# Relational Algebra

André Restivo

# Index

# Introduction

# Relational Model

- A set of relations defined by their schemas.
- Each relation is composed by attributes and tuples.
- Schema of a relation R with attributes a_1, a_2, a_3, ..., a_n:

```
? R(a_1, a_2, a_3, ..., a_n)
```

# Relational Model

- The cardinality (number of tuples) in relation R:

? |R|

- A tuple with attribute values v_1, v_2, v_3, …, v_n:

? t = <v_1, v_2, v_3, …, v_n>

- Attribute a_i belonging to relation R:

? R.a_i

- The domain of attribute a_i:

? Dom(a_i)

- The *null* value:

? \perp

# Example

?

Relation: Employee(id, name, salary, taxes, department)

Tuple: t = <1234, John, 1200, 200, 3>

Attribute: Employee.name

Domain: Dom(Employee.name) = text

# Example

Employee(id, name, salary, taxes, department)

| id | name | salary | taxes | departament |
|---|---|---|---|---|
| 1 | John Doe | 1000 | 200 | 1 |
| 2 | Jane Doe | 800 | 100 | 2 |
| 3 | John Smith | 1200 | 350 | 2 |
| 4 | Jane Roe | 1000 | 200 | 3 |
| 5 | John Doe | 1000 | 0 | \perp |

$|\text{Employee}| = 5$

# Unary Operators

# Projection

The result of a projection is defined as the set that is obtained when all tuples in R are restricted to the set <a_1, ..., a_n>

? $\Pi_{a_1,...,a_n}(R)$

Consider L as a list containing attributes from R:

? $S = \Pi_{L}(R)$

Relation S will only have the attributes from that list.

? If L does not contain a key from R, repeated tuples are eliminated.

# Example

? S = \Pi_{name, salary}(Employee)

| name | salary |
|---|---|
| John Doe | 1000 |
| Jane Doe | 800 |
| John Smith | 1200 |
| Jane Roe | 1000 |

? Notice that one line was eliminated.

# Projection

## Renaming and Arithmetic Operators

The projection operator can also be used to rename attributes:

```
? S = \Pi_{name, wages = salary}(Employee)
```

And calculate simple arithmetic expressions:

```
? S = \Pi_{name, wages = salary - taxes}(Employee)
```

For simplicity the result should be renamed.

# Example

? S = \Pi_{name, wages = salary - taxes}(Employee)

| id | name | salary | taxes | departament |
|---|---|---|---|---|
| 1 | John Doe | 1000 | 200 | 1 |
| 2 | Jane Doe | 800 | 100 | 2 |
| 3 | John Smith | 1200 | 350 | 2 |
| 4 | Jane Roe | 1000 | 200 | 3 |
| 5 | John Doe | 1000 | 0 | \perp |

| name | wages |
|---|---|
| John Doe | 800 |
| Jane Doe | 700 |
| John Smith | 850 |
| Jane Roe | 800 |
| John Doe | 1000 |

# Rename

Renaming the relation R to S:

? \rho_S(R)

Renaming attribute a to atribute x in relation R(a,b,c):

? \rho_{a/x}(R) \Rightarrow R(x,b,c)

Renaming attribute a to atribute x and c to atribute y in relation R(a,b,c):

? \rho_{a/x, c/y}(R) \Rightarrow R(x,b,y)

# Selection

Select a set of tuples where a certain condition $c$ holds:

? $S = \sigma_c(R)$

- $c$ is a condition involving attributes from $R$.
- The condition can contain arithmetic ($+ - \times \div$), conditional ($< > \leq \geq \neq$) and logical ($\vee \wedge \neg$) operators.
- $S$ has the same attributes as $R$.

# Example

? S = \sigma_{salary < 1000 \vee department = 3}(Employee)

