Optimization of Logical Queries

Task:

Consider the following relational schema:

- Emp(eid, did, sal, hobby)
- Dept(did, dname, floor, phone)
- Finance(did, budget, sales, expenses)

For the following SQL statement:

1. Translate the query into the relational algebra.
2. Remove redundant joins from the select-project-join subexpressions in the obtained logical query plan.
3. By means of the algebraic laws, further optimize the obtained expression.

Optimization of Logical Queries

Task (continued)

```
SELECT D.floor
FROM Dept D, Emp E
WHERE
 (D.floor = 1
  OR D.floor IN
    ( SELECT D2.floor FROM Dept D2, Finance F1
      WHERE  F1.budget > 150 AND D2.did = F1.did)
 )
 AND E.did = D.did
 AND E.did IN (SELECT F2.did FROM Finance F2, Emp E2
               WHERE F2.did = E.did AND E2.did = D.did
               AND E2.eid = E.eid AND F2.expenses = 300)
```

Optimization of Logical Queries

Solution: translation into the relational algebra

First, we normalize the query to a form with only EXISTS and NOT EXISTS subqueries:

```
SELECT D.floor
FROM Dept D, Emp E
WHERE
 (D.floor = 1 OR EXIST
    ( SELECT D2.floor FROM Dept D2, Finance F1
      WHERE  F1.budget > 150 AND D2.did = F1.did
      AND D2.floor = D.floor) )
 AND E.did = D.did
 AND EXISTS (SELECT F2.did FROM Finance F2, Emp E2
             WHERE F2.did = E.did AND E2.did = D.did
             AND E2.eid = E.eid AND F2.expenses = 300
             AND E.did = F2.did)
```

Optimization of Logical Queries

Conjunctive Normal Form

```
SELECT D.floor
FROM Dept D, Emp E
WHERE ( D.floor = 1
  AND E.did = D.did
  AND EXISTS (SELECT F2.did FROM Finance F2, Emp E2
   WHERE F2.did = E.did AND E2.did = D.did
   AND E2.eid = E.eid AND F2.expenses = 300 AND E.did = F2.did)
) OR (
  EXIST ( SELECT D2.floor FROM Dept D2, Finance F1
   WHERE  F1.budget > 150 AND D2.did = F1.did
   AND D2.floor = D.floor)
  AND E.did = D.did
  AND EXISTS (SELECT F2.did FROM Finance F2, Emp E2
   WHERE F2.did = E.did AND E2.did = D.did
   AND E2.eid = E.eid AND F2.expenses = 300 AND E.did = F2.did) )
```

Optimization of Logical Queries

Normalize to UNION

```
Q1 = SELECT D.floor
     FROM Dept D, Emp E
     WHERE D.floor = 1
      AND E.did = D.did
      AND EXISTS (SELECT F2.did FROM Finance F2, Emp E2
          WHERE F2.did = E.did AND E2.did = D.did
          AND E2.eid = E.eid AND F2.expenses = 300
          AND E.did = F2.did)
```

Optimization of Logical Queries

Normalize to UNION

```
Q2 = SELECT D.floor
FROM Dept D, Emp E
WHERE
 EXIST ( SELECT D2.floor FROM Dept D2, Finance F1
         WHERE  F1.budget > 150 AND D2.did = F1.did
         AND D2.floor = D.floor)
 AND E.did = D.did
 AND EXISTS (SELECT F2.did FROM Finance F2, Emp E2
             WHERE F2.did = E.did AND E2.did = D.did
             AND E2.eid = E.eid AND F2.expenses = 300
             AND E.did = F2.did)
```

The new query is Q1 `UNION` Q2.

Optimization of Logical Queries

Translation of the innermost subqueries

```
SELECT F2.did FROM Finance F2, Emp E2
          WHERE F2.did = E.did AND E2.did = D.did
          AND E2.eid = E.eid AND F2.expenses = 300
          AND E.did = F2.did
```

This subquery is translated as follows:

$$e_1 = \boldsymbol{\pi}_{F_2.\texttt{did},E.*,D.*}\boldsymbol{\sigma}_{F_2.\texttt{did}=E.\texttt{did}\wedge E_2.\texttt{did}=D.\texttt{did}\wedge E_2.\texttt{eid}=E.\texttt{eid}}$$
$$\boldsymbol{\sigma}_{F_2.\texttt{expenses}=300\wedge E.\texttt{did}=F_2.\texttt{did}}(\boldsymbol{\rho}_D(\texttt{Dept})\times\boldsymbol{\rho}_E(\texttt{Emp})\times\boldsymbol{\rho}_{F2}(\texttt{Finance})\times\boldsymbol{\rho}_{E2}(\texttt{Emp}))$$

