Lecture 3

Chapter 2 Sections 4 & 5, Relational Algebra & Constraints

John Connor

August 20, 2019

Why Do We Need the Relational Algebra?

Why Do We Need the Relational Algebra?

Is this “just theory”?

Why Do We Need the Relational Algebra?

Is this "just theory"? No way!

Why Do We Need the Relational Algebra?

Is this "just theory"? No way!

In addition to being the foundation of SQL, these ideas are found everywhere in functional programing. If you program in JavaScript, Python, Scala, or a .NET language you will use these operations every day!

Definition: Attribute

An attribute is a name and a type.

Definition: Attribute

An attribute is a name and a type.
Some examples or attributes

1. `ssn:number`
2. `name:string`
3. `birthday:date`

Definition: Schema

A schema is a name and a set of attributes which gives a specification for a multiset.

Definition: Relation

A relation is a schema and a multiset of tuples which conform to the schema.

Example: Student

An example schema:

`Student(name:string, emplid:number, age:number)`

And a relation for this schema:

$$\{(\text{John Doe}, 123, 24), (\text{Jane Doe}, 456, 21)\}$$

Tablular Form

Instead of writing it all out in set notation, we will usually write the data in a tabular format

name	**emplid**	**number**
John Doe	123	24
Jane Doe	456	21

Table: *Student*

Operations

Most operations are defined in the "obvious" way, with the additional requirement that the two relations must be "compatible"; they must have the same schema.

Operations

Most operations are defined in the "obvious" way, with the additional requirement that the two relations must be "compatible"; they must have the same schema.

1. Union ($\cup$)
2. Intersection ($\cap$)
3. Difference ($-$)
4. Product ($\times$)
5. Projection (π)
6. Selection (σ)
7. Rename (ρ)
8. Natural Joins ($\bowtie$)
9. Theta Joins (θ)

Old Stuff

Union (∪)

name	address	gender	birthdate
Carrie Fisher	123 Maple St., Hollywood	F	9/9/99
Mark Hamill	456 Oak Rd., Brentwood	M	8/8/88

Table: R

name	address	gender	birthdate
Carrie Fisher	123 Maple St., Hollywood	F	9/9/99
Harrison Ford	789 Palm Dr., Beverly Hills	M	7/7/77

Table: S

Union ($\cup$)

Union ($\cup$)

name	**address**	**gender**	**birthdate**
Carrie Fisher	123 Maple St., Hollywood	F	9/9/99
Mark Hamill	456 Oak Rd., Brentwood	M	8/8/88
Harrison Ford	789 Palm Dr., Beverly Hills	M	7/7/77

Table: $R \cup S$

Intersection ($\cap$)

Intersection ($\cap$)

name	address	gender	birthdate
Carrie Fisher	123 Maple St., Hollywood	F	9/9/99

Table: $R \cap S$

New Stuff

Difference ($-$)

Difference ($-$)

name	**address**	**gender**	**birthdate**
Mark Hamill	456 Oak Rd., Brentwood	M	8/8/88

Table: $R - S$

Projection (π)

name	**gender**
Carrie Fisher	F
Mark Hamill	M

Table: $\pi_{name,gender}(R)$

Rename (ρ)

Assume you have two relations

name	**address**
Carrie Fisher	123 Maple St., Holywood
Mark Hamill	456 Oak Rd., Brentwood

Table: X

full name	**mailing address**
John Connor	1337 Haxor St., New York
Julius Caeser	1 Royal Palace Ln., Rome

Table: Y

Rename (ρ)

Assume you have two relations

name	**address**
Carrie Fisher	123 Maple St., Holywood
Mark Hamill	456 Oak Rd., Brentwood

Table: X

full name	**mailing address**
John Connor	1337 Haxor St., New York
Julius Caeser	1 Royal Palace Ln., Rome

Table: Y

These two relations have different schemas, so how can you perform a union, intersection or difference operation?

Rename (ρ)

full name	**mailing address**
Carrie Fisher	123 Maple St., Holywood
Mark Hamill	456 Oak Rd., Brentwood

Table: $\rho_{\text{name=fullname,address=mailing address}}(X)$

Product

The product does *not* require the relations to have the same schema.

