BST        | July     | 1948 |      2 |
|       4 | LMS        | November | 1936 |      7 |
|       5 | Mind       | October  | 1950 |      7 |
|       6 | AMS        | Month    | 1941 |      7 |
|       7 | AAAI       | July     | 2012 |      9 |
|       8 | NIPS       | July     | 2012 |      9 |
+---------+------------+----------+------+--------+
```

Which books were published before 1960 or after 2000?

Review Question 2

Which books were published before 1960 or after 2000?

$\sigma_{year<1960}(book) \cup \sigma_{year>2000}(book)$

```
+---------+------------+----------+------+--------+
| book_id | book_title | month    | year | editor |
```

3	BST	July	1948	2
4	LMS	November	1936	7
5	Mind	October	1950	7
6	AMS	Month	1941	7
7	AAAI	July	2012	9
8	NIPS	July	2012	9

Review Question 3

Given:

worker

id	name	supervisor_id	shelter_id
1	Tom	NULL	1
2	Jie	1	1
3	Ravi	2	1
4	Alice	2	1
5	Aparna	NULL	2
6	Bob	5	2
7	Xaoxi	6	2
8	Rohan	6	2

shelter

id	name
1	Howell
2	Mansell

How would we find the names of all the workers who work at Mansell?

Review Question 3 Answer

How would we find all the workers who work at Mansell?

$SHELTERS \leftarrow \rho_{sid,sname}(shelter)$

$MANSELLERS \leftarrow$

$$worker \bowtie_{shelter\_id=sid \land sname='Mansell'} (SHELTERS)$$

Gives:

id	name	supervisor_id	shelter_id	sid	sname
5	Aparna	NULL	2	2	Mansell
6	Bob	5	2	2	Mansell
7	Xaoxi	6	2	2	Mansell
8	Rohan	6	2	2	Mansell

then …

Review Question 3 Answer

$\pi_{name}(MANSELLERS)$

gives:

name
Aparna
Bob
Xaoxi
Rohan

Full inline expression:

$\pi_{name}(worker \bowtie_{shelter\_id=sid \land sname='Mansell'} \rho_{(sid,sname)}(shelter))$