



CS 450

Relational Algebra 1

Prof. Jessica Lin



Relational Query Languages

- Query languages: Allow manipulation and **retrieval of data** from a database.
- Relational model supports simple, powerful QLs:
 - Strong formal foundation based on logic.
 - Allows for much optimization.
- Query Languages **!=** programming languages!
 - QLs not expected to be “Turing complete”.
 - QLs not intended to be used for complex calculations.
 - QLs support easy, efficient access to large data sets.

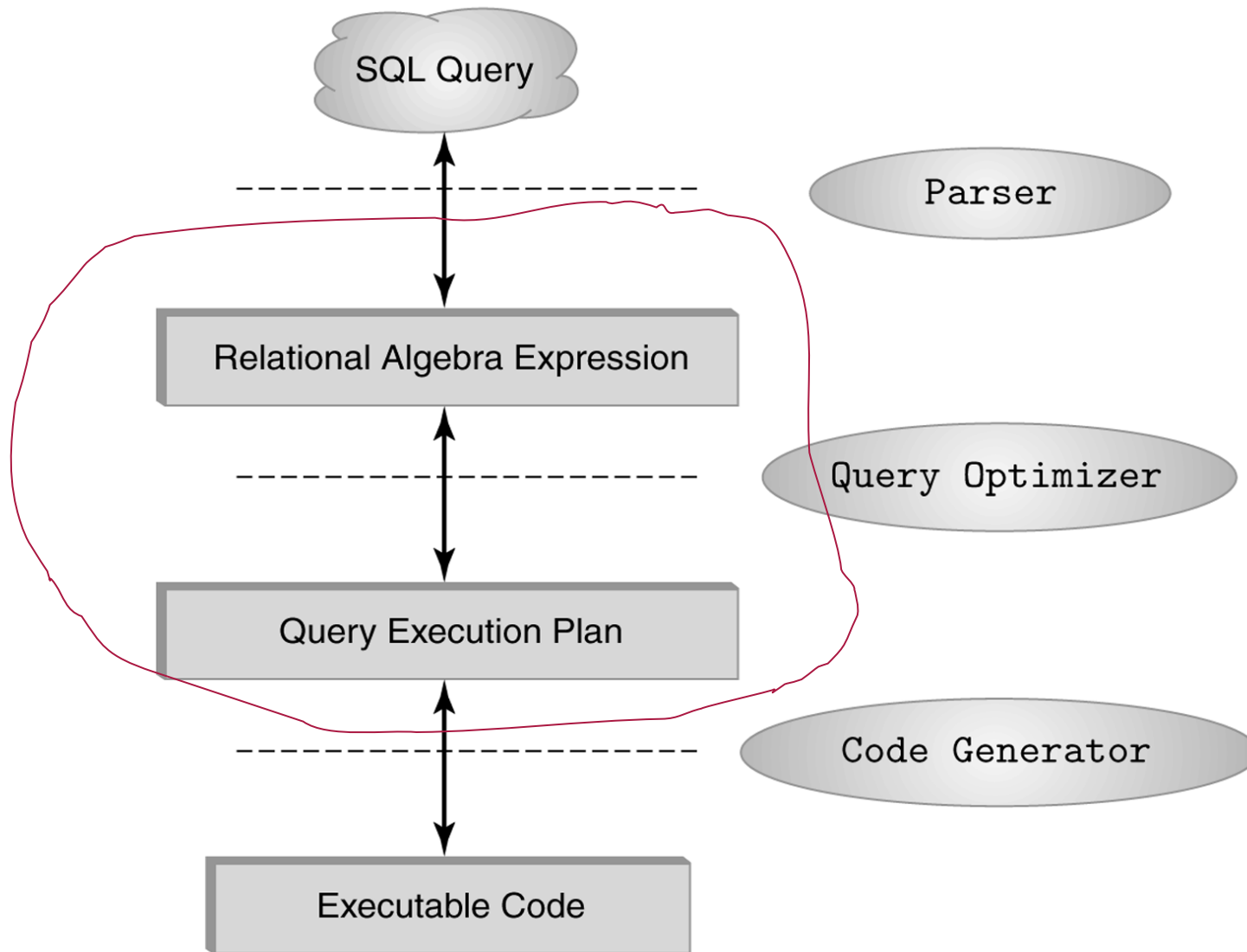
Formal Relational Query Languages

Two mathematical Query Languages form the basis for “real” languages (e.g. SQL), and for implementation:

- ① Relational Algebra: More **operational**, very useful for representing execution plans.
- ② Relational Calculus: Lets users describe what they want, rather than how to compute it. (**Non-operational**, declarative.)

👉 *Understanding Algebra is key to understanding SQL, and query processing!*

The Role of Relational Algebra in a DBMS



Algebra Preliminaries

- A query is applied to *relation instances*, and the result of a query is also a relation instance.
 - *Schemas of input* relations for a query are **fixed** (but query will run regardless of instance!)
 - The **schema for the result** of a given query is also **fixed!** Determined by definition of query language constructs.

Relational Algebra

- Procedural language
- Five basic operators

SQL is closely based on relational algebra.

- **selection**

select

- **projection**

project

- **union**

(why no intersection?)

- **set difference**

difference

- **Cross product**

Cartesian product

- There are some other operators which are composed of the above operators. These show up so often that we give them special names.
- The operators take one or two relations as inputs and give a new relation as a result.

Select Operation – Example

- Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

Intuition: The **select** operation allows us to retrieve some rows of a relation (by “some” I mean anywhere from none of them to all of them)

Here I have retrieved all the rows of the relation r where the value in field A equals the value in field B , and the value in field D is greater than 5.

- $\sigma_{A=B \wedge D > 5}(r)$

lowercase
Greek sigma

A	B	C	D
α	α	1	7
β	β	23	10

Select Operation

- Notation: $\sigma_p(r)$ lowercase Greek sigma σ
- p is called the **selection** predicate
- Defined as:

$$\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$$

Where p is a formula in propositional calculus consisting of terms connected by : \wedge (**and**), \vee (**or**), \neg (**not**)

Each term is one of:

$$\langle \text{attribute} \rangle \text{ op} \quad \langle \text{attribute} \rangle \text{ or } \langle \text{constant} \rangle$$

where op is one of: $=, \neq, >, \geq, <, \leq$

- Example of selection:

$$\sigma_{name='Lee'}(professor)$$

Project Operation – Example I

- Relation r :

A	B	C
α	10	7
α	20	1
β	30	1
β	40	2

Intuition: The **project** operation allows us to retrieve some columns of a relation (by “some” I mean anywhere from none of them to all of them)

- $\pi_{A,C}(r)$

A	C
α	7
α	1
β	1
β	2

Here I have retrieved columns **A** and **C**.

Greek alphabet pi.

Note: The textbook uses upper case Π .

Project Operation – Example II

- Relation r :

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

Intuition: The project operation removes duplicate rows, since relations are sets.

- $\pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Here there are two rows with $A = \alpha$ and $C = 1$. So one was discarded.

Project Operation

- Notation:

$$\pi_{A_1, A_2, \dots, A_k}(r) \quad \text{Greek alphabet pi}$$

where A_1, A_2 are attribute names and r is a relation name.

- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets.

Union Operation – Example

Relations r, s :

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

$r \cup s$:

A	B
α	1
α	2
β	1
β	3

Intuition: The **union** operation concatenates two relations, and removes duplicate rows (since relations are sets).

Here there are two rows with $A = \alpha$ and $B = 2$. So one was discarded.

Union Operation

- Notation: $r \cup s$
- Defined as:

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

For $r \cup s$ to be valid, r and s need to be “compatible.”

1. r, s must have the *same arity* (same number of attributes)
2. The attribute domains must be *the same* (e.g., 2nd column of r deals with the same type of values as does the 2nd column of s).