Optimization of Logical Queries

Translation of the innermost subqueries

```
SELECT D2.floor FROM Dept D2, Finance F1
      WHERE  F1.budget > 150 AND D2.did = F1.did
      AND D2.floor = D.floor
```

This subquery is translated as follows:

$$e_2 = \boldsymbol{\pi}_{D_2.\texttt{floor},D.*}\boldsymbol{\sigma}_{F_1.\texttt{budget}>150 \wedge D_2.\texttt{did}=F_1.\texttt{did}}$$
$$\boldsymbol{\sigma}_{D_2.\texttt{floor}=D.\texttt{floor}}(\boldsymbol{\rho}_D(\texttt{Dept}) \times \boldsymbol{\rho}_{D2}(\texttt{Dept}) \times \boldsymbol{\rho}_{F1}(\texttt{Finance}))$$

Optimization of Logical Queries

Translation of the Middle Queries

```
Q1 = SELECT D.floor FROM Dept D, Emp E
      WHERE D.floor = 1 AND E.did = D.did
       AND EXISTS (SELECT F2.did FROM Finance F2, Emp E2
           WHERE F2.did = E.did AND E2.did = D.did
           AND E2.eid = E.eid AND F2.expenses = 300
           AND E.did = F2.did)
```

The translation of the from part gives

$$e_3 = (\boldsymbol{\rho}_D(\texttt{Dept}) \times \boldsymbol{\rho}_E(\texttt{Emp}))$$

To de-correlate we compute:

$$f = \hat{e_3} \bowtie \boldsymbol{\pi}_{D.*,E.*}(e_1)$$

Note that $\hat{e_3}$ is empty and hence

$$f = \boldsymbol{\pi}_{D.*,E.*}(e_1)$$

To this expression we add the WHERE and SELECT clause:

$$e_4 = \boldsymbol{\pi}_{D.\texttt{floor}}(\boldsymbol{\sigma}_{D.\texttt{floor}=1 \wedge E.did=D.did}(\boldsymbol{\pi}_{D.*,E.*}(e_1))$$

Optimization of Logical Queries

Translation of the Middle Queries

```
Q2 = SELECT D.floor
 FROM Dept D, Emp E
 WHERE
  EXIST ( SELECT D2.floor FROM Dept D2, Finance F1
          WHERE  F1.budget > 150 AND D2.did = F1.did
          AND D2.floor = D.floor)
  AND E.did = D.did
  AND EXISTS (SELECT F2.did FROM Finance F2, Emp E2
              WHERE F2.did = E.did AND E2.did = D.did
              AND E2.eid = E.eid AND F2.expenses = 300
              AND E.did = F2.did)
```

The translation of the from part gives

$$e_5 = (\boldsymbol{\rho}_D(\texttt{Dept}) \times \boldsymbol{\rho}_E(\texttt{Emp}))$$

To de-correlate we compute:

$$f' = \hat{e}_5 \bowtie (\boldsymbol{\pi}_{D.*,E.*}(e_1) \bowtie \boldsymbol{\pi}_{D.*}(e_2)) = (\boldsymbol{\pi}_{D.*,E.*}(e_1) \bowtie \boldsymbol{\pi}_{D.*}(e_2))$$

To this expression we add the WHERE and SELECT clause:

$$e_6 = \boldsymbol{\pi}_{D.\texttt{floor}}\boldsymbol{\sigma}_{E.did=D.did}(\boldsymbol{\pi}_{D.*,E.*}(e_1) \bowtie \boldsymbol{\pi}_{D.*}(e_2))$$

Optimization of Logical Queries

Translation of the Whole Query

```
Q1 UNION Q2
```

Since the schemas of e_4 and e_6 are the same, the union is straightforward:

$$e = e_4 \cup e_6$$

Written in full:

