Relational Algebra (RA)

Week 6/7 – CSC 343

Michael Liut

Department of Mathematical and Computational Sciences
University of Toronto Mississauga

October 23 and November 6, 2019

After today, you be able to:

1. Understand the differences between the core relational algebra operands and operators.
2. Understand the rules of precedence.
3. Differentiate between bag and set semantics.
4. Utilize expression trees to help formulate your RA solution.
5. Translate SQL to RA and RA to SQL.

Relational Query Languages

Query Languages allow manipulation and retrieval of data from a database.

- The relational model supports simple and powerful Query Languages, those which:
  1. have a formal foundation built on logic; and
  2. allow optimization.
- **Query Languages != Programming Languages**
  - Query Languages are not intended for complex calculations.
  - Query Languages support easy and efficient access to large data sets.

Formal Relational Query Languages

There are two mathematical Query Languages that form the basis of SQL and for its implementation:

Relational Algebra is more operational. It is useful for representing execution plans.

Relational Calculus is non-operational, it is declarative. It allows users to describe what they want, rather than how to compute it.

What is Algebra?

In general, it is a mathematical structure that is defined by the author. For our purposes, it is a system that consists of:

Operands – variables or values from which new values can be constructed.

Operators – symbols which denote procedures that construct new values from inputted/given values.

What is Relational Algebra?

It is an algebra whose operands are relations or variables that represent relations.

- The operators are designed to perform the most common operations that users need to do with relations in a database.
  - The result is an algebra that can be used as a Query Language for relations.

What is Relational Algebra?

- Operands: tables (relations)
- Operators:
  - Regular set operators (union, intersection, difference)
  - Choose only rows you want (selection)
  - Choose only columns you want (projection)
  - Combine tables (join)
  - ... and more!

Base Operators

- σ is Selection
  - Selection is used to specify a certain row.
  - $\sigma_c(R)$ where c is a list of conditions involving the attribute(s) R.
- π is Projection
  - Projection is used to specify a certain column.
  - $\pi_\ell(R)$ where ℓ is a list of attributes involving the attribute(s) R.
- x is Cartesian Product
  - Cartesian Product is the combination of two (or more) relations.
  - $R := R_1 \; x \; R_2$ where the tuples of R_1 and tuples R_2 are paired together to form R.

Base Operators

- $\cup$ is Union
  - Union is used to combine two relations into one.
  - Suppose a tuple t appears in R_1 m times, and in R_2 n times. Then in the union, t appears $m + n$ times.
- $\cap$ is Intersection
  - Intersection is used to see what overlaps in two relations.
  - Suppose a tuple t appears in R_1 m times, and in R_2 n times. Then in the union, t appears $min(m, n)$ times.
- $-$ is Difference
  - Difference is basically subtraction, it is used to see what is left over in R_1 when $R_1 - R_2$.
  - Suppose a tuple t appears in R_1 m times, and in R_2 n times. Then in the union, t appears $max(0, m - n)$ times.

defined for union compatible relations

Extended Relational Algebra

$\bowtie$ is Natural Join

- Connects (aka joins) two relations by equating attributes of the same name and by projecting out one copy of each pair of equated attributes.

$\bowtie_c$ is Theta Join

- Where the condition c is:
  1. $a\ \theta\ b$, where a and b are attribute names; or
  2. $a\ \theta\ x$, where a is an attribute name and x is value.

  θ here represents a binary relational operator belonging to $\{\leq,\ <,\ =,\ >,\ \geq\}$.
- Connects (aka joins) two relations by equating attributes based on some condition

$\bowtie_c$ is Equijoin iff c's θ is the quality operator (i.e. $=$).

there are additional types of joins not included in here

Extended Relational Algebra

ρ does Renaming

- $R_1 := \rho_{R_1(A_1,\ldots,A_n)}(R_2)$, where R_1 becomes a new relation with the attributes $A_1, \ldots, A_n$ of R_2.
- $\rho_{a/b}(R)$, where b is an attribute of R and a is the renamed attribute name.

