Relational Algebra

CS 564- Fall 2016

ACKs: *Dan Suciu, Jignesh Patel, AnHai Doan*

Relational Query Languages

- allow the *manipulation* and *retrieval* of data from a database
- two types of query languages:
  - Declarative: describe what a user wants, rather than how to compute it
    - Tuple Relational Calculus (**TRC**)
    - Domain Relational Calculus (**DRC**)
  - Procedural: operational, useful for representing execution plans
    - Relational Algebra (**RA**)

QUERY VS PROGRAMMING LANGUAGES

Relational Query Languages:

- are not "Turing complete", which means they cannot capture any type of computation
- are not intended to be used for complex calculations
- support easy, efficient access to large datasets

What is Relational Algebra?

- **algebra**: mathematical system consisting of
  - operands: variables or values from which new values can be constructed
  - operators: symbols denoting procedures that construct new values from given values
- **relational algebra**: an algebra whose operands are relations or variables that represent relations
  - operators do the most common things that we need to do with relations in a database
  - can be used as a query language for relations

Relational Algebra Prelim

- **Query**:
  - Input: relational instances
  - Output: relational instances
  - specified using the schemas
    - may produce different results for different instances
    - the schema of the result is fixed
- there are two types of notation for attributes:
  - positional (e.g. 2, 4)
  - named-field (e.g. C.name, Person.SSN)

Relational Algebra Prelim

- Basic operations:
  - *Selection* {σ}: selects a subset of rows
  - *Projection* {π}: deletes columns
  - *Cross-product* {$\times$}: combines two relations
  - *Set-difference* {$-$}
  - *Union* {$\cup$}
- When the relations have named fields:
  - *Renaming* {ρ}
- Additional operations:
  - *Intersection, join, division*

BASIC OPERATIONS

SELECTION

Notation: $\sigma_C(R)$

- C is a condition that refers to the attributes of R
- outputs the rows of R that satisfy C
- output schema: same as input schema

Example

- $\sigma_{age>24}(Person)$
- $\sigma_{age>24\ and\ age\leq 28}(Person)$
- $\sigma_{age>24\ and\ name=\text{“}Paris\text{”}}(Person)$

Selection Example

Person

SSN	name	age	phoneNumber
934729837	Paris	24	608-374-8422
934729837	Paris	24	603-534-8399
123123645	John	30	608-321-1163
384475687	Arun	25	206-473-8221

$\sigma_{age>24}(Person)$

SSN	name	age	phoneNumber
123123645	John	30	608-321-1163
384475687	Arun	25	206-473-8221

PROJECTION

Notation: $\pi_{A_1,A_2,\ldots,A_n}(R)$

- outputs only the columns $A_1, A_2, \ldots, A_n$
- removes any duplicate tuples (why?)
- output schema: $R(A_1, A_2, \ldots, A_n)$

Example

- $\pi_{SSN,age}(Person)$
- $\pi_{SSN,phoneNumber,age}(Person)$

Projection Example

Person

SSN	name	age	phoneNumber
934729837	Paris	24	608-374-8422
934729837	Paris	24	603-534-8399
123123645	John	30	608-321-1163
384475687	Arun	20	206-473-8221

$\pi_{SSN,name}(Person)$

SSN	name
934729837	Paris
123123645	John
384475687	Arun

UNION

Notation: $R_1 \cup R_2$

- outputs all tuples in R_1 or R_2
- both relations must have the same schema!
- output schema: same as input

A	B
a_1	b_1
a_2	b_1
a_2	b_2

$\cup$

A	B
a_1	b_1
a_3	b_1
a_4	b_4

$=$

A	B
a_1	b_1
a_2	b_1
a_2	b_2
a_3	b_1
a_4	b_4

DIFFERENCE

Notation: $R_1 - R_2$

- outputs all tuples in R_1 and not in R_2
- both relations must have the same schema!
- output schema: same as input

A	B
a_1	b_1
a_2	b_1
a_2	b_2

$-$

A	B
a_1	b_1
a_3	b_1
a_4	b_4

$=$

A	B
a_2	b_1
a_2	b_2

CROSS-PRODUCT

Notation: $R_1 \times R_2$

- matches each tuples in R_1 with each tuple in R_2
- input schema: $R_1(A_1, A_2, \ldots, A_n)$, $R_2(B_1, B_2, \ldots, B_m)$
- output schema: $R(A_1, \ldots, A_n, B_1, \ldots, B_m)$

CROSS-PRODUCT EXAMPLE

Person

SSN	name
934729837	Paris
123123645	John

Dependent

depSSN	depname
934729837	Helen
934729837	Bob

$$Person \times Dependent$$

SSN	name	depSSN	depname
934729837	Paris	934729837	Helen
123123645	John	934729837	Bob
934729837	Paris	934729837	Bob
123123645	John	934729837	Helen

RENAMING

Notation: $\rho_{A_1,A_2,\ldots,A_n}(R)$

- does not change the instance, only the schema!
- input schema: $R(B_1, B_2, \ldots, B_n)$
- output schema: $R(A_1, \ldots, A_n)$

Why is it necessary?