$$e = \boldsymbol{\pi}_{D.\texttt{floor}} \boldsymbol{\sigma}_{D.\texttt{floor}=1 \wedge E.did=D.did}$$
$$\boldsymbol{\pi}_{D.*,E.*} \boldsymbol{\sigma}_{F_2.\texttt{did}=E.\texttt{did} \wedge E_2.\texttt{did}=D.\texttt{did} \wedge E_2.\texttt{eid}=E.\texttt{eid} \wedge F_2.\texttt{expenses}=300 \wedge E.\texttt{did}=F_2.\texttt{did}}$$
$$(\boldsymbol{\rho}_D(\texttt{Dept}) \times \boldsymbol{\rho}_E(\texttt{Emp}) \times \boldsymbol{\rho}_{F2}(\texttt{Finance}) \times \boldsymbol{\rho}_{E2}(\texttt{Emp}))$$
$$\cup$$
$$\boldsymbol{\pi}_{D.\texttt{floor}} \boldsymbol{\sigma}_{E.did=D.did}($$
$$[\boldsymbol{\pi}_{D.*,E.*} \boldsymbol{\sigma}_{F_2.\texttt{did}=E.\texttt{did} \wedge E_2.\texttt{did}=D.\texttt{did} \wedge E_2.\texttt{eid}=E.\texttt{eid} \wedge F_2.\texttt{expenses}=300 \wedge E.\texttt{did}=F_2.\texttt{did}}$$
$$(\boldsymbol{\rho}_D(\texttt{Dept}) \times \boldsymbol{\rho}_E(\texttt{Emp}) \times \boldsymbol{\rho}_{F2}(\texttt{Finance}) \times \boldsymbol{\rho}_{E2}(\texttt{Emp}))]$$
$$\bowtie [\boldsymbol{\pi}_{D.*} \boldsymbol{\sigma}_{F_1.\texttt{budget}>150 \wedge D_2.\texttt{did}=F_1.\texttt{did} \wedge D_2.\texttt{floor}=D.\texttt{floor}}$$
$$(\boldsymbol{\rho}_D(\texttt{Dept}) \times \boldsymbol{\rho}_{D2}(\texttt{Dept}) \times \boldsymbol{\rho}_{F1}(\texttt{Finance}))])$$

Optimization of Logical Queries

Redundant Joins Removal

The query comprises the following maximal select-project-join subexpressions:

- $\boldsymbol{\pi}_{D.\texttt{floor}}\boldsymbol{\sigma}_{D.\texttt{floor}=1 \wedge E.did=D.did}\boldsymbol{\pi}_{D.*,E.*}\boldsymbol{\sigma}_{\ldots}(\boldsymbol{\rho}_D(\texttt{Dept}) \times \boldsymbol{\rho}_E(\texttt{Emp}) \times \boldsymbol{\rho}_{F2}(\texttt{Finance}) \times \boldsymbol{\rho}_{E2}(\texttt{Emp}))$
- $[\boldsymbol{\pi}_{D.*,E.*}\boldsymbol{\sigma}_{\ldots}(\boldsymbol{\rho}_D(\texttt{Dept}) \times \boldsymbol{\rho}_E(\texttt{Emp}) \times \boldsymbol{\rho}_{F2}(\texttt{Finance}) \times \boldsymbol{\rho}_{E2}(\texttt{Emp}))]$
- $(\boldsymbol{\rho}_D(\texttt{Dept}) \times \boldsymbol{\rho}_{D2}(\texttt{Dept}) \times \boldsymbol{\rho}_{F1}(\texttt{Finance}))$

Note that "F_1.budget > 150" cannot be included in a select-project-join expression. Also note that the third expression does not contain redundant joins (Why?).

Optimization of Logical Queries

Redundant Joins Removal

The first expression corresponds to:

$$Q_1(\text{“1”}) \leftarrow \texttt{Dept}(a_1, a_2, \text{“1”}, a_4), \texttt{Emp}(b_1, a_1, b_3, b_4), \texttt{Finance}(a_1, c_2, c_3, \text{“300”}), \texttt{Emp}(b_1, a_1, d_3, d_4)$$

The first and third atoms cannot be removed (Why?)

We check whether we can remove the second atom:
$Q_2(\text{“1”}) \leftarrow \texttt{Dept}(a_1, a_2, \text{“1”}, a_4), \texttt{Finance}(a_1, c_2, c_3, \text{“300”}), \texttt{Emp}(b_1, a_1, d_3, d_4)$

The corresponding canonical database: $D_2(\text{“1”}) =$
$\{\texttt{Dept}(\dot{a}_1, \dot{a}_2, \text{“1”}, \dot{a}_4), \texttt{Finance}(\dot{a}_1, \dot{c}_2, \dot{c}_3, \text{“300”}), \texttt{Emp}(\dot{b}_1, \dot{a}_1, \dot{d}_3, \dot{d}_4)\}$

Clearly $(\text{“1”}) \in Q_1(D_2)$ because of the matching

$a_1 \mapsto \dot{a}_1$	$a_2 \mapsto \dot{a}_2$	$a_4 \mapsto \dot{a}_4$	
$b_1 \mapsto \dot{b}_1$	$b_3 \mapsto \dot{d}_3$	$b_4 \mapsto \dot{d}_4$	
$c_2 \mapsto \dot{c}_2$	$c_3 \mapsto \dot{c}_3$	$d_3 \mapsto \dot{d}_3$	$d_4 \mapsto \dot{d}_4$

hence $Q_2 \subseteq Q_1$. The other direction always holds. Hence $Q_1 \equiv Q_2$

Optimization of Logical Queries

Redundant Joins Removal

No other atom can be removed (Why?).