δ does Duplicate Elimination

- This is used for duplicate elimination in bag semantics.
- $R_1 := \delta(R_2)$ will result in R_1 containing one copy of each tuple that appears in R_2 one or more times.

Extended Relational Algebra

- τ is the Sorting of tuples
  - $R_1 := \tau_L(R_2)$, where L is a list of R_2's attributes.
  - R_1 is the sorted list of tuples of R_2, based on the order specified in list L. Ties are broken arbitrarily.
  - Items are sorted in ascending order by default. This means if you are attempting to sort in descending order, you will denote the list the prefix $-$. e.g. $R_1 := \tau_{-L}(R_2)$
  - **Fun Fact:** τ is the only operator whose result is neither a set nor a bag. $\rightarrow$ **Told you it would be fun! :-)**

Extended Relational Algebra

- γ is Grouping and Aggregation.
  - The aggregation operations are: AVG, MIN, MAX, SUM, and COUNT.
  - $R_1 := \gamma_L(R_2)$, where L is a list of individual grouping attributes or an aggregation operation of an attribute.

    note: An arrow to a new attribute name would rename it.
    (e.g. $AVG(salary) \rightarrow pay$)

RA Rules of Precedence

1. $[\ \sigma,\ \pi,\ \rho\ ]$ (highest)
2. $[\ x,\ \bowtie\ ]$
3. $[\ \cap\ ]$
4. $[\ \cup,\ -\ ]$ (lowest)

Note: you may combine operators with parentheses and precedence rules.

Bag vs. Set Semantics

A bag (aka multiset) allows the repetition of objects.

A set is a collection of distinct objects.

e.g. $\{1, 2, 1, 3\}$ is a bag.

e.g. $\{1, 2, 3\}$ is a bag and a set.

Bag vs. Set Semantics

- Real RDBMSs treat relations as bags of tuples.
- SQL is a bag language.
- Fun Fact: some operations, like projection, are more efficient on bags than sets.

Why bags?

- This is primarily due to performance. The elimination of duplicates is often computationally expensive, as it requires sorting.

Bag Laws ! = Set Laws

- Some algebraic laws that hold for sets also hold for bags, but not all of them do.

  e.g. the `commutative` law for $\cup$ does hold for bags.

  i.e. $R_1 \cup R_2 = R_2 \cup R_1$

  e.g. the `idempotent` law, in general, for $\cup$ does not hold for bags.

  i.e. $R_1 \cup R_1 \; != \; R_1$

  e.g. $\{1\} \cup \{1\} = \{1, 1\} \; != \; \{1\}$

Union Compatibility

- $\cup$, $\cap$, and $-$ are three operators defined as being union compatible relations.

  i.e. they have two relations with the same set of attributes, and for each attribute, they have the same type (domain).

Selection

Recall: it selects the row which satisfies its condition.

$$Johnny's\ Menu := \sigma_{bar="Johnny's"}(Sells)$$

Sells

bar	beer	price
Johnny's	Bud	5.50
Johnny's	Miller	5.75
Brandy's	Bud	5.50
Brandy's	Miller	5.00

Johnny's Menu

bar	beer	price
Johnny's	Bud	5.50
Johnny's	Miller	5.75

Projection

The resulting *schema* (i.e. Price) contains exactly the same fields that are in the projection list, with the same names.

$$Prices := \pi_{beer,price}(Sells)$$

Sells

bar	beer	price
Johnny's	Bud	5.50
Johnny's	Miller	5.75
Brandy's	Bud	5.50
Brandy's	Miller	5.00