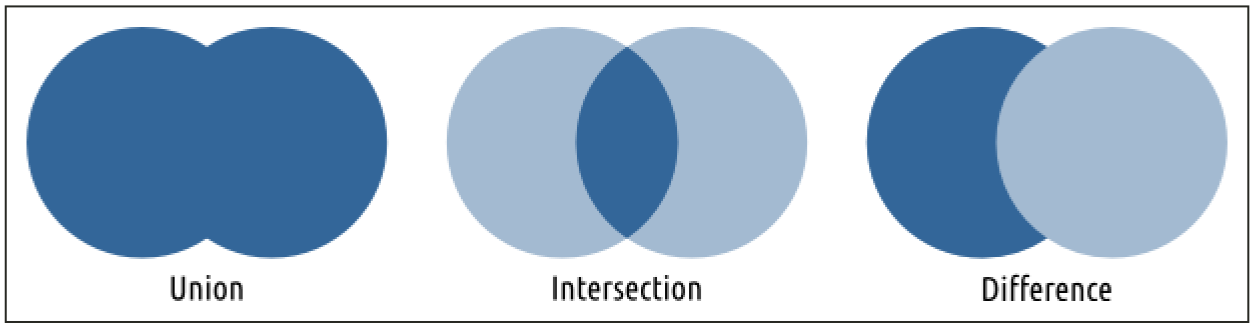
Employees with a salary smaller than 1000 or that work at department 3.

| id | name | salary | departament |
| --- | --- | --- | --- |
| 1 | John Doe | 1000 | 1 |
| 2 | Jane Doe | 800 | 2 |
| 3 | John Smith | 1200 | 2 |
| 4 | Jane Roe | 1000 | 3 |
| 5 | John Doe | 1000 | \perp |

| id | name | salary | departament |
| --- | --- | --- | --- |
| 2 | Jane Doe | 800 | 2 |
| 4 | Jane Roe | 1000 | 3 |

# Set Operators

# Union, Intersection and Difference



# Union, Intersection and Difference

The two relations involved must be union-compatible:

- they must have the same number of attributes
- the domain of each attribute must be the same in both R and S

?

$$R \cup S = \{x: x \in R \vee x \in S\}$$

$$R \cap S = \{x: x \in R \wedge x \in S\}$$

$$R - S = \{x: x \in R \wedge x \notin S\}$$

# Union Example

Employee

| id | name | salary | taxes | departament |
|---|---|---|---|---|
| 1 | John Doe | 1000 | 200 | 1 |
| 2 | Jane Doe | 800 | 100 | 2 |
| 3 | John Smith | 1200 | 350 | 2 |
| 4 | Jane Roe | 1000 | 200 | 3 |
| 5 | John Doe | 1000 | 0 | \perp |

Manager

| id | name | salary | taxes | departament |
|---|---|---|---|---|
| 1 | John Doe | 1000 | 200 | 1 |
| 6 | Big Boss | 5000 | 0 | \perp |

Employee \cup Manager

| id | name | salary | taxes | departament |
|---|---|---|---|---|
| 1 | John Doe | 1000 | 200 | 1 |
| 2 | Jane Doe | 800 | 100 | 2 |
| 3 | John Smith | 1200 | 350 | 2 |
| 4 | Jane Roe | 1000 | 200 | 3 |
| 5 | John Doe | 1000 | 0 | \perp |
| 6 | Big Boss | 5000 | 0 | \perp |

# Intersection Example

Employee

|id|name|salary|taxes|departament |-|-|javascript/#46
|1 | John Doe | 1000 | 200 | 1 |2 | Jane Doe | 800 | 100 | 2
|3 | John Smith | 1200 | 350 | 2 |4 | Jane Roe | 1000 | 200
| 3 |5 | John Doe | 1000 | 0 | \perp

Manager

| id | name | salary | taxes | departament |
|---|---|---|---|---|
| 1 | John Doe | 1000 | 200 | 1 |
| 6 | Big Boss | 5000 | 0 | \perp |

Employee \cap Manager

| id | name | salary | taxes | departament |
|---|---|---|---|---|
| 1 | John Doe | 1000 | 200 | 1 |

# Difference Example

Employee

| id | name | salary | taxes | departament |
|---|---|---|---|---|
| 1 | John Doe | 1000 | 200 | 1 |
| 2 | Jane Doe | 800 | 100 | 2 |
| 3 | John Smith | 1200 | 350 | 2 |
| 4 | Jane Roe | 1000 | 200 | 3 |
| 5 | John Doe | 1000 | 0 | \perp |

Manager

| id | name | salary | taxes | departament |
|---|---|---|---|---|
| 1 | John Doe | 1000 | 200 | 1 |
| 6 | Big Boss | 5000 | 0 | \perp |

Employee - Manager

| id | name | salary | taxes | departament |
|---|---|---|---|---|
| 2 | Jane Doe | 800 | 100 | 2 |
| 3 | John Smith | 1200 | 350 | 2 |
| 4 | Jane Roe | 1000 | 200 | 3 |
| 5 | John Doe | 1000 | 0 | \perp |

# Joins

# Cartesian Product

The cartesian product $R \times S$ is the set of all ordered pairs $(r, s)$ where $r \in R$ and $s \in S$.

$$R \times S = \{ < r,s > : r \in R, s \in S \}$$

The cartesian product between relations $R(a_1,...,a_n)$ and $S(b_1,...,b_m)$ is a relation with $n+m$ attributes $(a_1, ..., a_n, b_1, ..., b_m)$ where there is a tuple for each possible combination of tuples from $R$ and $S$.

