Lecture 3

Chapter 2 Section 4, Relational Algebra

John Connor

September 7, 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 of 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.

Definition: Database

A database is a set of schemas and their relations. (The book gives a much more technical definition, which we will not need.)

Example: Student

For example, if we have a database for storing information about movies, then it may contain a schema similar to the following:

$$\text{Actor(name : string, address : string, birthdate : date)}$$

And a relation for this schema:

{ (Carrie Fisher, 123 Maple St., Hollywood, 7/7/77),
(Mark Hamill, 456 Oak Rd., Brentwood, 8/8/88) }

Tablular Form

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

Actors

name	address	birthdate
Carrie Fisher	123 Maple St., Hollywood	7/7/77
Mark Hamill	456 Oak Rd., Brentwood	8/8/88

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

R

name	address	birthdate
Carrie Fisher	123 Maple St., Hollywood	7/7/77
Mark Hamill	456 Oak Rd., Brentwood	8/8/88

Union

R

name	address	birthdate
Carrie Fisher	123 Maple St., Hollywood	7/7/77
Mark Hamill	456 Oak Rd., Brentwood	8/8/88

S

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

Union

R

name	address	birthdate
Carrie Fisher	123 Maple St., Hollywood	7/7/77
Mark Hamill	456 Oak Rd., Brentwood	8/8/88

S

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

$R \cup S$

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

Intersection

R

name	address	birthdate
Carrie Fisher	123 Maple St., Hollywood	7/7/77
Mark Hamill	456 Oak Rd., Brentwood	8/8/88

S

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

Intersection

R

name	address	birthdate
Carrie Fisher	123 Maple St., Hollywood	7/7/77
Mark Hamill	456 Oak Rd., Brentwood	8/8/88

S

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

$R \cap S$

name	address	birthdate
Carrie Fisher	123 Maple St., Hollywood	7/7/77

Difference

R

name	address	birthdate
Carrie Fisher	123 Maple St., Hollywood	7/7/77
Mark Hamill	456 Oak Rd., Brentwood	8/8/88

S

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

Difference

R

name	address	birthdate
Carrie Fisher	123 Maple St., Hollywood	7/7/77
Mark Hamill	456 Oak Rd., Brentwood	8/8/88

S

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

$R \setminus S$

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

Difference

R

name	address	birthdate
Carrie Fisher	123 Maple St., Hollywood	7/7/77
Mark Hamill	456 Oak Rd., Brentwood	8/8/88

S

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

$R \setminus S$

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

$S \setminus R$

name	address	birthdate
Harrison Ford	789 Palm Dr., Beverly Hills	7/7/77

New Stuff

Projection

(π and "projection" both start with a "p" sound.)

R

name	address	birthdate
Carrie Fisher	123 Maple St., Hollywood	7/7/77
Mark Hamill	456 Oak Rd., Brentwood	8/8/88

Projection

(π and "projection" both start with a "p" sound.)

R

name	address	birthdate
Carrie Fisher	123 Maple St., Hollywood	7/7/77
Mark Hamill	456 Oak Rd., Brentwood	8/8/88

$\pi_{\text{name}}(R)$

name
Carrie Fisher
Mark Hamill

Projection

(π and "projection" both start with a "p" sound.)

R

name	address	birthdate
Carrie Fisher	123 Maple St., Hollywood	7/7/77
Mark Hamill	456 Oak Rd., Brentwood	8/8/88

$\pi_{\text{name}}(R)$

name
Carrie Fisher
Mark Hamill

$\pi_{\text{birthdate,name}}(R)$

birthdate	name
7/7/77	Carrie Fisher
8/8/88	Mark Hamill

Union of different schemas

$\pi_{\text{birthdate,name}}(\text{R})$

birthdate	name
7/7/77	Carrie Fisher
8/8/88	Mark Hamill

$\pi_{\text{name,birthdate}}(\text{R})$

name	birthdate
Carrie Fisher	7/7/77
Mark Hamill	8/8/88

$$\pi_{\text{name,birthdate}}(\text{R}) \ \cup \ \pi_{\text{birthdate,name}}(\text{R}) = ?$$