The optimal query is hence
$Q_2(\text{“1”}) \leftarrow \texttt{Dept}(a_1, a_2, \text{“1”}, a_4), \texttt{Finance}(a_1, c_2, c_3, \text{“300”}), \texttt{Emp}(b_1, a_1, d_3, d_4)$

Translating this query back to the relational algebra, we obtain:

$$\boldsymbol{\pi}_{D.\texttt{floor}}([\boldsymbol{\sigma}_{D.\texttt{floor}=1 \wedge E_2.did=D.did \wedge F_2.\texttt{did}=E_2.\texttt{did} \wedge E_2.\texttt{did}=D.\texttt{did} \wedge F_2.\texttt{expenses}=300} (\boldsymbol{\rho}_D(\texttt{Dept}) \times \boldsymbol{\rho}_{F2}(\texttt{Finance}) \times \boldsymbol{\rho}_{E2}(\texttt{Emp}))])$$

Optimization of Logical Queries

Redundant Joins Removal

The second expression is:

$$[\boldsymbol{\pi}_{D.*,E.*}\boldsymbol{\sigma}_{F_2.\texttt{did}=E.\texttt{did}\wedge E_2.\texttt{did}=D.\texttt{did}\wedge E_2.\texttt{eid}=E.\texttt{eid}\wedge F_2.\texttt{expenses}=300\wedge E.\texttt{did}=F_2.\texttt{did}} (\boldsymbol{\rho}_D(\texttt{Dept}) \times \boldsymbol{\rho}_E(\texttt{Emp}) \times \boldsymbol{\rho}_{F2}(\texttt{Finance}) \times \boldsymbol{\rho}_{E2}(\texttt{Emp}))]$$

Translated:

$$Q_3(a_1, a_2, a_3, a_4, b_1, b_3, b_4) \leftarrow \texttt{Dept}(a_1, a_2, a_3, a_4), \texttt{Emp}(b_1, a_1, b_3, b_4), \texttt{Finance}(a_1, c_2, c_3, \text{``300''}), \texttt{Emp}(b_1, a_1, d_3, d_4)$$

We cannot remove the second atom, this time (why?)

Optimization of Logical Queries

Redundant Joins Removal

However, with a similar mapping as for the first expression, the fourth atom can be removed, and we obtain:

$$Q_4(a_1, a_2, a_3, a_4, b_1, b_3, b_4) \leftarrow \texttt{Dept}(a_1, a_2, a_3, a_4), \texttt{Emp}(b_1, a_1, b_3, b_4), \\ \texttt{Finance}(a_1, c_2, c_3, \text{“300”})$$

We have thus $Q_4 \subseteq Q_3$. The other direction always holds. Hence $Q_3 \equiv Q_4$.

Translating this query back to the relational algebra, we obtain:

$$[\boldsymbol{\pi}_{D.*,E.*}\boldsymbol{\sigma}_{F_2.\texttt{did}=E.\texttt{did}\wedge E.\texttt{did}=D.\texttt{did}\wedge F_2.\texttt{expenses}=300\wedge E.\texttt{did}=F_2.\texttt{did}} \\ (\boldsymbol{\rho}_D(\texttt{Dept}) \times \boldsymbol{\rho}_E(\texttt{Emp}) \times \boldsymbol{\rho}_{F2}(\texttt{Finance}))]$$

Optimization of Logical Queries

Redundant Joins Removal

The third expression is:

$$(\boldsymbol{\rho}_D(\texttt{Dept}) \times \boldsymbol{\rho}_{D2}(\texttt{Dept}) \times \boldsymbol{\rho}_{F1}(\texttt{Finance}))$$

Translated:

$$\begin{aligned} Q_5(a_1, \ldots, a_4, b_1, \ldots, b_4, c_1, \ldots, c_4) \leftarrow & \texttt{Dept}(a_1, a_2, a_3, a_4), \texttt{Dept}(b_1, b_2, b_3, b_4), \\ & \texttt{Finance}(c_1, c_2, c_3, c_4) \end{aligned}$$

No atoms can be removed (why?)

Optimization of Logical Queries

Redundant Joins Removal

The optimized expression is therefore:

$$e = \boldsymbol{\pi}_{D.\texttt{floor}}\Big([\boldsymbol{\sigma}_{D.\texttt{floor}=1 \wedge E_2.did=D.did \wedge F_2.\texttt{did}=E_2.\texttt{did} \wedge E_2.\texttt{did}=D.\texttt{did} \wedge F_2.\texttt{expenses}=300}$$

$$(\boldsymbol{\rho}_D(\texttt{Dept}) \times \boldsymbol{\rho}_{F2}(\texttt{Finance}) \times \boldsymbol{\rho}_{E2}(\texttt{Emp}))]\Big)$$

