CS3DB3/SE4DB3/SE6DB3 TUTORIAL

Xiao Jiao Wang

Feb 25, 2015

Relational Algebra

- IMPORTANT: relational engines work on **bags**, no set !!!

Union, intersection, and difference

- Union: ∪ Intersection: ∩ Difference: –
- **Note**: Both operands must have the same relation schema.
- Example 1

R:

Product Name	Unit price
Melon	800G
Strawberry	150G
Melon	800G

S:

Product Name	Unit price
Melon	800G
Strawberry	150G
Apple	120G

R—S

Product Name	Unit price
Melon	800G

Selection and projection

- Selection: σ_c (R)
  - Picking all tuples of R that satisfy C.
  - C is a condition that refers to attributes of R.
- Projection: $\pi_L(R)$
  - L is a list of attributes from the schema of R.
  - Constructed by looking at each tuple of R.
  - Schema of result contains exactly the fields in the projection list, with the same names that they had in relation R.
- Example 2

R:

Sname	Rating
Yuppy	9
Lubber	8
Guppy	5
Rusty	10

$\pi_{name,rating}(\sigma_{rating>8}\ (R))$

Sname	Rating
Yuppy	9
Rusty	10

Don't forget schema

Renaming and Product

- Products and joins: compositions of relations.
- Renaming : $\rho_{R1(A1,\ \ldots,\ An)}$ (R2)
  - Gives a new schema to a relation.
  - Makes R1 be a relation with attributes A1, … , An and the same tuples as R2.
- Product: R3:=R1 × R2
  - Also called cross-product or Cartesian product.
  - Pair each tuple t1 of R1 with each tuple t2 of R2 and concatenation t1 t2 is a tuple of R3.
    - # of tuples in R3 = (# of tuples in R1)×(# of tuples in R2)
  - Schema of R3 is the attributes of R1 and then R2, in order.
  - **Beware:** R1 and R2 have the common attribute A
    - In relational algebra, use renaming to distinguish.

Renaming and Product (Cont.)

- Example 3

R1
(2 tuples)

Name	Price
Melon	800G
Apple	120G

R2
(3 tuples)

Fruit	Place
Melon	Canada
Lemon	Spain
Apple	France

R1 × R2
(2*3=6 tuples)

Name	Price	Fruit	Place
Melon	800G	Melon	Canada
Melon	800G	Lemon	Spain
Melon	800G	Apple	France
Apple	120G	Melon	Canada
Apple	120G	Lemon	Spain
Apple	120G	Apple	France

Schema:
the attributes of R1 and then R2, in order.

R1 and R2 have no common attributes.

Renaming and Product (Cont.)

- Example 4

R1
(2 tuples)

Name	Price
Melon	800G
Apple	120G

R2
(3 tuples)

Name	Place
Melon	Canada
Lemon	Spain
Apple	France

R1 × R2
(2*3=6 tuples)

R1.Name	Price	R2.Name	Place
Melon	800G	Melon	Canada
Melon	800G	Lemon	Spain
Melon	800G	Apple	France
Apple	120G	Melon	Canada
Apple	120G	Lemon	Spain
Apple	120G	Apple	France

R1 and R2 have a common attribute.

Theta-Join

- Theta-Join: $R3 := R1 \bowtie_c R2$
  - Take the product R1 × R2.
  - Then apply σ_c to the result.
- Example 5

R1

Name	Price
Melon	800G
Apple	120G

R2

Fruit	Place
Melon	Canada
Lemon	Spain
Apple	France

$R1 \bowtie_{Name=Fruit} R2$

Name	Price	Fruit	Place
Melon	800G	Melon	Canada
Apple	120G	Apple	France

Theta-Join (Cont.)

- Example 5

R1

Name	Price
Melon	800G
Apple	120G

R2

Name	Place
Melon	Canada
Lemon	Spain
Apple	France

$$R1 \bowtie_{R1.Name=R2.Name} R2$$

R1.Name	Price	R2.Name	Place
Melon	800G	Melon	Canada
Apple	120G	Apple	France

There are **4** columns.

Natural Join

- Natural Join: R3:=R1 ⋈ R2
  - Connects two relations by:
    - Equating attributes of the same name, and
    - Projecting out one copy of each pair of equated attributes.
- Example 6

R1

Name	Price
Melon	800G
Apple	120G

R2

Name	Place
Melon	Canada
Lemon	Spain
Apple	France

R1 ⋈ R2

Name	Price	Place
Melon	800G	Canada
Apple	120G	France

There are **3** columns.

Precedence of relational operators

- [σ, π, ρ] (highest)
- [$\times$, $\bowtie$]
- $\cap$
- [$\cup$, $-$]

Duplicate Elimination and Sorting

- Duplicate elimination: $\delta(\mathrm{R})$
  - **Recall**: relational engines work on bags.
  - Consists of one copy of each tuple that appears in R2 one or more times.
  - SQL: SELECT DISTINCT …
- Sorting: $\tau_{\mathrm{L}}(\mathrm{R})$
  - L is a list of some of the attributes of R2.
  - Sorted first on the value of the first attribute on L, then on the second attributes of L, and so on.

