# Executing a query plan naively

# Overview over this video

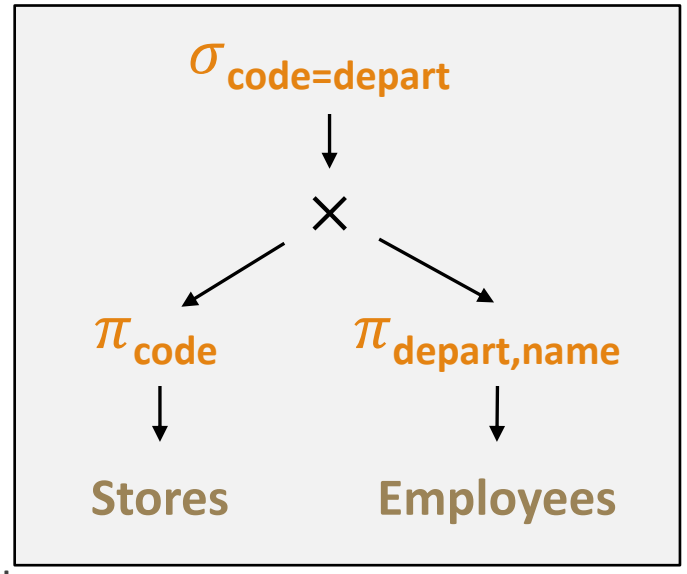
We will see a fairly naïve approach to executing a query plan – the next video will show a more advanced approach to some of it

# Executing Query Plans

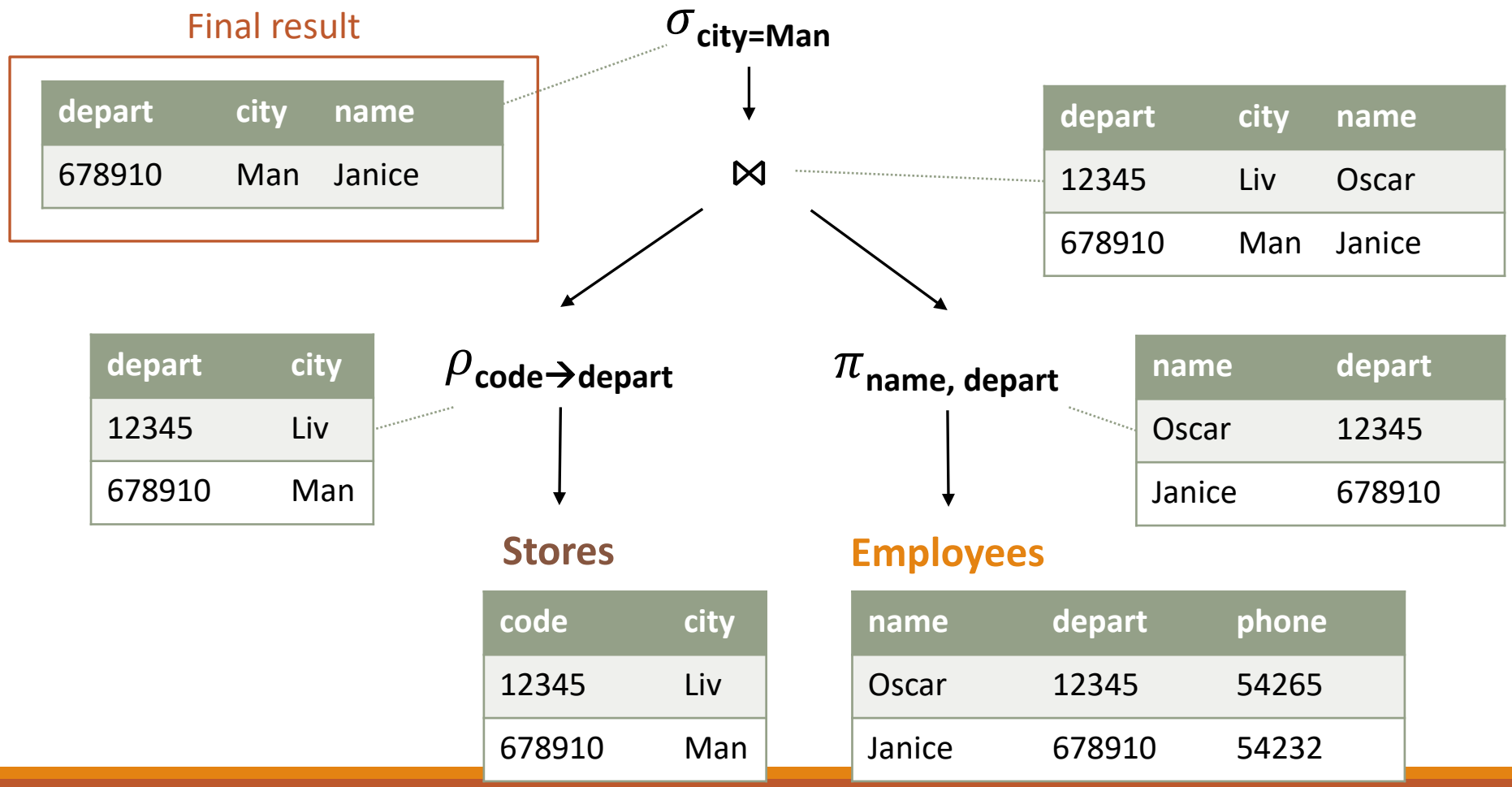
Query plans tells us exactly how to compute the result to a query