$$\cup$$

$$\boldsymbol{\pi}_{D.\texttt{floor}}\Big($$

$$[\boldsymbol{\pi}_{D.*,E.*}\boldsymbol{\sigma}_{F_2.\texttt{did}=E.\texttt{did} \wedge E.\texttt{did}=D.\texttt{did} \wedge F_2.\texttt{expenses}=300 \wedge E.\texttt{did}=F_2.\texttt{did}}$$

$$(\boldsymbol{\rho}_D(\texttt{Dept}) \times \boldsymbol{\rho}_E(\texttt{Emp}) \times \boldsymbol{\rho}_{F2}(\texttt{Finance}))]$$

$$\bowtie [\boldsymbol{\pi}_{D.*}\boldsymbol{\sigma}_{F_1.\texttt{budget}>150 \wedge D_2.\texttt{did}=F_1.\texttt{did} \wedge D_2.\texttt{floor}=D.\texttt{floor}}$$

$$(\boldsymbol{\rho}_D(\texttt{Dept}) \times \boldsymbol{\rho}_{D2}(\texttt{Dept}) \times \boldsymbol{\rho}_{F1}(\texttt{Finance}))]\Big)$$

Cost-based plan selection

Task

(refer to the handouts for the full exercise)

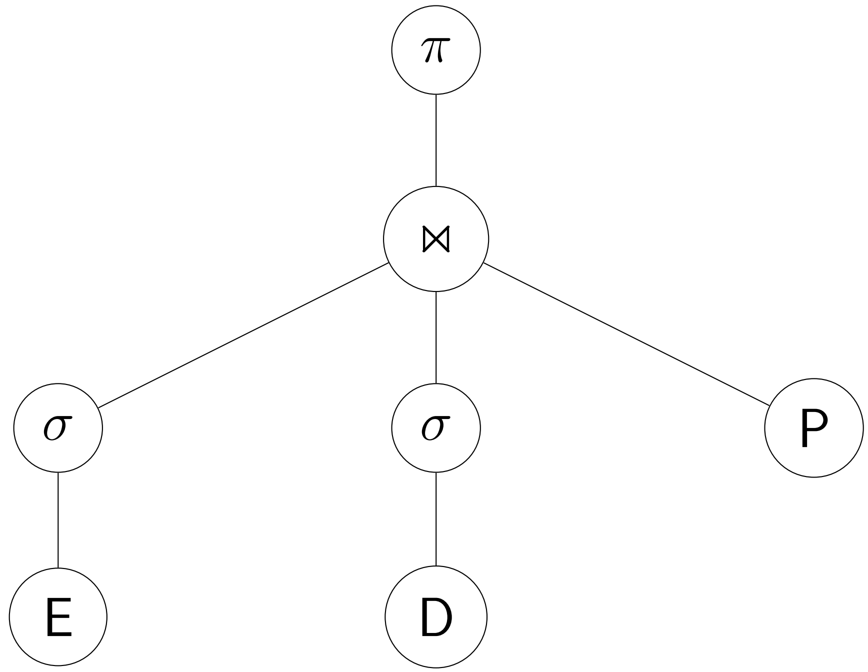
Construct a sufficiently optimal physical query plan for:

$$\pi_{\texttt{E.eid},\texttt{D.did},\texttt{P.pid}} \sigma_{\texttt{E.sal}=50000}(E) \bowtie \sigma_{\texttt{D.budget} \geq 20000}(D) \bowtie P$$

Assume that employee salaries are uniformly distributed over the range $[10009, 110008]$ and that project budgets are uniformly distributed over $[10000, 30000]$. There are clustered indexes available on `E.sal`, `D.did` and `P.pid`.