Prices

beer	price
Bud	5.50
Bud	5.75
Miller	5.50
Miller	5.00

Extended Projection

$$R_2 := \pi_{A+B \to C,\ A \to A_1,\ A \to A_2}(R_1)$$

R_1

A	B
1	2
3	4
5	6

R_2

C	A_1	A_2
3	1	1
7	3	3
11	5	5

Cartesian Product

R_1

A	B
1	2
3	4

R_2

B	C
5	6
7	8
9	10

$R_3 := R_1 \times R_2$

A	R_1.B	R_2.B	C
1	2	5	6
1	2	7	8
1	2	9	10
3	4	5	6
3	4	7	8
3	4	9	10

Union

R_1

A	B
1	2
3	4
1	2

R_2

A	B
1	2
3	4
5	6

$R_1 \cup R_2$

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

Final Result

$R_1 \cup R_2$

A	B
1	2
3	4
5	6

Intersection

R_1

A	B
1	2
3	4
1	2

R_2

A	B
1	2
3	4
5	6

$R_1 \cap R_2$

A	B
1	2
3	4

Difference

R_1

A	B
1	2
3	4
1	2

R_2

A	B
1	2
3	4
5	6

$R_1 - R_2$

A	B
1	2

Natural Join

Sells(

bar,	beer,	price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Coors	3.00

)

Bars(

bar,	addr
Joe's	Maple St.
Sue's	River Rd.

)

BarInfo := Sells ⋈ Bars

Note: Bars.name has become Bars.bar to make the natural join non-trivial

BarInfo(

bar,	beer,	price,	addr
Joe's	Bud	2.50	Maple St.
Joe's	Milller	2.75	Maple St.
Sue's	Bud	2.50	River Rd.
Sue's	Coors	3.00	River Rd.

)

Theta Join

Sells(

bar,	beer,	price)
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Coors	3.00

Bars(

name,	addr)
Joe's	Maple St.
Sue's	River Rd.

BarInfo := Sells $\bowtie_{\text{Sells.bar = Bars.name}}$ Bars

BarInfo(

bar,	beer,	price,	name,	addr)
Joe's	Bud	2.50	Joe's	Maple St.
Joe's	Miller	2.75	Joe's	Maple St.
Sue's	Bud	2.50	Sue's	River Rd.
Sue's	Coors	3.00	Sue's	River Rd.

In this example, note that θ is $=$, therefore, this is also an equijoin.