Proceed from bottom to top:

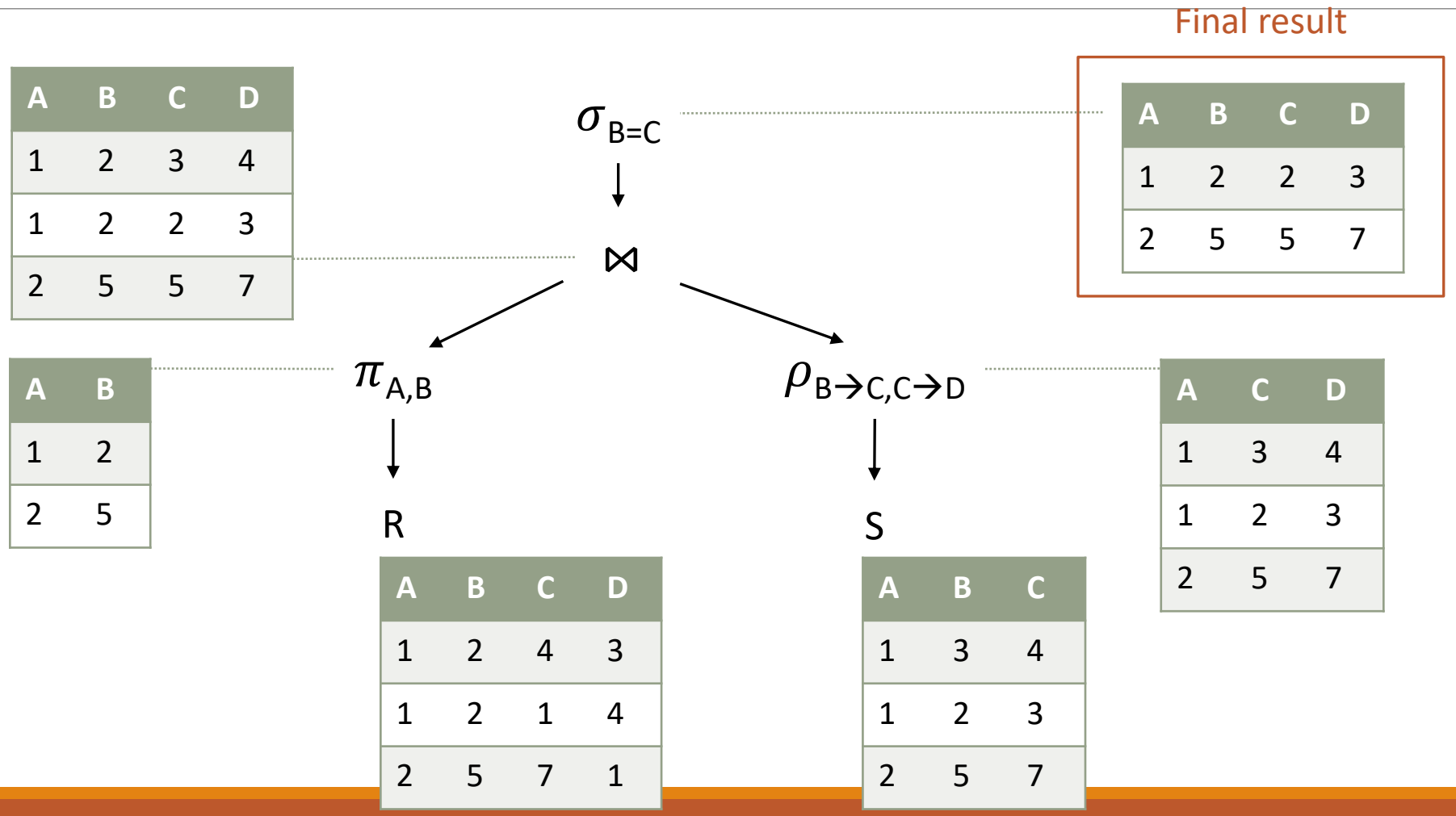
- Compute an **intermediate result** for each node
- For a **leaf** labeled with relation **R**, the intermediate result is **R**.
- For an **inner node** labeled with operator **op**, get the intermediate result by applying **op** to the childrens' intermediate results.
- **Result of the query** = intermediate result of the root



