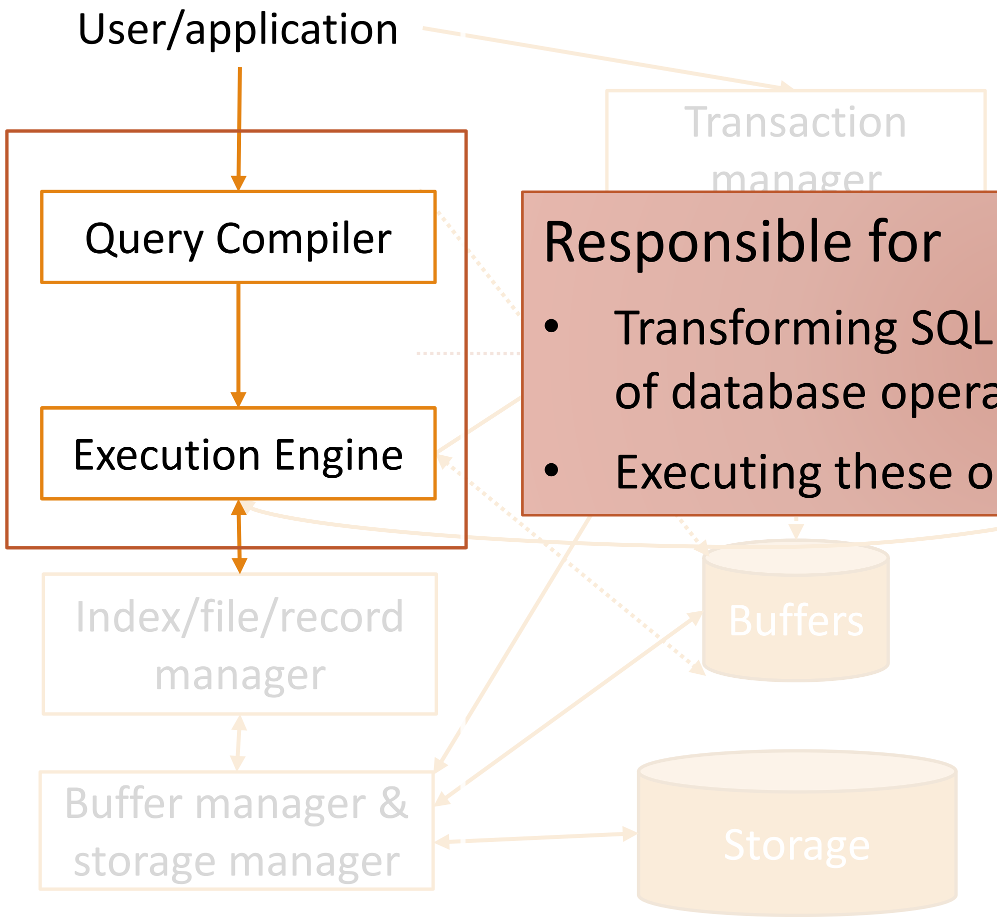
Introduction to query processing

Overview over this video

This video is meant to serve as an introduction to query processing AND also how and what a query plan is (basically, a way to look at relational algebra)

Where we are…



Responsible for

- Transforming SQL queries into sequences of database operations
- Executing these operations

SQL Queries

Here typically
SELECT statements

SQL query: SQL SELECT/INSERT/UPDATE/DELETE statement

```
SELECT id AS employee_id
FROM Employees
WHERE department = ’Liv’;
```

Employees(name, id, departmnet)