Grouping and Aggregation

- Grouping and Aggregation : $\gamma_L(R)$
  - L is a list of elements that are either
    - Grouping attributes
    - AGG(A), where AGG is one of the aggregation operators such as **SUM, AVG,COUNT, MIN, MAX** and **A** is an attribute.
      - An arrow and a new attribute name renames the component
  - **Example:** $\gamma_{A,B,AVG(C)\rightarrow X}(R)$

R:

A	B	C
1	2	3
4	5	6
1	2	5

Group R according to all the grouping attributes on list L

Group R by A and B

A	B	C
1	2	3
1	2	5
4	5	6

Within each group, compute AGG(A) for each aggregation on list L

Result has one tuple for each group

A	B	X
1	2	4
4	5	6

SQL and relational algebra

- **SELECT** A1, A2, …, An
  FROM R1, R2, …, Rm
  WHERE P

  is equivalent to the multiset relational algebra expression

Don’t forget the parenthesis since σ has a higher Precedence than [×, ⋈]

$$\prod_{A1, A2, \cdots, An} (\sigma_P (R1 \times R2 \times \cdots \times Rm))$$

SQL and relational algebra (Cont.)

- Example 1 **Takes** (id, course_id, semester, year, grade)

  Teaches(name, course_id, semester, year)

  - Find the IDs of all courses who were taught by an instructor named Jones.
  - SQL
    - **SELECT** Teaches.course_id

      FROM Takes, Teaches

      WHERE name = 'Jones' AND Takes.course_id = Teaches.course_id;
  - Relation algebra
    - WAY 1: $\Pi_{course\_id}(\sigma_{name='Jones'}\ (Takes \bowtie Teaches))$

      Because of common attribute
    - WAY 2: $\Pi_{Teaches.course\_id}(\sigma_{name='Jones'}\ (Takes \bowtie_{Takes.course\_id = Teaches.course\_id} Teaches))$
    - WAY 3: $\Pi_{Teaches.course\_id}(\sigma_{name='Jones'\ \wedge\ Takes.course\_id = Teaches.course\_id}\ (Takes \times Teaches))$

SQL and relational algebra (Cont.)

- Example 2
  - **Works** (<u>pname</u>, cname, salary)
  - Find the names of all employees who earn more than every employee of “First Bank”.
  - SQL

```
SELECT pname
FROM Works
WHERE salary >ALL (SELECT salary
                   FROM Works
                   WHERE cname= 'First Bank');
```

- Relational algebra

$$R1 := \Pi_{w1.pname}(\rho_{w1}(Works) \bowtie_{w1.salary \leqslant w2.salary \wedge w2.cname=\text{‘First Band’}} \rho_{w2}(Works))$$

$$Result := \Pi_{pname}(Works) - R1$$

Assignment:
create temporary relation names

SQL and relational algebra (Cont.)

- **SELECT** A1, A2, **AGG**(A3) AS AGG3
  FROM R1, R2,…, Rm
  WHERE P
  GROUP BY A1, A2
  - Is equivalent to the multiset relational algebra expression

$$\gamma_{A1,A2,AGG(A3)\rightarrow AGG3}(\sigma_P(R1\times R2\times\ldots\times Rm))$$

- If only display attribute A1 and AGG3, then

$$\Pi_{A1,AGG3}(\gamma_{A1,A2,AGG(A3)\rightarrow AGG3}(\sigma_P(R1\times R2\times\ldots\times Rm)))$$

SQL and relational algebra (Cont.)

- Example 3
  - **Takes (<u>student_id, course_id, semester, year</u>, grade)**
  - Find the enrollment of each course that was offered in Fall 2009.
  - SQL

```
SELECT course_id, count(*) as enrollment
FROM Takes
WHERE year=2009 AND semester='Fall'
GROUP BY course_id;
```

  - Relational Algebra

$$\gamma_{\text{course\_id, count(*)}\rightarrow\text{enrollment}}\left(\sigma_{\text{year}=2009\ \wedge\ \text{semester}=\text{“Fall”}}(\text{Takes})\right)$$

SQL and relational algebra (Cont.)

- Example 4
  - **Takes (<u>student_id, course_id, semester, year</u>, grade)**
  - Find the maximum enrollment in Fall 2009.
  - SQL

```
SELECT MAX(enrollment)
FROM (SELECT course_id, count(*) as enrollment
          FROM Takes
          WHERE year=2009 AND semester='Fall'
          GROUP BY course_id);
```

  - Relational Algebra

$$R := \gamma_{\text{course\_id, count(*)} \rightarrow \text{enrollment}} \left(\sigma_{\text{year}=2009 \wedge \text{semester}=\text{“Fall”}} (\text{Takes}) \right)$$

$$\text{Result} := \gamma_{\max(\text{enrollment})}(R)$$