Relational Algebra

CMSC 206 – Database Management Systems

Revision History
20 Nov 2022: Examples update – Ria Borromeo
03 October 2020: Main content – Thomas Laurent

Outline

1. Introduction
2. Relational Algebra
    - Unary relational operations
    - Operations from set theory
    - Binary relational operations
    - Additional relational operations
3. Conclusion
4. Appendix

Introduction

- The relational algebra is a formal language, not an implementation. It works on the relational model and is rooted in Mathematics. *It is an Algebra in the Mathematical sense.*
- It gives us a very solid base to prove the answers of our DBMS and optimize things later on.
- It is the base of SQL, the query language used by most (all?) relational DBMS. We will discover SQL in the next lesson, and start working on concrete databases.
- In this lecture, we will define the **operations** of relational algebra and **expressions**, which are formed by combining operations.

Introduction

<u>**TAKE HOME POINTS:**</u>

- Relational algebra is a formal language on the relational model.
- Relational algebra consists of different operations.
- We need to start thinking about how we can use these operations to manipulate our data.

Relational Algebra

Relational Algebra

Unary Relational Operations

SELECT - σ - definition

- The select operation is used to obtain a subset of the tuples from a relation that contains all tuples satisfying a selection condition.
- We need a list of male employees:

$$MALES \leftarrow \sigma_{Sex="M"} (EMPLOYEE)$$

EMPLOYEE	Fname	Lname	Sex	SSN
	Franklin	Wong	M	333445555
	Jennifer	Wallace	F	987654321
	Ramesh	Narayan	M	666884444

SELECT - σ - syntax

- $\sigma_{\texttt{condition}}$(RELATION)
  - condition:
    - <attributename><comparisonoperator><attributename>
    - <attributename><comparisonoperator><constant>
    - <condition>AND/OR<condition>
    - NOT<condition>
  - **Examples:** `Sex = "M"; Age < 40; Sex = "M" AND Age < 40; Sex = "M" AND (Age < 40 OR Rich = True)`

  - **RELATION:**
    - a relation name
    - a relational algebra expression

Other points about SELECT - σ

- $\sigma_{\texttt{condition}}(\texttt{RELATION})$
- Selects tuples satisfying a condition
- The resulting relation has the same schema as the operand.
- The resulting relation's population is less or equal to that of the operand.

PROJECT - π - definition

- The project operation is used to obtain a subset of the attributes from a relation.
- Suppose we need the first name and SSN of all employees:
  - NAMES $\leftarrow \pi_{Fname,SSN}$(EMPLOYEE)

EMPLOYEE	Fname	Lname	Sex	SSN
	Franklin	Wong	M	333445555
	Jennifer	Wallace	F	987654321
	Ramesh	Narayan	M	666884444

Other points about PROJECT - π

- $\pi_{attributes}(RELATION)$
- Selects attributes from a relation
- The resulting relation has a different schema than the operand.
- Duplicates are eliminated.
- If the attributes on which we project form a super key then the result and operand have the same population.

The output of a SELECT operation: row(s)
The output of a PROJECT operation: column(s)

RENAME - ρ - definition

- The rename operation lets us rename a relation, its attributes or both.

- For example, suppose we want to rename the EMPLOYEE relation WORKER and the Sex attribute gender:

$$\rho_{WORKER(FName,\ LName,\ Gender,\ SSN)}(EMPLOYEE)$$

RENAME - ρ - syntax

- Rename without operator:
  - NEWR $\leftarrow$ R
  - R(FN, LN, SSN) $\leftarrow$ $\pi_{Fname,Lname,SSN}$(EMPLOYEE)

- Renaming with the operator:
  - Renaming relation and attributes: $\rho_{S(B_1,B_2,\ldots,B_n)}$ (R)
  - Renaming relation only: ρ_S (R)
  - Renaming attributes only: $\rho_{(B_1,B_2,\ldots,B_n)}$ (R)

Relational Algebra

Operations from Set Theory

Union, Intersection, and Set Difference

- These operators are binary operators and impose ***type compatibility*** between the two relations.
- Two relations $R_1(A_1, \ldots, A_n)$ and $R_2(B_1, \ldots, B_m)$ are ***type compatible*** or ***union compatible*** when:
  - They have he same degree: $m = n$
  - Their attributes have the same domains:

    $\forall_i \in \{1..n\}, \mathrm{Dom}(A_i) = \mathrm{Dom}(B_i)$
- The operators are:
  - R ∪ S: set of tuples in R ***or*** S. Duplicate tuples are eliminated.
  - R ∩ S: set of tuples in ***both*** R ***and*** S.
  - R − S: set of tuples in R ***but not*** in S.