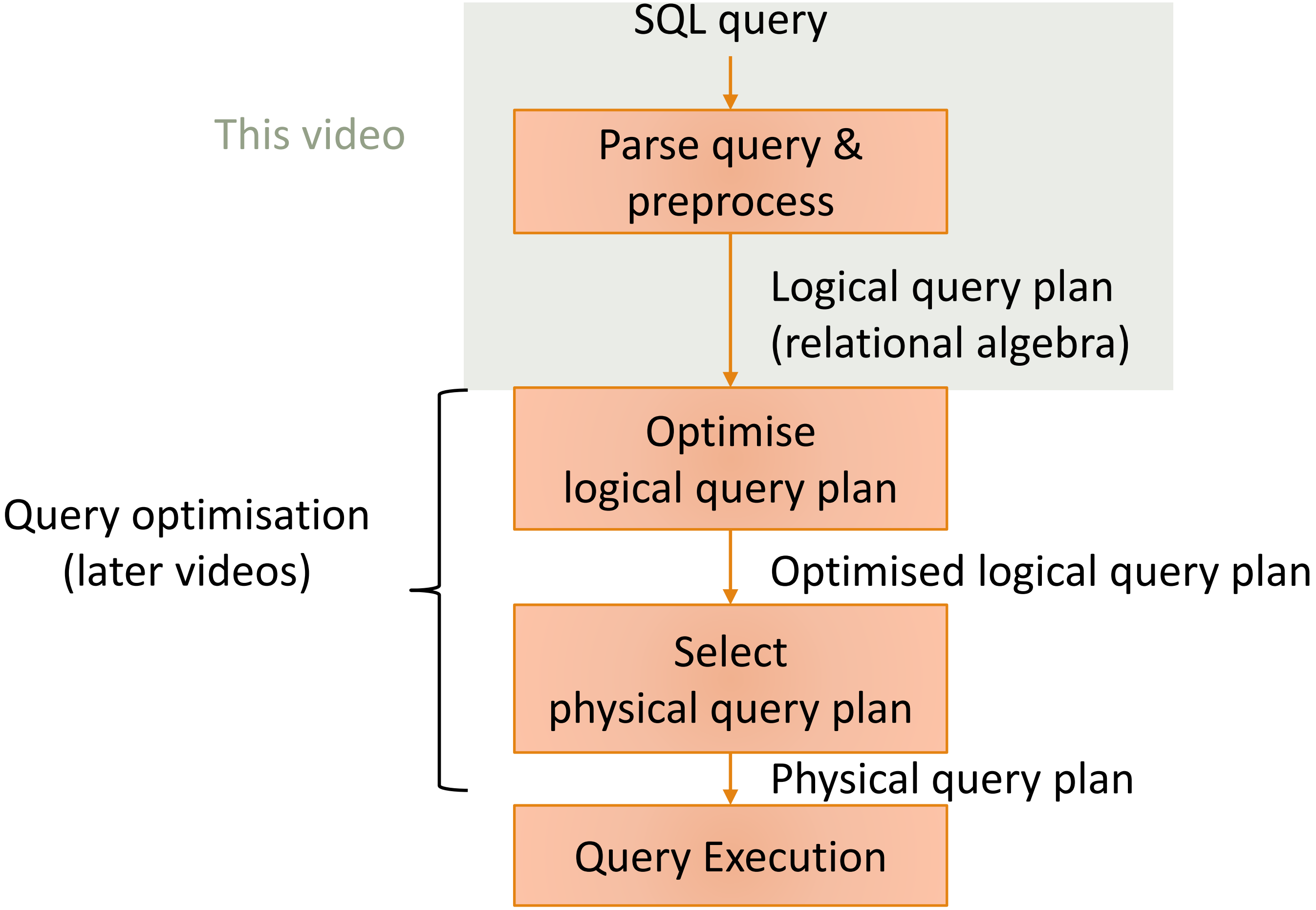
Salary(e_id, month, amount)

```
SELECT name, sum(amount)
FROM Employees, Salary
WHERE id = e_id AND department = ‘Liv’
GROUP BY name;
```

Declarative: tells the DBMS **what we want**, ***not* how to get it**

DBMS selects a good sequence of database operations
to execute the query

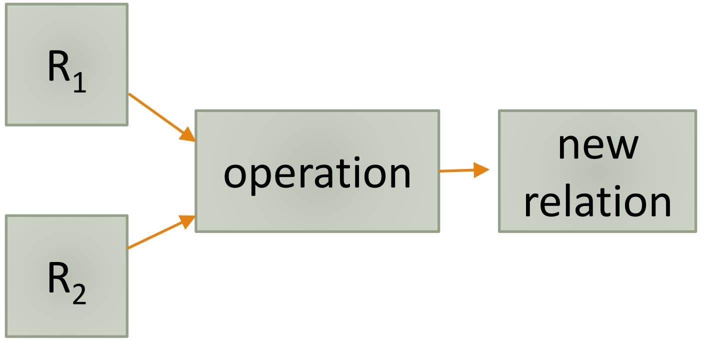
How Does Query Processing Work?



SQL query
This video
Parse query & preprocess
Logical query plan (relational algebra)
Optimise logical query plan
Query optimisation (later videos)
Optimised logical query plan
Select physical query plan
Physical query plan
Query Execution

Relational Algebra

Set of **operations** that can be applied to **relations** to **compute new relations**



Basic relational algebra:

- Selection (σ)
- Projection (π)
- Cartesian product (×)
- Union (∪)
- Renaming (ρ)
- Natural join (⋈)
- Semijoin (⋉)
- Many others can be defined

If you are unsure what any of these means, go back and watch the SQL query – Misc video (and the second tutorial for Union (∪))

Query Plans

A **relational algebra expression** that is obtained from an SQL query is also called a **(logical) query plan**

```
SELECT department, name
FROM Stores, Employees
WHERE department=worksAt AND city='Liverpool';
```

SQL query

↓

$$\pi_{\text{department,name}}(\sigma_{\text{department=worksAt AND city='Liverpool'}}(\text{Stores} \times \text{Employees}))$$

