CMSC 206: Database Management Systems

Relational algebra

Thomas Laurent

03/10/2020

University of the Philippines Open University

Table of contents

Introduction

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 optimise things later on.

It is the base of SQL, the query language used by most (all?) relational DBMS. We will discover SQL next week, and start working on concrete databases.

Here we will define the *operations* of relational algebra. Multiple operations together form an *expression*.

Introduction

Take home points

- Relational algebra is a formal language on the relational model
- Different operations of the relational algebra
- 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:

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

EMPLOYEE	FName	LName	Sex	Ssn
	Franklin	Wong	M	333445555
	Jennifer	Wallace	F	987654321
	Ramesh	Narayan	M	666884444

SELECT - σ - syntax

- $\sigma_{\text{condition}}$(RELATION)
  - condition:
    - <attribute name> <comparison operator> <attribute name>
    - <attribute name> <comparison operator> <constant>
    - <condition> AND/OR <condition>
    - NOT <condition>

    Example: 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

SELECT - σ - memo

- $\sigma_{\text{condition}}$(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

Used to obtain a *subset* of the *attributes* from a relation.

We need the first name and Ssn of all employees:

$$\text{NAMES} \leftarrow \pi_{\text{Fname,Ssn}}(\text{EMPLOYEE})$$

FName	LName	Sex	Ssn
Franklin	Wong	M	333445555
Jennifer	Wallace	F	987654321
Ramesh	Narayan	M	666884444

PROJECT - π - memo

- $\pi_{\text{attributes}}(\text{RELATION})$
- 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*.

RENAME - ρ - definition

The rename operation lets us rename a relation, its attributes or both.
We want to rename the EMPLOYEE relation WORKER and the Sex attribute gender.

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

RENAME - ρ - syntax

- Rename without operator:
  - NEWR $\leftarrow$ R
  - R(FN,LN,SSN) $\leftarrow \pi_{\text{Fname,Lname,Ssn}}$(EMPLOYEE)
- Renaming with the operator:
  - Renaming relation and attributes: $\rho_{S(B1,B2,...,B_n)}(R)$
  - Renaming relation only: $\rho_S(R)$
  - Renaming attributes only: $\rho_{(B1,B2,...,B_n)}(R)$

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 $R_1(A_1, ..., A_n)$ and $R_2(B_1, ..., B_m)$ are *type compatible* or *union compatible* when:

- They have he same degree: $m = n$
- Their attributes have the same domains:
  $\forall i \in \{1..n\}, Dom(A_i) = Dom(B_i)$

The operators are:

- R $\cup$ S: set of tuples in R *or* S. Duplicate tuples are eliminated.
- R $\cap$ 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 - $\times$

Binary operator that 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)
Creates all tuple combinations from the two relations.

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 $\bowtie_{\text{condition}}$ 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 - $\div$

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

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

  i.e. $R \div S$, is the set of tuples that, when joined with every tuple in S, are in R.

Division - examples

S	B	C
	3	5

$R \div S$	A

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
	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

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 object and their price including tax (in Japan):

$$\pi_{\text{id},1.1\times\text{price}}(\text{OBJECT})$$

OUTER JOIN operation

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

- Several flavours:
  - 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

$R ⟖_{A=C} S$	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

Conclusion

The relational algebra is an important theoretical foundation for relational DBMSes that goes in hand with the relational model we saw last week.

We have seen a list of operations that might seem very abstract now, but that will take all their sense next week, when we study SQL and start really playing with DBs in a DBMS.