# Example


Final result
depart city name
678910 Man Janice
$\sigma_{city=Man}$
$\bowtie$
depart city name
12345 Liv Oscar
678910 Man Janice
depart city
12345 Liv
678910 Man
$\rho_{code \rightarrow depart}$
$\pi_{name, depart}$
name depart
Oscar 12345
Janice 678910
Stores
code city
12345 Liv
678910 Man
Employees
name depart phone
Oscar 12345 54265
Janice 678910 54232

# Example 2


Final result
A B C D
1 2 3 4
1 2 2 3
2 5 5 7
$\sigma_{B=C}$
$\bowtie$
A B C D
1 2 2 3
2 5 5 7
$\pi_{A,B}$
A B
1 2
2 5
$\rho_{B\rightarrow C,C\rightarrow D}$
A C D
1 3 4
1 2 3
2 5 7
R
A B C D
1 2 4 3
1 2 1 4
2 5 7 1
S
A B C
1 3 4
1 2 3
2 5 7

# How to ”Apply” An Operator?

How to compute $\sigma_{\text{condition}}(R)$?

R:

have to read entire file

```
for each tuple t in R:
    if t satisfies condition:
        output t
```

Is there a faster way? Sometimes, e.g. if it is sorted and the condition is “easy” or we we have an index on it

How to compute $\pi_{\text{attribute list}}(R)$?

Similar! Read **R** only once…

```
for each tuple t in R:
    output the restriction
        of t to the attributes
        in attribute list
```

Is there a faster way? Not really, if you actually do it – can do it lazily though

How to compute **R** ⋈ **S**?

Many different ways…

# Naïve Computation of Joins

Nested Loop Join Algorithm:

**Compute R ⋈ S:**

**for each** tuple **r** in **R**:
  **for each** tuple **s** in **S**:
    **if r** and **s** have the same values for all common attributes:
      **output r ⋈ s**

the tuple obtained by joining **r** and **s**

**Stores**

| depart | city |
|---|---|
| 12345 | Liv |
| 678910 | Man |

**r**

⋈

**Employees**

| name | depart |
|---|---|
| Oscar | 12345 |
| Janice | 678910 |
| David | 678910 |

**s**

=

| depart | city | name |
|---|---|---|
| 12345 | Liv | Oscar |
| | | |
| | | |

**r ⋈ s**

# Naïve Computation of Joins

Nested Loop Join Algorithm:

**Compute R ⋈ S:**

**for each** tuple **r** in **R**:
  **for each** tuple **s** in **S**:
    **if r** and **s** have the same values for all common attributes:
      **output r ⋈ s**

the tuple obtained by joining **r** and **s**

**Stores**

| depart | city |
|---|---|
| 12345 | Liv |
| 678910 | Man |

**r**

⋈

**Employees**

| name | depart |
|---|---|
| Oscar | 12345 |
| Janice | 678910 |
| David | 678910 |

**s**

=

| depart | city | name |
|---|---|---|
| 12345 | Liv | Oscar |
| | | |
| | | |

# Naïve Computation of Joins

Nested Loop Join Algorithm:

**Compute R ⋈ S:**

**for each** tuple **r** in **R**:
  **for each** tuple **s** in **S**:
    **if r** and **s** have the same values for all common attributes:
      **output r ⋈ s**

the tuple obtained by joining **r** and **s**

**Stores**

| depart | city |
|---|---|
| 12345 | Liv |
| 678910 | Man |

**r**

⋈

**Employees**

| name | depart |
|---|---|
| Oscar | 12345 |
| Janice | 678910 |
| David | 678910 |