Although the field types must be the same, the names can be different. For example I can union *professor* and *lecturer* where:

professor(*PID* : string, *name* : string)

lecturer(*LID* : string, *first_name* : string)

Related Operation: Intersection

Relations r, s :

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- Similar to Union operation.

- But Intersection is NOT one of the five basic operations.

$r \cap s$:

A	B
α	2

- Intuition: The **intersection** operation computes the common rows between two relations

Set Difference Operation – Example

Relations r, s :

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

$r - s$:

A	B
α	1
β	1

Intuition: The **set difference** operation returns all the rows that are in r but not in s .

Set Difference Operation

- Notation $r - s$

- Defined as:

$$r - s = \{t \mid t \in r \text{ and } t \notin s\}$$

- Set differences must be taken between *compatible* relations.
 - r and s must have the *same arity*
 - attribute domains of r and s must be compatible
- Note that in general $r - s \neq s - r$

Cross (Cartesian)-Product Operation: Example

Relations r, s :

A	B
---	---

α	1
β	2

r

C	D	E
---	---	---

α	10	a
β	10	a
β	20	b
γ	10	b

S

$r \times S$:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Intuition: The **cross product** operation returns all possible combinations of rows in r with rows in S .

In other words the result is every possible pairing of the rows of r and S .

Cross (Cartesian)-Product Operation

- Notation $r \times s$
- Defined as:

$$r \times s = \{t \ q \mid t \in r \text{ and } q \in s\}$$

- Assume that attributes of $r(R)$ and $s(S)$ are disjoint. (That is, $R \cap S = \emptyset$).
- If attributes names of $r(R)$ and $s(S)$ are not disjoint, then renaming must be used.

Composition of Operations

- We can build expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

$r \times s$:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

A	B	C	D	E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b

“take the cross product of r and s , then return only the rows where A equals C ”

$\sigma_{A=C}(r \times s) \rightarrow$

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.

Example:

$$\rho_{myRelation} (r - s)$$

Renaming columns (rename A to A2):

$$\rho_{myRelation(A \rightarrow A2)} (r - s)$$

Take the set difference of r and s , and call the result $myRelation$
Renaming in relational algebra is essentially the same as assignment in a programming language

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	1
β	1

$myRelation$

Rename Operation

If a relational-algebra expression Y has arity n , then

$$\rho_{X(A \rightarrow A1, B \rightarrow A2, \dots)} Y$$

returns the result of expression Y under the name X , and with the attributes renamed to $A1, A2, \dots, An$.

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

S

For example,

$$\rho_{myRelation(A \rightarrow E, B \rightarrow K)} (r - s)$$

Take the set difference of r and s , and call the result *myRelation*, while renaming the first field to E , and the second field to K .

E	K
α	1
β	1

myRelation

Sailors Example

Sailors(sid, sname, rating, age)

Boats(bid, bname, color)

Reserves(sid, bid, day)

Example Instances

- “Sailors” and “Reserves” relations for our examples.

R1

<u>sid</u>	<u>bid</u>	<u>day</u>
22	101	10/10/96
58	103	11/12/96

S1

<u>sid</u>	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Algebra Operations

- Look what we want to get from the following table:

S2

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Selection

- Selects rows that satisfy *selection condition*.
- No duplicates in result! (Why?)
- *Schema* of result identical to schema of (only) input relation.

S2

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$$\sigma_{rating > 8}(S2) =$$

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

Projection

- Deletes attributes that are not in *projection list*.
- *Schema* of result contains exactly the fields in the projection list, with the same names that they had in the (only) input relation.
- Projection operator has to eliminate *duplicates!* (Why??)
 - Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it.

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

$\pi_{sname, rating}(S2)$

age
35.0
55.5

$\pi_{age}(S2)$

Composition of Operations

- *Result* relation can be the *input* for another relational algebra operation! (*Operator composition*)

S2

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$$\pi_{sname, rating}(\sigma_{rating > 8}(S2)) =$$

sname	rating
yuppy	9
rusty	10

What do we want to get from two relations?

