These lecture notes include some material from Professors Bertossi, Kolaitis, Guagliardo and Libkin

Relational Algebra 3

Lecture Handout

Dr Eugenia Ternovska

Simon Fraser University

In this part

- Independence of the Basic Relational Algebra (RA) Operations
- Strength from Unity and Combination
- Relational Completeness
- SQL vs. Relational Algebra
- Additional Operators
- Combining Operators to Form Queries
- Three Types of RA Notations:
  - Algebraic RA Expressions
  - Trees
  - Linear Notations

Independence of the Basic Relational Algebra Operations

Question: Are all the basic relational algebra operations really needed? Can one of them be expressed in terms of the other five?

Theorem: Each of the basic relational algebra operations is independent of the other five, that is, it **cannot** be expressed by a relational algebra expression that involves only the other five.

Proof Idea: For each relational algebra operation, we need to discover a property that is possessed by that operation, but is not possessed by any relational algebra expression that involves only the other five operations.

Independence of the Basic Relational Algebra Operations

Proof Sketch: (projection and cartesian product only)

Property of projection: It is the only operation whose output may have arity smaller than its input.

Show, by induction, that the output of every relational algebra expression in the other five basic relational algebra is of arity at least as big as the maximum arity of its arguments.

Property of cartesian product: It is the only operation whose output has arity bigger than its inputs.

Show, by induction, that the output of every relational algebra expression in the other five basic relational algebra is of arity at most as big as the maximum arity of its arguments.

Exercise: Complete this proof.

Strength from Unity and Combination

By itself, each basic relational algebra operation has limited expressive power, as it carries out a specific and rather simple task.

When used in combination, however, the basic relational algebra operations can express interesting and, quite often, rather complex queries including derived operations (e.g., natural join, division)

The basic relational algebra operations are independent of each other: none can be expressed in terms of the other.

Thus, Codd's choice of the basic relational algebra operations has been very judicious.

Relational Completeness

Definition (Codd – 1972): A database query language L is **relationally complete** if it is at least as expressive as relational algebra, i.e., every relational algebra expression E has an equivalent expression F in L.

Relational completeness provides a benchmark for the expressive power of a database query language.

Every commercial database query language should be at least as expressive as relational algebra.

Exercise: Explain why SQL is relationally complete, after we discuss how RA and SQL are related (next)

SQL vs. Relational Algebra

SQL	Relational Algebra
SELECT	Projection π
FROM	Cartesian Product $\times$
WHERE	Selection σ

Semantics of SQL via interpretation to Relational Algebra

SELECT $R_{i_1}.A_1, \ldots, R_{i_m}.A_m$
FROM $R_1, \ldots, R_k$
WHERE Ψ

$$= \pi R_{i_1}.A_1, \ldots, R_{i_m}.A_m(\sigma\Psi(R_1 \times \cdots \times R_k))$$

Additional Operators

There are additional relational algebra operators
- Usually used in the context of query optimization

Duplicate elimination – ϵ (or δ)

- Used to turn a bag (multiset) into a set

Aggregation operators

- e.g. sum, average

Grouping – γ

- Used to partition tuples into groups
- Typically used with aggregation

Combining Operators to Form Queries

Example

What are the titles and years of movies made by Fox that are at least 100 minutes long?

How can we express it using the operations of RA?

Three types of notation

– will discuss in the next few slides:

1. Algebraic Expressions of RA (main notation)
2. Expression trees
3. Sequences of assignment statements

(2) and (3) are, essentially, small variations on (1)

(1) Algebraic Expressions of RA

Definition: A relational algebra expression is a string of symbols obtained from relation schemas using union, difference, cartesian product, projection, selection and renaming.

Context-free grammar for relational algebra expressions:

$$E := R, S, \ldots \mid (E_1 \cup E_2) \mid (E_1 - E_2) \mid (E_1 \times E_2) \mid \pi_L(E) \mid \sigma_\Theta(E) \mid \rho\, E,$$

where
$R, S, \ldots$ are relation schemas (i.e., relation names with attributes)
L is a list of attributes
Θ is a condition.

• Sometime, precedence order for operations rules are introduced, to omit parentheses, simplify notations and improve readability

(1) Algebraic Expressions of RA (Cont.)