The cardinality of the resulting relation is equal to the product between the cardinalities of the original relations:

? $|R \times S| = |R| \times |S|$

# Example

Employee

| id | name | departament |
|---|---|---|
| 1 | John Doe | 1 |
| 2 | Jane Doe | 2 |
| 3 | John Smith | 2 |
| 4 | John Doe | \perp |

Department

| number | designation |
|---|---|
| 1 | Marketing |
| 2 | Accounting |

Employee \times Department

| id | name | department | number | designation |
|---|---|---|---|---|
| 1 | John Doe | 1 | 1 | Marketing |
| 2 | Jane Doe | 2 | 1 | Marketing |
| 3 | John Smith | 2 | 1 | Marketing |
| 4 | John Doe | \perp | 1 | Marketing |
| 1 | John Doe | 1 | 2 | Accounting |
| 2 | Jane Doe | 2 | 2 | Accounting |
| 3 | John Smith | 2 | 2 | Accounting |
| 4 | John Doe | \perp | 2 | Accounting |

# Conditional Join

A cartesian product between relations $R$ and $S$ followed by a selection on condition $c$:

? $$R \bowtie_c S$$

The same as a cartesian product followed by a selection:

? $$R \bowtie_c S = \sigma_c(R \times S)$$

Allows the combination of relations that are associated by a foreign key.

# Example

Employee

| id | name | departament |
|---|---|---|
| 1 | John Doe | 1 |
| 2 | Jane Doe | 2 |
| 3 | John Smith | 2 |
| 4 | John Doe | \perp |

Department

| number | designation |
|---|---|
| 1 | Marketing |
| 2 | Accounting |

Employee \bowtie_{department=number} Department

| id | name | department | number | designation |
|---|---|---|---|---|
| 1 | John Doe | 1 | 1 | Marketing |
| 2 | Jane Doe | 2 | 2 | Accounting |
| 3 | John Smith | 2 | 2 | Accounting |

# Natural Join

A particular case of a join where the condition is the equality of attributes on both relations having the same name.

? R \bowtie S

Attributes used in the condition are merged together.

# Example

Employee

| id | name | number |
|---|---|---|
| 1 | John Doe | 1 |
| 2 | Jane Doe | 2 |
| 3 | John Smith | 2 |
| 4 | John Doe | \perp |

Department

| number | designation |
|---|---|
| 1 | Marketing |
| 2 | Accounting |

Employee \bowtie Department

| id | name | number | designation |
|---|---|---|---|
| 1 | John Doe | 1 | Marketing |
| 2 | Jane Doe | 2 | Accounting |
| 3 | John Smith | 2 | Accounting |

# Semijoin

A join where only attributes from one of the relations is kept.

? $R \ltimes S = \Pi_R (R \bowtie S)$

? $R \rtimes S = \Pi_S (R \bowtie S)$

# Examplejavascript/#46

Employee

| id | name | number |
|---|---|---|
| 1 | John Doe | 1 |
| 2 | Jane Doe | 2 |
| 3 | John Smith | 2 |
| 4 | John Doe | \perp |

Department

| number | designation |
|---|---|
| 1 | Marketing |
| 2 | Accounting |
| 3 | Transports |

Employee \ltimes Department

| id | name | number |
|---|---|---|
| 1 | John Doe | 1 |
| 2 | Jane Doe | 2 |
| 3 | John Smith | 2 |

Employee \rtimes Department

| number | designation |
|---|---|
| 1 | Marketing |
| 2 | Accounting |

# Outer Join

A join operation where unmatched tuples are part of the result set. This tuples can come from the R relation (left), the S relation (right) or from both (full).