R1

<u>sid</u>	<u>bid</u>	<u>day</u>
22	101	10/10/96
58	103	11/12/96

S1

<u>sid</u>	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

What about: Who reserved boat 101?

Or: Find the name of the sailor who reserved boat 101.

Cross-Product

- Each row of S1 is paired with each row of R1.
- *Result schema* has one field per field of S1 and R1, with field names inherited.

sid1	sname	rating	age	sid2	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

➔ Renaming operator (because of naming conflict):

$$\rho_{S1_New(sid \rightarrow sid1)}(S1) \times \rho_{R1_New(sid \rightarrow sid2)}(R1)$$



Why does this cross product help

Query: Find the name of the sailor who reserved boat 101.

Why does this cross product help

Query: Find the name of the sailor who reserved boat 101.

$$Temp \leftarrow \rho_{S1_New(sid \rightarrow sid1)}(S1) \times \rho_{R1_New(sid \rightarrow sid2)}(R1)$$

$$Result = \pi_{Sname}(\sigma_{sid1=sid2 \wedge bid=101}(Temp))$$

Note my use of “temporary” relation Temp using the assignment operator \leftarrow .

Why does this cross product help

Query: Find the name of the sailor who reserved boat 101.

$$\text{Result} = \pi_{Sname} (\sigma_{S1.sid=R1.sid \wedge R1.bid=101} (S1 \times R1))$$

Now use the “dot” notation to distinguish the attribute sid without renaming.

Another example

- Find the name of the sailor having the highest rating.

$AllR \leftarrow \pi_{rating} S2$

$Result? = \pi_{Sname} (\sigma_{S2.rating < AllR.rating} (S2 \times AllR))$

What's in "Result?" ?

Does it answer our query?

S2

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

AllR

rating
9
8
5
10

sid	sname	S2.rating	age	AllR.rating
28	yuppy	9	35.0	9
28	yuppy	9	35.0	8
28	yuppy	9	35.0	5
28	yuppy	9	35.0	10
31	lubber	8	55.5	9
31	lubber	8	55.5	8
31	lubber	8	55.5	5
31	lubber	8	55.5	10
44	guppy	5	35.0	9
44	guppy	5	35.0	8
44	guppy	5	35.0	5
44	guppy	5	35.0	10
58	rusty	10	35.0	9
58	rusty	10	35.0	8
58	rusty	10	35.0	5
58	rusty	10	35.0	10

$AllR \leftarrow \pi_{rating} S2$

Result? = $\pi_{Sname} (\sigma_{S2.rating < AllR.rating} (S2 \times AllR))$

Union, Intersection, Set-Difference

- All of these operations take two input relations, which must be *compatible*:
 - Same number of fields.
 - ‘Corresponding’ fields have the same type.
- What is the *schema* of result?

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

$S1 \cup S2$

sid	sname	rating	age
22	dustin	7	45.0

$S1 - S2$

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

$S1 \cap S2$

Back to our query

- Find the name of the sailor having the highest rating.

$$AllR \leftarrow \pi_{rating} S2$$

$$Tmp \leftarrow \pi_{Sid, Sname} (\sigma_{S2.rating < AllR.rating} (S2 \times AllR))$$

$$Result = \pi_{Sname} (\pi_{Sid, Sname} (S2) - Tmp)$$

* Why not project on Sid only for Tmp?

Relational Algebra (Summary)

- Basic operations:
 - Selection (σ) Selects a subset of rows from relation.
 - Projection (π) Deletes unwanted columns from relation.
 - Cross-product (\times) Allows us to combine two relations.
 - Set-difference ($-$) Tuples in reln. 1, but not in reln. 2.
 - Union (\cup) Tuples in reln. 1 and in reln. 2.

Also,

- Rename (ρ) Changes names of the attributes
- Since each operation returns a relation, **operations can be composed!** (Algebra is “closed”.)
- Use of temporary relations recommended.