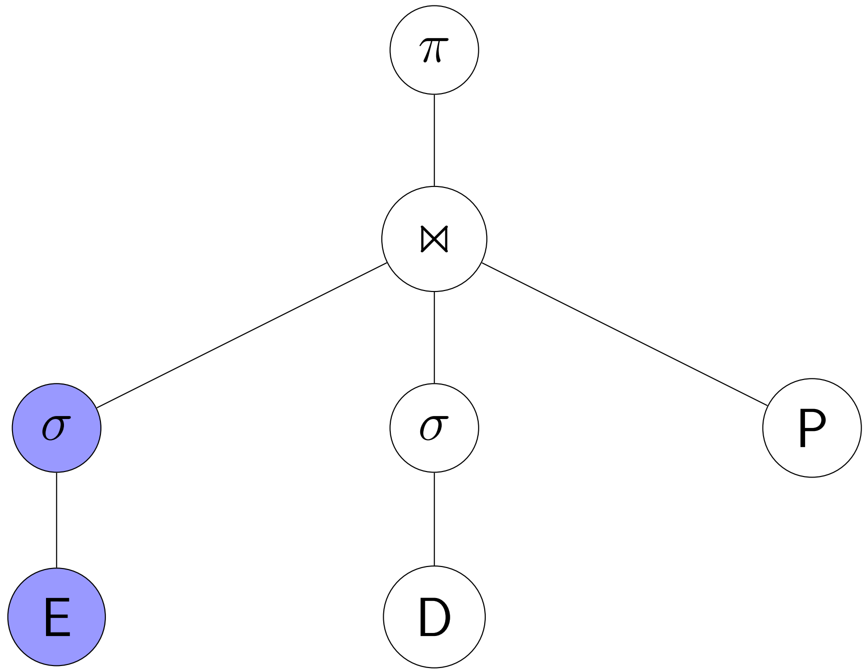
Cost-based plan selection

Solution



Cost-based plan selection

Solution



Cost-based plan selection

Solution

Subexpression:

$$\sigma_{\texttt{E.sal}=50000}(E)$$

First possibility: we use the clustered index on `E.sal` to get the records such that $\texttt{E.sal} = 50000$.

The number of tuples that satisfy the salary requirement is estimated to:

$$\left\lceil \frac{1}{110008 - 10009 + 1} \text{ selectivity } \times 20000 \text{ employees} \right\rceil = 1 \text{ tuples}$$

Hence, the result can be stored in 1 block:

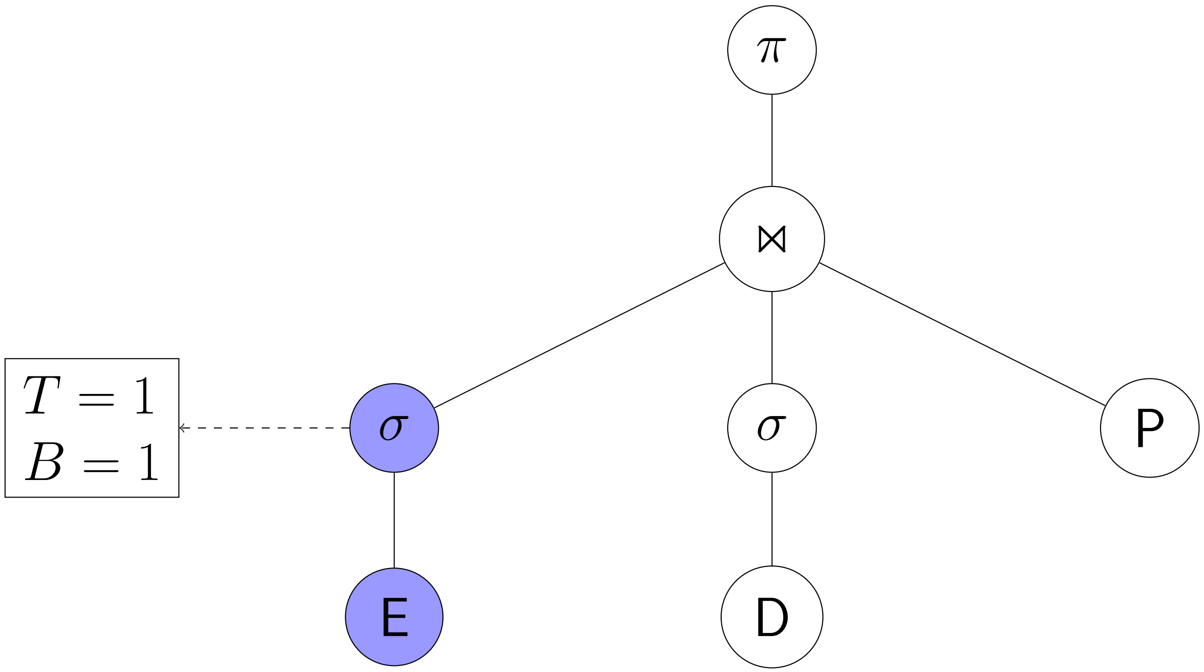
$$\left\lceil \frac{20 \text{ bytes}}{4000 \text{ bytes/block}} \right\rceil = 1 \text{ block}$$

A table scan would cost:

$$\frac{20000 \text{ tuples}}{\left\lfloor \frac{4000 \text{ bytes/block}}{20 \text{ bytes/tuple}} \right\rfloor} = 100 \text{ block I/Os}$$