Union of different schemas

$\pi_{\text{birthdate,name}}(R)$

birthdate	name
7/7/77	Carrie Fisher
8/8/88	Mark Hamill

$\pi_{\text{name,birthdate}}(R)$

name	birthdate
Carrie Fisher	7/7/77
Mark Hamill	8/8/88

$$\pi_{\text{name,birthdate}}(R) \ \cup \ \pi_{\text{birthdate,name}}(R) = \text{X}$$

Union of different schemas

$\pi_{\text{birthdate,name}}(\mathrm{R})$

birthdate	name
7/7/77	Carrie Fisher
8/8/88	Mark Hamill

$\pi_{\text{name,birthdate}}(\mathrm{R})$

name	birthdate
Carrie Fisher	7/7/77
Mark Hamill	8/8/88

$\pi_{\text{name,birthdate}}(\mathrm{R}) \cup \pi_{\text{birthdate,name}}(\mathrm{R}) =$ X

This operation is undefined, as the schemas are not compatible.

Rename

(ρ starts with an "r" sound, just like "rename".)

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

R

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

S

fullname	**addr**
John Connor	1337 Haxor St., New York
Julius Caeser	1 Royal Palace Ln., Rome

Rename

(ρ starts with an "r" sound, just like "rename".)

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

R

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

S

fullname	addr
John Connor	1337 Haxor St., New York
Julius Caeser	1 Royal Palace Ln., Rome

We use the rename operation:

$$\rho_{\text{fullname}=\text{name},\text{addr}=\text{address}}(S)$$

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

Select (σ starts with an "s" sound, just like "select".)

σ allows us to filter out tuples from a relation when they do not match the predicate.

R

name	**salary**	**expenses**
Carrie Fisher	1234567	99999999
Mark Hamill	1234567	1000000
John Connor	1	99999999
Julius Caeser	99999999	1

Select (σ starts with an "s" sound, just like "select".)

σ allows us to filter out tuples from a relation when they do not match the predicate.

R

name	salary	expenses
Carrie Fisher	1234567	99999999
Mark Hamill	1234567	1000000
John Connor	1	99999999
Julius Caeser	99999999	1

We can create a new relation containing only the tuples where the salary is greater than the expenses.

Select (σ starts with an "s" sound, just like "select".)

σ allows us to filter out tuples from a relation when they do not match the predicate.

R

name	**salary**	**expenses**
Carrie Fisher	1234567	99999999
Mark Hamill	1234567	1000000
John Connor	1	99999999
Julius Caeser	99999999	1

We can create a new relation containing only the tuples where the salary is greater than the expenses.

$\sigma_{\text{expenses}<\text{salary}}(\text{R})$

name	**salary**	**expenses**
Mark Hamill	1234567	1000000
Julius Caeser	99999999	1

Select (σ starts with an “s” sound, just like “select”.)

σ allows us to filter out tuples from a relation when they do not match the predicate.

R

name	**salary**	**expenses**
Carrie Fisher	1234567	99999999
Mark Hamill	1234567	1000000
John Connor	1	99999999
Julius Caeser	99999999	1

We can create a new relation containing only the tuples where the salary is equal to 1.

Select (σ starts with an "s" sound, just like "select".)

σ allows us to filter out tuples from a relation when they do not match the predicate.

R

name	salary	expenses
Carrie Fisher	1234567	99999999
Mark Hamill	1234567	1000000
John Connor	1	99999999
Julius Caeser	99999999	1

We can create a new relation containing only the tuples where the salary is equal to 1.

$\sigma_{\text{salary}=1}(\text{R})$

name	salary	expenses
John Connor	1	99999999

Select (σ starts with an "s" sound, just like "select".)

σ allows us to filter out tuples from a relation when they do not match the predicate.

R

name	**salary**	**expenses**
Carrie Fisher	1234567	99999999
Mark Hamill	1234567	1000000
John Connor	1	99999999
Julius Caeser	99999999	1

We can create a new relation containing only the tuples where the expenses are 0.