**s**

=

| depart | city | name |
|---|---|---|
| 12345 | Liv | Oscar |
| | | |
| | | |

# Naïve Computation of Joins

Nested Loop Join Algorithm:

**Compute R ⋈ S:**

**for each** tuple **r** in **R**:
  **for each** tuple **s** in **S**:
    **if r** and **s** have the same values for all common attributes:
      **output r ⋈ s**

the tuple obtained by joining **r** and **s**

**Stores**

| depart | city |
|---|---|
| 12345 | Liv |
| 678910 | Man |

**r**

⋈

**Employees**

| name | depart |
|---|---|
| Oscar | 12345 |
| Janice | 678910 |
| David | 678910 |

**s**

=

| depart | city | name |
|---|---|---|
| 12345 | Liv | Oscar |
| | | |
| | | |

# Naïve Computation of Joins

Nested Loop Join Algorithm:

**Compute R ⋈ S:**

**for each** tuple **r** in **R**:
  **for each** tuple **s** in **S**:
    **if r** and **s** have the same values for all common attributes:
      **output r ⋈ s**

the tuple obtained by joining **r** and **s**

**Stores**

| depart | city |
|---|---|
| 12345 | Liv |
| 678910 | Man |

**r**

⋈

**Employees**

| name | depart |
|---|---|
| Oscar | 12345 |
| Janice | 678910 |
| David | 678910 |

**s**

=

| depart | city | name |
|---|---|---|
| 12345 | Liv | Oscar |
| | | |
| | | |

# Naïve Computation of Joins

Nested Loop Join Algorithm:

**Compute R ⋈ S:**

**for each** tuple **r** in **R**:
  **for each** tuple **s** in **S**:
    **if r** and **s** have the same values for all common attributes:
      **output r ⋈ s**

the tuple obtained by joining **r** and **s**

**Stores**

| depart | city |
|---|---|
| 12345 | Liv |
| 678910 | Man |

**r**

⋈

**Employees**

| name | depart |
|---|---|
| Oscar | 12345 |
| Janice | 678910 |
| David | 678910 |

**s**

=

| depart | city | name |
|---|---|---|
| 12345 | Liv | Oscar |
| 678910 | Man | Janice |
| | | |

**r ⋈ s**

# Naïve Computation of Joins

Nested Loop Join Algorithm:

**Compute R ⋈ S:**

**for each** tuple **r** in **R**:
  **for each** tuple **s** in **S**:
    **if r** and **s** have the same values for all common attributes:
      **output r ⋈ s**

the tuple obtained by joining **r** and **s**

**Stores**

| depart | city |
|---|---|
| 12345 | Liv |
| 678910 | Man |

**r**

⋈

**Employees**

| name | depart |
|---|---|
| Oscar | 12345 |
| Janice | 678910 |
| David | 678910 |

**s**

=

| depart | city | name |
|---|---|---|
| 12345 | Liv | Oscar |
| 678910 | Man | Janice |
| | | |

# Naïve Computation of Joins

Nested Loop Join Algorithm:

**Compute R ⋈ S:**

**for each** tuple **r** in **R**:
  **for each** tuple **s** in **S**:
    **if r** and **s** have the same values for all common attributes:
      **output r ⋈ s**

the tuple obtained by joining **r** and **s**

**Stores**

| depart | city |
|---|---|
| 12345 | Liv |
| 678910 | Man |

**r**

⋈

**Employees**

| name | depart |
|---|---|
| Oscar | 12345 |
| Janice | 678910 |
| David | 678910 |

**s**

=

| depart | city | name |
|---|---|---|
| 12345 | Liv | Oscar |
| 678910 | Man | Janice |
| 678910 | Man | David |

**r ⋈ s**

# Naïve Computation of Joins

Nested Loop Join Algorithm:

**Compute R ⋈ S:**

**for each** tuple **r** in **R**:
    **for each** tuple **s** in **S**:
        **if r** and **s** have the same values for all common attributes:
            **output r ⋈ s**

Slow: for each tuple **r** in **R** reads entire relation **S**

Running time: $O(|\mathbf{R}| \times |\mathbf{S}|)$

Number of tuples in **R**

Number of tuples in **S**

# Summary

Query plans are evaluated bottom-up – from the leaves to the root

Each operator can be computed in different ways

- Selection: e.g., linear scans (reading the relation once)
- Projection: linear scans
- Joins: Only Nested Loop Join in this video…