Product

The product does *not* require the relations to have the same schema.

A	B
1	2
3	4

Table: *R*

B	C	D
2	5	6
4	7	8
9	10	11

Table: *S*

Product

The product does *not* require the relations to have the same schema.

A	B
1	2
3	4

Table: R

B	C	D
2	5	6
4	7	8
9	10	11

Table: S

A	R.B	S.B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

Table: $R \times S$

Natural Join ($\bowtie$)

The natural join also does not require the relations to have the same schema.

Natural Join ($\bowtie$)

The natural join also does not require the relations to have the same schema.
It's more useful than the full product, since it "joins" rows from the two relations when they have equal values for the attributes they have in common.

Example: Natural Join ($\bowtie$)

A	B
1	2
3	4

Table: R

B	C	D
2	5	6
4	7	8
9	10	11

Table: S

Example: Natural Join ($\bowtie$)

A	**B**
1	2
3	4

Table: R

B	**C**	**D**
2	5	6
4	7	8
9	10	11

Table: S

A	R.**B**	S.**B**	**C**	**D**
1	2	2	5	6
3	4	4	7	8

Table: $R \bowtie S$

θ-Join

The θ-join “filters” the product of two relations by some condition, denoted C.

Example: θ-Join

name	**height**
John Connor	6
Julia Childs	5
Julius Caeser	5

Table: A

name	**salary**
John Connor	1
Julia Childs	1,000
Julius Caeser	1,000,000

Table: B

A.name	**height**	**B.name**	**salary**
John Connor	6	John Connor	1

Table: $A \bowtie_{salary<height} B$

Putting It All Together: Queries

These operations can be combined to form more general queries.

Putting It All Together: Queries

These operations can be combined to form more general queries. For example, to get a relation containing the title and release year of all movies from the 'Fox' studio with a duration of at least 100:

Putting It All Together: Queries

These operations can be combined to form more general queries. For example, to get a relation containing the title and release year of all movies from the 'Fox' studio with a duration of at least 100:

$$\pi_{title,year}\left(\sigma_{\text{length}\geq 100}(Movies) \cap \sigma_{studioName='Fox'}(Movies)\right)$$

Putting It All Together: Queries

This expression can be represented as a tree:

$$\pi_{title,year}\left(\sigma_{\text{length}\geq 100}(Movies) \cap \sigma_{studioName='Fox'}(Movies)\right)$$

.

Constraints

Given the relations

`Movies`(<u>title</u>, <u>year</u>, length, genre, studioName, producerC#)
`StarsIn`(moviesTitle, moviesYear, starName)
`MovieExec`(name, address, cert#, netWorth)

Constraints

Given the relations

$\texttt{Movies}(\underline{\text{title}}, \underline{\text{year}}, \text{length}, \text{genre}, \text{studioName}, \text{producerC\#})$
$\texttt{StarsIn}(\text{moviesTitle}, \text{moviesYear}, \text{starName})$
$\texttt{MovieExec}(\text{name}, \text{address}, \text{cert\#}, \text{netWorth})$

What constraint on the data does this express?

$$\pi_{\text{moviesTitle, moviesYear}}(\texttt{StarsIn}) \subseteq \pi_{\text{title, year}}(\texttt{Movies})$$

Constraints

Given the relations

$\texttt{Movies}(\underline{\text{title}}, \underline{\text{year}}, \text{length}, \text{genre}, \text{studioName}, \text{producerC\#})$
$\texttt{StarsIn}(\text{moviesTitle}, \text{moviesYear}, \text{starName})$
$\texttt{MovieExec}(\text{name}, \text{address}, \text{cert\#}, \text{netWorth})$

What constraint on the data does this express?

$$\pi_{\text{moviesTitle, moviesYear}}(\texttt{StarsIn}) \subseteq \pi_{\text{title, year}}(\texttt{Movies})$$

And this one?

$$\pi_{\text{producerC\#}}(\texttt{Movies}) \subseteq \pi_{\text{cert\#}}(\texttt{MovieExec})$$