Select (σ starts with an "s" sound, just like "select".)

σ allows us to filter out tuples from a relation when they do not match the predicate.

R

name	salary	expenses
Carrie Fisher	1234567	99999999
Mark Hamill	1234567	1000000
John Connor	1	99999999
Julius Caeser	99999999	1

We can create a new relation containing only the tuples where the expenses are 0.

$\sigma_{\text{expenses}=0}(\text{R})$

name	salary	expenses

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.

R

A	B
1	2
3	4

S

B	C	D
2	5	6
4	7	8
9	1	2

Product

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

R

A	B
1	2
3	4

S

B	C	D
2	5	6
4	7	8
9	1	2

$R \times S$

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

Product

If these were "just" sets then the cartesian product would give us:

$$\{(1,2),(3,4)\} \times \{(2,5,6),(4,6,8),(9,1,7)\} =$$

$$\begin{aligned}
\{&((1,2),(2,5,6)),\\
&((1,2),(4,6,8)),\\
&((1,2),(9,1,7)),\\
&((3,4),(2,5,6)),\\
&((3,4),(4,6,8)),\\
&((3,4),(9,1,7))\}
\end{aligned}$$

Product

If these were "just" sets then the cartesian product would give us:

$$\{(1,2),(3,4)\} \times \{(2,5,6),(4,6,8),(9,1,7)\} =$$

$$\begin{aligned}
\{&((1,2),(2,5,6)),\\
&((1,2),(4,6,8)),\\
&((1,2),(9,1,7)),\\
&((3,4),(2,5,6)),\\
&((3,4),(4,6,8)),\\
&((3,4),(9,1,7))\}
\end{aligned}$$

Product

If these were "just" sets then the cartesian product would give us:

$$\{(1,2),(3,4)\} \times \{(2,5,6),(4,6,8),(9,1,7)\} =$$

$$\begin{aligned}
\{&((1,2),(2,5,6)),\\
&((1,2),(4,6,8)),\\
&((1,2),(9,1,7)),\\
&((3,4),(2,5,6)),\\
&((3,4),(4,6,8)),\\
&((3,4),(9,1,7))\}
\end{aligned}$$

but since we are dealing with relations, our definition of the product smooshes the tuples together.

Product

If these were "just" sets then the cartesian product would give us:

$$\{(1,2),(3,4)\} \times \{(2,5,6),(4,6,8),(9,1,7)\} =$$

$$\begin{aligned}\{&((1,2,2,5,6)),\\ &((1,2,4,6,8)),\\ &((1,2,9,1,7)),\\ &((3,4,2,5,6)),\\ &((3,4,4,6,8)),\\ &((3,4,9,1,7))\}\end{aligned}$$

but since we are dealing with relations, our definition of the product smooshes the tuples together.

Qualified Attribute Names

In the product we had attributes named **R.B** and **S.B**. Why?

Qualified Attribute Names

In the product we had attributes named **R.B** and **S.B**. Why?

Without the relation names, the attribute names would have been ambiguous. Whenever attribute names would be ambiguous, we prepend the relation name to the attribute name.

Qualified Attribute Names

In the product we had attributes named **R.B** and **S.B**. Why?

Without the relation names, the attribute names would have been ambiguous. Whenever attribute names would be ambiguous, we prepend the relation name to the attribute name.

Every attribute "knows" what its relation is, but we only write it out when we must.

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 ($\bowtie$)

R

A	B
1	2
3	4

S

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

Example: Natural Join ($\bowtie$)

R

A	B
1	2
3	4

S

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

$R \bowtie S$

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

Example: Natural Join ($\bowtie$)

R

A	**B**
1	2
3	4

S

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

R $\bowtie$ S

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

Because R.**B** and S.**B** will always have the same values we will usually smoosh them together.

Example: Natural Join ($\bowtie$)

R

A	B
1	2
3	4

S

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

R $\bowtie$ S

A	B	C	D
1	2	5	6
3	4	7	8

Because R.**B** and S.**B** will always have the same values we will usually smoosh them together.

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:

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