RENAMING EXAMPLE

Person

SSN	name
934729837	Paris
123123645	John

Dependent

SSN	name
934729837	Helen
934729837	Bob

$$Person \times \rho_{depSSN,depname}\,(Dependent)$$

SSN	name	depSSN	depname
934729837	Paris	934729837	Helen
123123645	John	934729837	Bob
934729837	Paris	934729837	Bob
123123645	John	934729837	Helen

Derived Operations

INTERSECTION

Notation: $R_1 \cap R_2$

- outputs all tuples in R_1 and R_2
- output schema: same as input
- can be expressed as: $R_1 - (R_1 - R_2)$

A	B
a_1	b_1
a_2	b_1
a_2	b_2

$\cap$

A	B
a_1	b_1
a_3	b_1
a_4	b_4

$=$

A	B
a_1	b_1

JOIN (THETA JOIN)

Notation: $R_1 \bowtie_C R_2 = \sigma_C(R_1 \times R_2)$

- cross-product followed by a selection
- C can be any boolean-valued condition
- might have less tuples than the cross-product!

THETA JOIN EXAMPLE

Person

SSN	name	age
934729837	Paris	26
123123645	John	22

Dependent

dSSN	dname	dage
934729837	Helen	23
934729837	Bob	28

$$Person \bowtie_{Person.age>Dependent.dage} Dependent$$

SSN	name	age	dSSN	dname	dage
934729837	Paris	26	934729837	Helen	23

EQUI-JOIN

Notation: $R_1 \bowtie_C R_2$

- special case of join where the condition C contains only equalities between attributes!
- output schema: same as the cross-product

Example for $R(A, B), S(C, D)$

- $R \bowtie_{B=C} S$
- output schema: $T(A, B, C, D)$

NATURAL JOIN

Notation: $R_1 \bowtie R_2$

- equi-join on all the common fields
- the output schema has one copy of each common attribute

Person

SSN	name	age
934729837	Paris	26
123123645	John	22

Dependent

SSN	dname
934729837	Helen
934729837	Bob

$Person \bowtie Dependent$

SSN	name	age	dname
934729837	Paris	26	Helen
934729837	Paris	26	Bob

NATURAL JOIN

Natural Join $R \bowtie S$

- Input schema: $R(A, B, C, D), S(A, C, E)$
  - Output schema: *T(A, B, C, D, E)*
- Input schema: $R(A, B, C), S(D, E)$
  - Output schema: *T(A, B, C, D, E)*
- Input schema: $R(A, B, C), S(A, B, C)$
  - Output schema? *T(A, B, C,)*

SEMI-JOIN

Notation: $R_1 \ltimes R_2$

- natural join followed by projection on the attributes of R_1

Example:

- $R(A, B, C), S(B, D)$
- $R \ltimes S = \pi_{A,B,C}(R \bowtie S)$
- output schema: *T(A, B, C)*

DIVISION

Notation: R_1 / R_2

- suppose $R_1(A, B)$ and $R_2(B)$
- the output contains all values **a** such that for every tuple (**b**) in R_2, tuple **(a, b)** is in R_1
- output schema: *R(A)*

Division Example

A

A	B
a_1	b_1
a_1	b_2
a_1	b_3
a_2	b_1

B_1

B
b_2
b_3
b_1

B_2

B
b_1

A / B_1

A
a_1

A / B_2

A
a_1
a_2

Extending Relational Algebra

GROUP BY AGGREGATE

- is part of the so-called extended RA
- helps us to compute counts, sums, min, max, …

Examples

- What is the average age of the customers?
- How many people bought an iPad?

