# SQL Queries — misc. and relational algebra

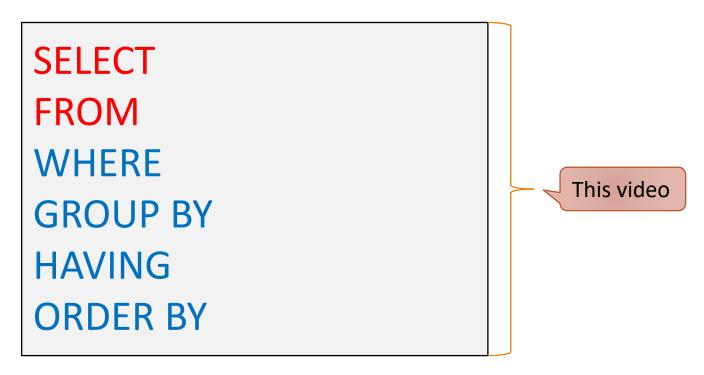
#### Overview of this video

Provide a better understanding of SQL queries as a whole, instead of as parts

(Relational algebra does focus on parts, but should let you better see how the parts compose, which is the point)

## SQL Queries

Queries in SQL have the following form:



## SQL Queries

Queries in SQL have the following form:



## Views: as saved queries

Intuitively, saved queries or virtual tables

SELECT first\_name, e\_id, COUNT(t\_id)
FROM Employees NATURAL JOIN Transactions
GROUP BY first\_name, e\_id;

first_name	e_id	COUNT(t_id)
Anne	1	2
William	3	1

#### **Employees**

birthday	first_name	family_name	e_id
1990-11-10	Anne	Smith	1
2000-02-05	David	Jones	2
1995-05-09	William	Taylor	3

#### **Transactions**

t_id	c_id	e_id
1	3	1
2	6	1
3	19	3

## Views: as saved queries

Intuitively, saved queries or virtual tables

CREATE VIEW Employee\_transaction\_count AS
SELECT first\_name, e\_id, COUNT(t\_id)
FROM Employees NATURAL JOIN Transactions
GROUP BY first\_name, e\_id;

#### **Employee\_transaction\_count**

first_name	e_id	COUNT(t_id)
Anne	1	2
William	3	1

#### **Employees**

birthday	first_name	family_name
1990-11-10	Anne	Smith
2000-02-05	David	Jones
1995-05-09	William	Taylor

#### **Transactions**

t_id	c_id	e_id
1	3	1
2	6	1
3	19	3

#### Views: as virtual tables

Intuitively, saved queries or virtual tables

CREATE VIEW Employee\_transaction\_count AS SELECT first\_name, e\_id, COUNT(t\_id) FROM Employees NATURAL JOIN Transactions GROUP BY first\_name, e\_id;

SELECT first\_name, COUNT(t\_id)
FROM Employee\_transaction\_count;



**Employee\_transaction\_count** 

first_name	COUNT(t_id)
Anne	2
William	1

#### More on views

By default data in views can't be modified (i.e. updated, delete from or inserted into)

Can be done with triggers, but that is not covered in this module

Views can be modified though:

CREATE OR REPLACE VIEW Employee\_transaction\_count AS SELECT first\_name, COUNT(t\_id) FROM Employees NATURAL JOIN Transactions GROUP BY first\_name, e\_id;

CREATE VIEW Employee\_transaction\_count AS
SELECT first\_name, e\_id, COUNT(t\_id)
FROM Employees NATURAL JOIN Transactions
GROUP BY first\_name, e\_id;

Or removed:

**DROP VIEW Employee\_transaction\_count;** 

## Relational Algebra

## Algebra

Algebra = branch of math

Example: Algebra with numbers

- Addition (written: +) = function that takes two numbers and returns a number
- Logarithm (written: log) = function that takes one number and returns another
  - Has a subscript the base

0

Can be composed, e.g.:  $\log_2 (3 + 8) + 4$ 

## Relational algebra

Algebra with tables ≈ SQL SELECT queries

Exception: uses set semantics (i.e. removes duplicates and has no order)

Relational algebra is crucial for optimization

Later in the course

We have seen what tables are (i.e., the thing that in relational algebra corresponds to numbers in algebra with numbers) and we will next see 5 relational algebra functions (i.e. the thing that corresponds to + or log from algebra with numbers)

## Projection $(\pi)$

 $\pi_{\text{attribute list}}(\mathbf{R})$  = restricts **R** to the attributes in attribute list

**SQL** query

#### SELECT DISTINCT

DISTINCT, due to sets

FROM Employees;

family\_name, birthday

#### **Employees**

birthday	first_name	family_name
1990-11-10	Anne	Smith
2000-02-05	David	Jones
1995-05-09	William	Taylor

translates into

 $\pi_{\text{family\_name,birthday}}(\text{Employees})$ 

family_name	birthday
Smith	1990-11-10
Jones	2000-02-05
Taylor	1995-05-09

## Renaming $(\rho)$

$$\rho_{A1 \rightarrow B1,A2 \rightarrow B2,...}(R)$$
 = renames attribute A1 to B1, attribute A2 to B2, ...