Left outer join:

```
? R ? _c S
```

Right outer join:

```
? R ?_c S
```

Full outer join:

```
? R ?_c S
```

# Example

Employee

| id | name | number |
|---|---|---|
| 1 | John Doe | 1 |
| 2 | Jane Doe | 2 |
| 3 | John Smith | 2 |
| 4 | John Doe | \perp |

Department

| number | designation |
|---|---|
| 1 | Marketing |
| 2 | Accounting |
| 3 | Transports |

Employee ? Department

| id | name | number | designation |
|---|---|---|---|
| 1 | John Doe | 1 | Marketing |
| 2 | Jane Doe | 2 | Accounting |
| 3 | John Smith | 2 | Accounting |
| 4 | John Doe | \perp | \perp |

Employee ? Department

| id | name | number | designation |
|---|---|---|---|
| 1 | John Doe | 1 | Marketing |
| 2 | Jane Doe | 2 | Accounting |
| 3 | John Smith | 2 | Accounting |
| \perp | \perp | 3 | Transports |

# Division

# Division

The restrictions of tuples in R to the attribute names unique to R for which it holds that all their combinations with tuples in S are present in R.

? R(a,b,c) \div S(b,c)

In this example, the result of the division will have one attribute a (the one that does not exist in S), containing the values of a that are combined with all values of S.

# Example

Works

| id | name | number | designation |
|---|---|---|---|
| 1 | John Doe | 1 | Big Rocket |
| 1 | John Doe | 2 | Thingamabob |
| 1 | John Doe | 3 | Take a Nap |
| 2 | Jane Doe | 2 | Thingamabob |
| 2 | Jane Doe | 3 | Take a Nap |
| 3 | Jack Doe | 1 | Big Rocket |
| 3 | Jack Doe | 2 | Thingamabob |

Project

| number | designation |
|---|---|
| 1 | Big Rocket |
| 2 | Thingamabob |
| 3 | Take a Nap |

Works \div Project

| id | name |
|---|---|
| 1 | John Doe |

# Division without Division

? $R(a_1,\ldots,a_n,b_1,\ldots,b_m) \div S(b_1, \ldots, b_m)$

? $R \div S = \Pi_{a_1,\ldots,a_n}(R) - \Pi_{a_1,\ldots,a_n} ( \Pi_{a_1,\ldots,a_n}(R) \times S - R)$

# Explanation

All tuples from the first n attributes of R:

? \Pi_{a_1,...,a_n}(R)

? \Pi_{id,name}(Works)

| id | name |
|---|---|
| 1 | John Doe |
| 2 | Jane Doe |
| 3 | Jack Doe |

# Explanation

Cartesian product with S gives all possible combinations of those attributes with the tuples of S:

? \Pi_{a_1,...,a_n}(R) \times S

? \Pi_{id,name}(Works) \times Project

| id | name | number | designation |
|---|---|---|---|
| 1 | John Doe | 1 | Big Rocket |
| 2 | Jane Doe | 1 | Big Rocket |
| 3 | Jack Doe | 1 | Big Rocket |
| 1 | John Doe | 2 | Thingamabob |
| 2 | Jane Doe | 2 | Thingamabob |
| 3 | Jack Doe | 2 | Thingamabob |
| 1 | John Doe | 3 | Take a Nap |
| 2 | Jane Doe | 3 | Take a Nap |
| 3 | Jack Doe | 3 | Take a Nap |

# Explanation

Removing the original R tuples we get the tuples that are not present in the original R relation:

? \Pi_{a_1,...,a_n}(R) \times S - R

? \Pi_{id,name}(Works) \times Project - Works

| id | name | number | designation |
|---|---|---|---|
| 2 | Jane Doe | 1 | Big Rocket |
| 3 | Jack Doe | 3 | Take a Nap |

# Explanation

Projecting again we get the first n attributes of those tuples:

? \Pi_{a_1,...,a_n} ( \Pi_{a_1,...,a_n}(R) \times S - R)

? \Pi_{id,name} (\Pi_{id,name}(Works) \times Project - Works)

| id | name |
|---|---|
| 2 | Jane Doe |
| 3 | Jack Doe |

# Explanation

? $\Pi_{a_1,...,a_n}(R) - \Pi_{a_1,...,a_n} ( \Pi_{a_1,...,a_n}(R) \times S - R)$

? $\Pi_{id,name}(Works) - \Pi_{id,name} (\Pi_{id,name}(Works) \times Project - Works)$

| id | name |
|---|---|
| 1 | John Doe |