Renaming

A	B	C
2	1	3
4	2	2

R

$\mathbf{P}:=\rho_{P(A\to D,C\to F)}(\mathrm{R})$

D	F
2	3
4	2

**this would require a column select of A and C prior to renaming

P

$\mathbf{S}:=\rho_{S(D,E,F)}(\mathrm{R})$

D	E	F
2	1	3
4	2	2

S

$\mathbf{Q}:=\rho_{Q}(\mathrm{R})$

A	B	C
2	1	3
4	2	2

Q

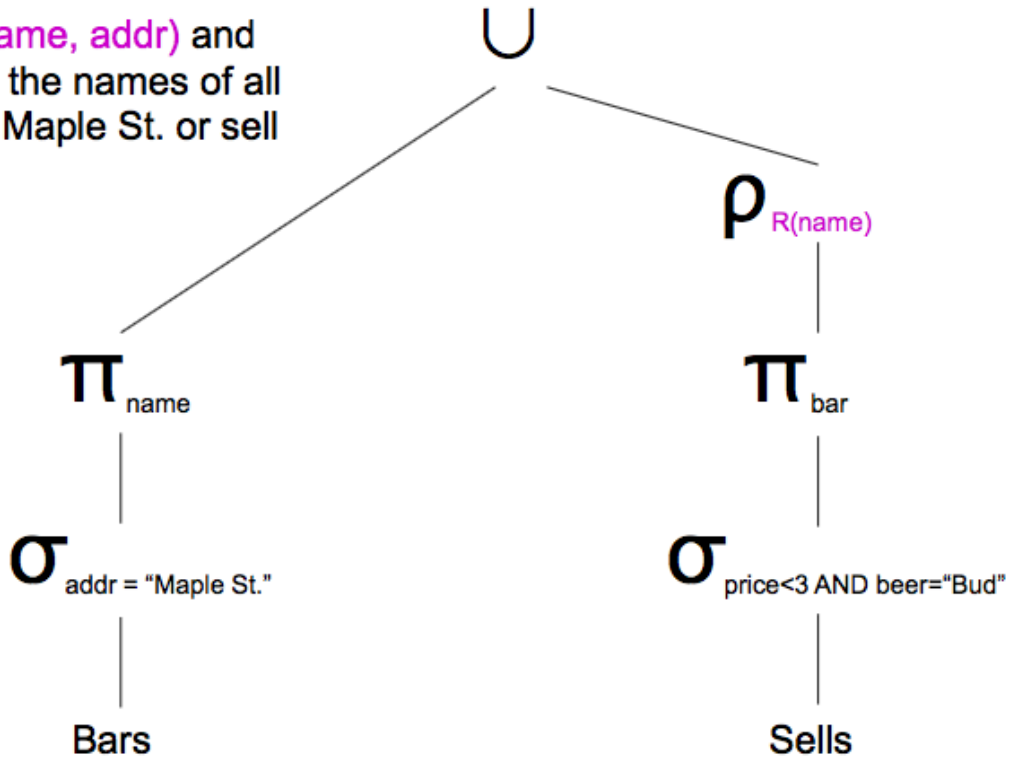
Expression Trees

- Leaves are operands – either variables standing for relations or particular, constant relations.
- Interior nodes are operators – applied to their child(ren).

e.g. Using the relations Bars(name, addr) and Sells(bar, beer, price) find the names of all the bars that are either on "Maple St." or sell "Bud" for less than $3.

Expression Trees

Using the relations Bars(name, addr) and Sells(bar, beer, price), find the names of all the bars that are either on Maple St. or sell Bud for less than $3.



Semantic Overview & Discussion

Blackboard Notes!

Think-Pair-Share

1. You will work in a group with your table, take a couple minutes to come up with a solution.
2. The first table will then turn around, and face the second table; the same with the third and forth tables, fifth and sixth, and so on and so forth.
3. You will then share your thoughts and spend a couple more minutes to conform to one answer.
4. You will now share your answer with the class.

The number on your table denotes the True/False question your group will solve.

True or False? – Solutions

True/False. Illustrate the validity by an example, or disprove the validity by a counter-example:

1. Let R_1 and R_2 be two relations, where R_1 has the entities $\{A, B\}$ and R_2 has the entities $\{B, C\}$. $\sigma_{A=B}(R_1 \times R_2) = (R_1 \bowtie_{A=B} R_2)$
2. The size of the resulting relation of $R_1 \bowtie R_2$ is always strictly smaller than the size of the resulting relation of $R_1 \times R_2$.
3. Let Q_1 and Q_2 be conjunctive queries.
   i. if $Q_2 \subseteq_{BAG} Q_1$, then $Q_2 \subseteq Q_1$
   ii. if $Q_2 \subseteq Q_1$, then $Q_2 \subseteq_{BAG} Q_1$

True or False? – Solutions

Question 1

Let R_1 and R_2 be two relations, where R_1 has the entities $\{A, B\}$ and R_2 has the entities $\{B, C\}$.

$$\sigma_{A=B}(R_1 \times R_2) = (R_1 \bowtie_{A=B} R_2)$$

True $\rightarrow$ these are tautologically equivalent.

True or False? – Solutions

Question 2

The size of the resulting relation of $R_1 \bowtie R_2$ is always strictly smaller than the size of the resulting relation of $R_1 \ x \ R_2$.

False $\rightarrow$ a counter-example would be one tuple in each of the relations R_1 and R_2, resulting in an equal result (one tuple) for both $R_1 \bowtie R_2$ and $R_1 \ x \ R_2$.

True or False? – Solutions

Question 3

i. Given the conjunctive queries Q_1 and Q_2, if $Q_2 \subseteq_{BAG} Q_1$, then $Q_2 \subseteq Q_1$.

False $\rightarrow \{1, 2, 1, 3\}$ is a BAG but not a SET.

ii. Given the conjunctive queries Q_1 and Q_2, if $Q_2 \subseteq Q_1$, then $Q_2 \subseteq_{BAG} Q_1$.

True $\rightarrow \{1, 2, 3\}$ is a BAG that is also a SET.

You may also refer to the example from the blackboard!

Recall: Intended Learning Outcomes

1. Understand the differences between the core relational algebra operands and operators.
2. Understand the rules of precedence.
3. Differentiate between bag and set semantics.
4. Utilize expression trees to help formulate your RA solution.
5. Translate SQL to RA and RA to SQL.