GROUP BY AGGREGATE

Notation: $\gamma_{X,Agg(Y)}(R)$

- group by the attributes in X
- aggregate the attribute in Y
  - SUM, COUNT, AVG (average), MIN, MAX

- Output schema: X + an extra (numerical) attribute

EXAMPLE

Person

SSN	name	age
934729837	Paris	24
123123645	John	30
384475687	Arun	21

$$\gamma_{AVG(age)}(Person)$$

AVG(age)
25

EXAMPLE

Person

SSN	name	age	phoneNumber
934729837	Paris	24	608-374-8422
934729837	Paris	24	603-534-8399
123123645	John	30	608-321-1163
384475687	Arun	21	206-473-8221

$$\gamma_{SSN,COUNT(phoneNumber)}(Person)$$

SSN	COUNT(phoneNumber)
934729837	2
123123645	1
384475687	1

CONSTRUCTING RA QUERIES

COMBINING RA OPERATORS

- We can build more complex queries by combining RA operators together

  e.g. standard algebra: $(x + 1) * y - z^2$

- There are 3 different notations:
  - sequence of assignment statements
  - expressions with operators
  - expression trees

COMBINING RA OPERATORS

Input schema: $R(B, C), S(A, B)$

- expressions with operators

  $\pi_A(\sigma_{C=1}(R) \bowtie S)$

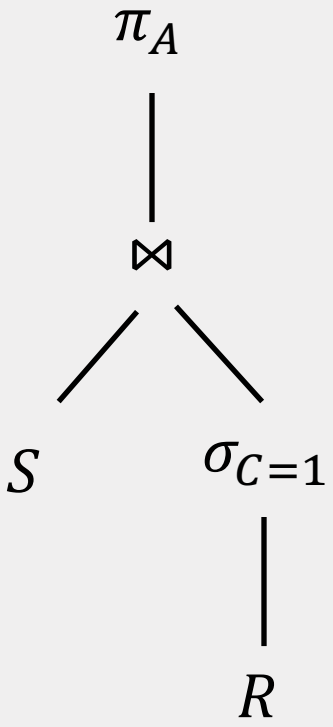
- sequence of assignment statements

  $R' = \sigma_{C=1}(R)$

  $R'' = R' \bowtie S$

  $R''' = \pi_A(R'')$

- expression trees



Expressive Power

- RA cannot express transitive closure!

Edges

From	To
a	b
b	c
a	d
c	d

Transitive closure computes all pairs of nodes connected by a directed path

EXAMPLES

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

Q1: *What are the names of the sailors who have reserved boat with name "100" ?*

EXAMPLES

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

Q2: *What are the names of the sailors who have reserved a red boat ?*

EXAMPLES

Sailors (<u>sid</u>, sname, rating, age)
Reserves (<u>sid</u>, <u>bid</u>, <u>day</u>)
Boats (<u>bid</u>, bname, color)

Q3: *What are the names of the sailors who have reserved a green* ***<u>or</u>*** *red boat ?*

EXAMPLES

Sailors (<u>sid</u>, sname, rating, age)
Reserves (<u>sid</u>, <u>bid</u>, <u>day</u>)
Boats (<u>bid</u>, bname, color)

Q4: *What are the names of the sailors who have reserved a green* ***<u>and</u>*** *a red boat ?*

EXAMPLES

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

Q5: *Find the names of the sailors who have reserved all boats with name "470".*

EXAMPLES

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, country)
Person (ssn, name, phone, city)

Q6: *Find the phone numbers of the people who have bought iPads from Fred (the salesman).*

EXAMPLES

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, country)
Person (ssn, name, phone, city)

Q7: *Find the names and phone numbers of the people who have bought products from the USA.*

EXAMPLES

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, country)
Person (ssn, name, phone, city)

Q8: *Find the names of the people who have bought products* ***only*** *from the USA.*

EXAMPLES

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, country)
Person (ssn, name, phone, city)

Q9: *Find the names of the people who have bought products from the USA but not from Greece.*

EXAMPLES

Product (pid, name, price, category, maker-cid)
Purchase (buyer-ssn, seller-ssn, store, pid)
Company (cid, name, country)
Person (ssn, name, phone, city)

Q10: *Find the names of the people who have bought products from the USA and live in Madison.*

EXAMPLES

City (cid, name, population)
Flight (fid, length, start-city, end-city, aid)
Airline (aid, name, profit)

Q11: *Find the flight ids for flights that start in a city with id "MSN" and end in a city with id "LON".*

EXAMPLES

City (cid, name, population)
Flight (fid, length, start-city, end-city, aid)
Airline (aid, name, profit)

Q12: *Find the names of the cities that have a flight for every airline with profit more than 0.*

RECAP

- Relational Algebra
  - query language for relations
- Basic Operations
  - selection, projection
  - difference, union
  - cross-product, renaming
- Derived Operations
  - join, natural join, equi-join, division, etc