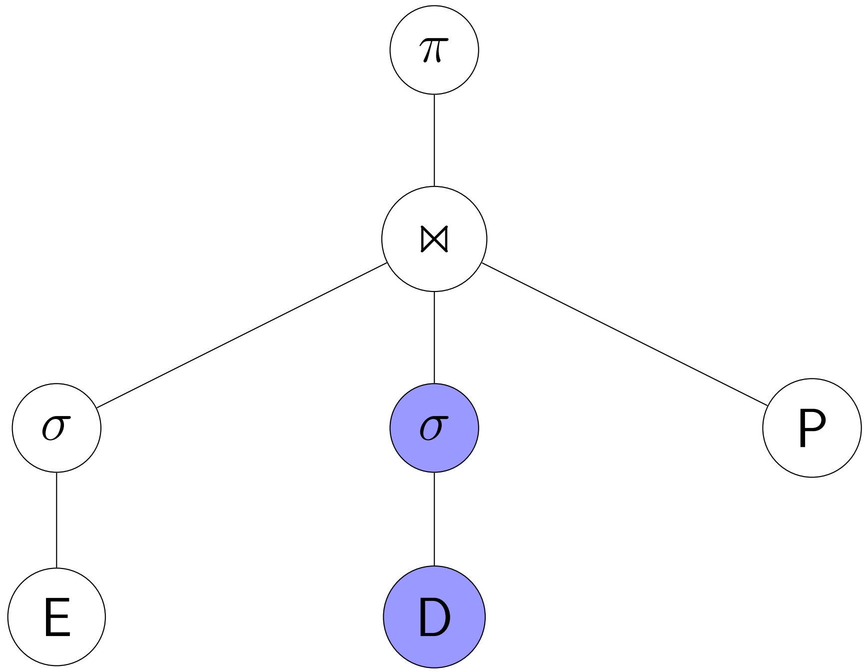
Cost-based plan selection

Solution



Cost-based plan selection

Solution



Cost-based plan selection

Solution

Subexpression:

$$\sigma_{\texttt{D.budget} \geq 20000}(D)$$

The number of tuples returned is estimated to 2501:

$$\left\lceil \underbrace{\frac{30000 - 20000 + 1}{30000 - 10000 + 1}}_{\text{selectivity}} \times 5000 \text{ departments} \right\rceil = 2501 \text{ tuples}$$

This corresponds to 26 Blocks:

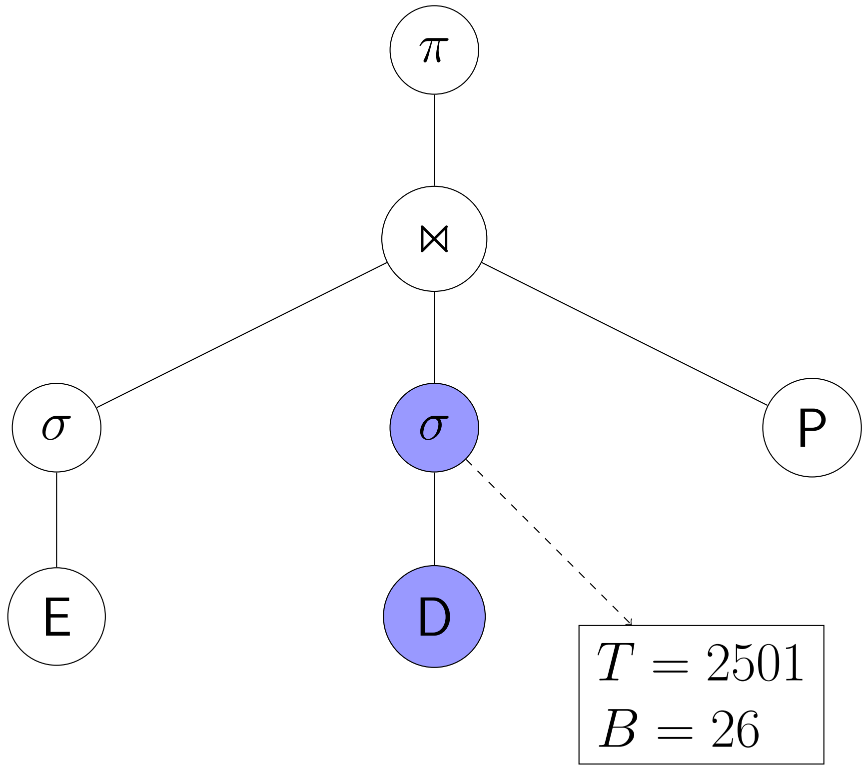
$$\frac{2501}{\left\lfloor \frac{4000 \text{ bytes/block}}{40 \text{ bytes/tuple}} \right\rfloor} = 26 \text{ blocks}$$

Since no index is available, a table scan is our only possibility:

$$\frac{5000}{\left\lfloor \frac{4000 \text{ bytes/block}}{40 \text{ bytes/tuple}} \right\rfloor} = 50 \text{ blocks}$$