Example

What are the titles and years of movies made by Fox that are at least 100 minutes long?

$$\pi_{title,year}(\sigma_{lengh \geq 100}(Movies) \cap \sigma_{studioName='Fox'}(Movies))$$

1. Select those Movies tuples that have $length >= 100$
2. Select those Movies tuples that have $studioName = `Fox'$
3. Compute the intersection of (1) and (2)
4. Project the relation from (3) onto attributes title and year

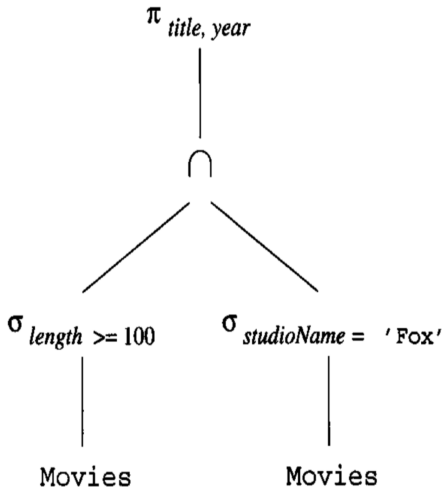
(2) Tree Expression

Example

What are the titles and years of movies made by Fox that are at least 100 minutes long?

$$\pi_{title,year}(\sigma_{lengh \geq 100}(Movies) \cap \sigma_{studioName='Fox'}(Movies))$$

- Evaluate bottom-up
- Apply the operator at each non-leaf node to its children
- Two selection nodes: steps (1) and (2)
- Intersection node: step (3)
- Projection node: step (4)



(3) Linear Notation

Example

What are the titles and years of movies made by Fox that are at least 100 minutes long?

$$\pi_{title,year}(\sigma_{lengh \geq 100}(Movies) \cap \sigma_{studioName=`Fox'}(Movies))$$

In Linear Notation, we write:

$$R(t, y, l, i, s, p) := \sigma_{lengh \geq 100}(Movies)$$

$$S(t, y, l, i, s, p) := \sigma_{studioName=`Fox'}(Movies)$$

$$Answer(title, year) := \pi_{t,y}(R \cap S)$$

(3) Linear Notation (Cont.)

• A set of assignments that compute a value for each temporary relation on the left of ":="

• Any algebraic expression can be used on the right of ":="

• Each temporary relation name has a parenthesized list of attributes for that relation

• Flexible order, as long as the values are ready

• The name $Answer$ will be used conventionally for the result of the final step

• The names for temporary relations on the left of ":=" correspond to interior nodes of the tree

Same Query, Different (but Equivalent) Expressions

Note: There could be more than one relational algebra expression that represents the same query

Examples

$$\pi_{title,year}(\sigma_{lengh \geq 100}(Movies) \cap \sigma_{studioName='Fox'}(Movies))$$

$$\pi_{title,year}(\sigma_{lengh \geq 100 \text{ AND } studioName='Fox'}(Movies))$$

The two expressions are semantically equivalent. They represent different computations of the same query

Equivalent Expressions & Query Optimization

Inside DBMSs:

- Answers to queries are computed using internal DBMS representations that are, essentially, relational algebra expressions
- The same query may have many equivalent algebraic expressions
- Some are much more quickly evaluated than others
- Query optimization: replace one expression of relational algebra by an **equivalent** expression that is more efficiently evaluated

Semantic query equivalence is a very important notion

Efficiency (1)

Consecutive selections can be combined into a single one:

$$\sigma_{\theta_1}\big(\sigma_{\theta_2}(R)\big) = \sigma_{\theta_1 \wedge \theta_2}(R)$$

Example

$Q_1 = \sigma_{\text{City}\neq\text{'Edinburgh'}}\big(\sigma_{\text{Age}<33}(\text{Customer})\big)$

$Q_2 = \sigma_{\text{City}\neq\text{'Edinburgh'} \wedge \text{Age}<33}(\text{Customer})$

$Q_1 = Q_2$ but Q_2 **faster** than Q_1 in general

Efficiency (2)

Projection can be pushed inside selection

$$\pi_\alpha\big(\sigma_\theta(R)\big) = \sigma_\theta\big(\pi_\alpha(R)\big)$$

only if all attributes mentioned in θ appear in α

Example

$Q_1 = \pi_{\text{Name,City,Age}}\big(\sigma_{\text{City}\neq\text{'Edinburgh'} \wedge \text{Age}<33}(\text{Customer})\big)$

$Q_2 = \sigma_{\text{City}\neq\text{'Edinburgh'} \wedge \text{Age}<33}\big(\pi_{\text{Name,City,Age}}(\text{Customer})\big)$

Question: Which one is more efficient?

Relational Algebra Summary

Relational Algebra operators:

- Five core operations: selection, projection, Cartesian product, union and set difference (plus renaming)
- Additional (derived) operations are defined in terms of the core (basic) operations,
  e.g. intersection, several kinds of joins, division

• SQL and Relational Algebra can express the same class of queries (SQL is relationally complete)

• Multiple Relational Algebra queries can be equivalent

- Same semantics but different performance
- Query equivalence forms the basis for optimization

Acknowledgements

[1] Database Systems: The Complete Book, 2nd EditionHector Garcia-Molina, Jeffrey D. Ullman, Jennifer WidomPrentice Hall, 2009

[2] Database System Concepts, Seventh EditionAvi Silberschatz, Henry F. Korth, S. SudarshanMcGraw-Hill, March 2019www.db-book.com

Additional references and resources used in preparation of this course are listed on the course webpage or mentioned in slides.