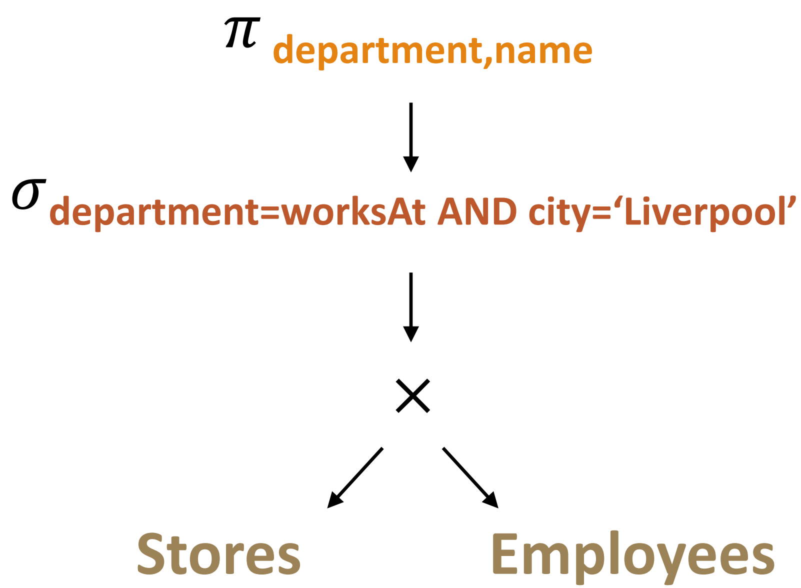
Query plan for the query

Query plans are typically **represented as trees**

Query Plans As Trees

$$\pi_{\text{department,name}}(\sigma_{\text{department=worksAt AND city='Liverpool'}}(\text{Stores} \times \text{Employees}))$$

Tree representation:



$\pi_{\text{department,name}}$

$\sigma_{\text{department=worksAt AND city='Liverpool'}}$

$\times$

Stores

Employees

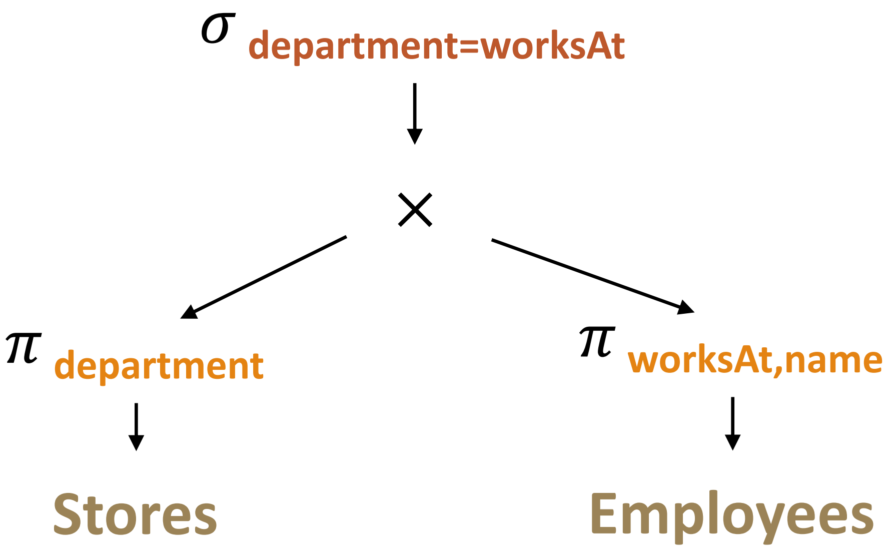
- Leaves = input relations
- Inner nodes = operators

- Such trees are evaluated from the leaves to the root

Another example

Represent the following query plan as a tree:

$$\sigma_{\text{department=worksAt}}(\pi_{\text{department}}(\text{Stores}) \times \pi_{\text{worksAt,name}}(\text{Employees}))$$



$\sigma_{\text{department=worksAt}}$

$\times$

$\pi_{\text{department}}$

$\pi_{\text{worksAt,name}}$

Stores

Employees

Equivalent Query Plans

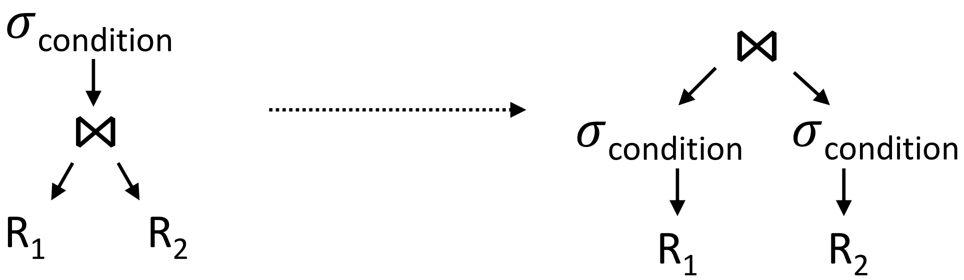
There are typically **many different query plans**

DBMSs aim to **select a best possible query plan**

Relational algebra is better suited than SQL for this

- Can use **equivalence laws** of relational algebra to generate a query plan for the same query that can be executed faster!
- Example:
  - $\sigma_{\text{condition}}(R_1 \bowtie R_2) = \sigma_{\text{condition}}(R_1) \bowtie \sigma_{\text{condition}}(R_2)$

Details will come later…



Summary

DBMS translate SQL queries into relational algebra expressions, also called **(logical) query plans**

The DBMS will then

- Optimise the logical query plan by using equivalence laws (later…)
- Select suitable algorithms for computing each operator in the logical query plan (later…)