Relational Algebra

Binary Relational Operations

CROSS PRODUCT - ×

- This binary operator creates all tuple combinations from two (2) relations.
- It imposes that the two relations do not have attributes of the same name. To avoid this limitation, we consider that all attribute names are prepended with the table name (e.g. here R's A attribute is R.A. If S had an A attribute, it would be S.A)

R	A	B
	a	b
	b	c

S	C	D	E
	c	d	e
	b	a	b
	a	a	c

R×S	A	B	C	D	E
	a	b	c	d	e
	a	b	b	a	b
	a	b	a	a	c
	b	c	c	d	e
	b	c	b	a	b
	b	c	a	a	c

(INNER) JOIN - $\bowtie$

- R $\bowtie_{condition}$ S
- Equivalent to a CROSS PRODUCT followed by a SELECT
- Links corresponding tuples from different relations
- We classify joins under different names:
  - THETA JOIN: general condition (i.e. the basic join)
  - EQUIJOIN: only check for equality of attributes
  - NATURAL JOIN (noted *): EQUIJOIN on attributes of the same name in both relations

JOIN - ⋈ - Example

R	A	B
	a	b
	b	c

S	C	D	E
	c	d	e
	b	a	b
	a	a	c

$R \bowtie_{A!=C} S$	A	B	C	D	E
	a	b	c	d	e
	a	b	b	a	b
	b	c	c	d	e
	b	c	a	a	c

NATURAL JOIN - * - Example

R	A	B
	a	b
	b	c

S	A	D	E
	c	d	e
	b	a	b
	a	a	c

R∗S	A	B	D	E
	a	b	a	c
	b	c	a	b

Division - ÷

- $R(A_1,\ldots,A_n,\ldots,A_m) \div S(A_1,\ldots,A_n)$
- S's attributes must be a subset of R's
- The result relation has attributes A_{n+1} to A_m
- $R \div S = \{< a_{n+1},\ldots,a_m > \mid$

$$\forall < a_1,\ldots,a_n > \in S, < a_1,\ldots,a_n,\ldots,a_m > \in R\}$$

R ÷ S, is the set of tuples that, when joined with every tuple in S, are in R.

Division - Examples

R	A	B	C
	1	1	1
	1	2	0
	1	2	1
	1	3	0
	2	1	1
	2	3	3
	3	1	1
	3	2	0
	3	2	1

S	B	C
	3	5

$R \div S$	A

S	B	C
	1	1
	2	0

$R \div S$	A
	1
	3

S	B	C
	1	1

$R \div S$	A
	1
	2
	3

Relational Algebra

Additional Relational Operations

Generalized projection - π

- This operation extends the projection operator to allow to *project on functions of the attributes*.
- For example, if we have a database of objects with their prices, excluding tax, and we want to query the database for all objects and their price including tax (e.g. 10%):
  - $\pi_{id, 1.1 \times price}$ (OBJECT)

OUTER JOIN Operation

- Same as a JOIN but keeps tuples which do not have a match in the other table (padding with NULL)

- Several flavors:
  - LEFT OUTER JOIN - ⟕ - keeps tuples from the left relation
  - RIGHT OUTER JOIN - ⟖ - keeps tuples from the right relation
  - FULL OUTER JOIN - ⟗ - keeps tuples from both relations

OUTER JOIN Operation - Example

R	A	B
	a	b
	b	c

S	C	D	E
	c	d	e
	b	a	b
	a	a	c

$R \rtimes_{A=C} S$	A	B	C	D	E
	a	b	a	a	c
	b	c	b	a	b
	NULL	NULL	c	d	e

OUTER UNION Operation

- Extends union to relations that are ***not type compatible*** but that have ***some attributes in common***.
- It is equivalent to a FULL OUTER JOIN on the common attributes.

Conclusion

- Relational Algebra is an important theoretical foundation for relational DBMSes that goes in hand with the relational model.
- We have seen a list of operations that might seem very abstract now, but that will take make sense later when we study SQL. ☺