SQL query

SELECT DISTINCT birthday AS bday, first\_name, family\_name FROM Employees;

**Employees** 

birthday	first_name	family_name
1990-11-10	Anne	Smith
2000-02-05	David	Jones
1995-05-09	William	Taylor

	•	translates into	
	$ ho_{ m birthday}$	<sub>ay</sub> (Employees)	<b>-</b>
Relational algebra expression			'

bday	first_name	family_name
1990-11-10	Anne	Smith
2000-02-05	David	Jones
1995-05-09	William	Taylor

Selection  $(\sigma)$ 

The SELECT keyword in SQL has nothing to do with the selection operator!

 $_{ondition}(R)$  = set of all tuples in R that satisfy the condition

SQL query

**SELECT DISTINCT \*** 

**FROM Employees** 

WHERE first\_name='Anne';

**Employees** 

birthday	first_name	family_name	
1990-11-10	Anne	Smith	
2000-02-05	David	Jones	
1995-05-09	William	Taylor	

translates into

 $\sigma_{\text{first\_name='Anne'}}$  (Employees)

birthday	first_name	family_name		
1990-11-10	Anne	Smith		

## Cartesian Product (X)

 $R_1 \times R_2$  = pairs each tuple in  $R_1$  with each tuple in  $R_2$ 

**SQL** query

**SELECT DISTINCT \* FROM Modules, Lecturers**;

#### **Modules**

code	year	sem
COMP105	1	1
COMP201	2	1

#### Lecturers

name	module		
John	COMP105		
Sebastian	COMP201		

translates into

**Modules** × **Lecturers** 

	code	year	sem	name	module
	COMP105	1	1	John	COMP105
>	COMP105	1	1	Sebastian	COMP201
	COMP201	2	1	John	COMP105
	COMP201	2	1	Sebastian	COMP201

## Natural Join (⋈)

 $R_1 \bowtie R_2$  = pairs each tuple in  $R_1$  with each tuple in  $R_2$  with matching common attributes

SELECT DISTINCT \*
FROM Employees NATURAL JOIN
Transactions;

#### **Employees**

birthday	first_name	family_name	e_id
1990-11-10	Anne	Smith	1
2000-02-05	David	Jones	2
1995-05-09	William	Taylor	3

#### **Transactions**

t_id	c_id	e_id
1	3	1
2	6	1
3	19	3

translates into

**Employees** ⋈ **Transactions** 

birthday	first_name	family_ name	e_id	t_id	c_id
1990-11-10	Anne	Smith	1	1	3
1990-11-10	Anne	Smith	1	2	6
1995-05-09	William	Taylor	3	3	19

## Many others...

Besides the mentioned, there are many others, e.g. for:

- 1. Left/right semi-join ( $\bowtie/\bowtie$ )
  - I.e. returns the rows from the left/right table that would be matched to a row on the other side in a Natural Join – it was covered in SQL queries - optional part

#### GROUP BY

 Relational Algebra for GROUP BY is complex and the two books differ in how they define it (different syntax, symbols and so on) and overall, I feel it is sufficient to know that you CAN define it (both definitions writes down what the GROUP BY statements says and the various aggregates used) and leave it at that

## Key understanding

The goal of showing you relational algebra now (instead of next to optimization) is that understanding how you can combine the different parts is easier:

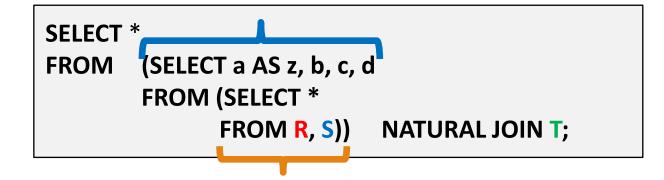
You can do it however you want!

Math example from earlier:

$$\log_2(3+8)+4$$

Relational algebra example:

$$\circ \rho_{a \to z}(R \times S) \bowtie T$$



Say the schema for R was R(a,b), for S was S(c,d) and for T was T(e,z)