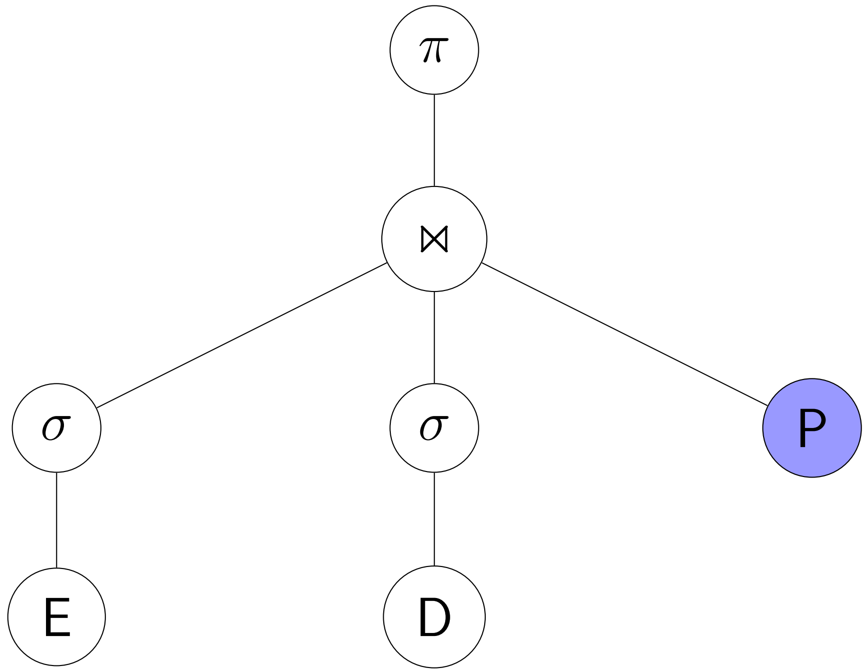
Cost-based plan selection

Solution



Cost-based plan selection

Solution



Cost-based plan selection

Solution

Subexpression:

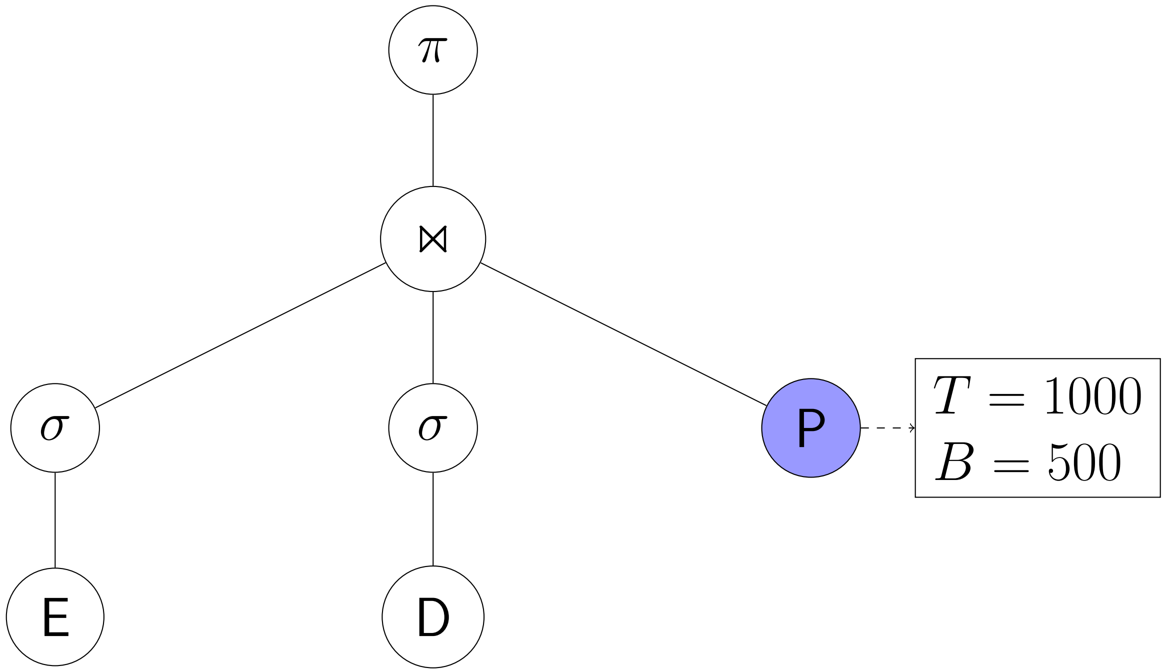
$$P$$

A table scan on P requires 500 block I/O's. This is also the estimated number of blocks returned:

$$\frac{1000 \text{ tuples}}{\left\lfloor \frac{4000 \text{ bytes/block}}{2000 \text{ bytes/tuple}} \right\rfloor} = 500 \text{ blocks}$$

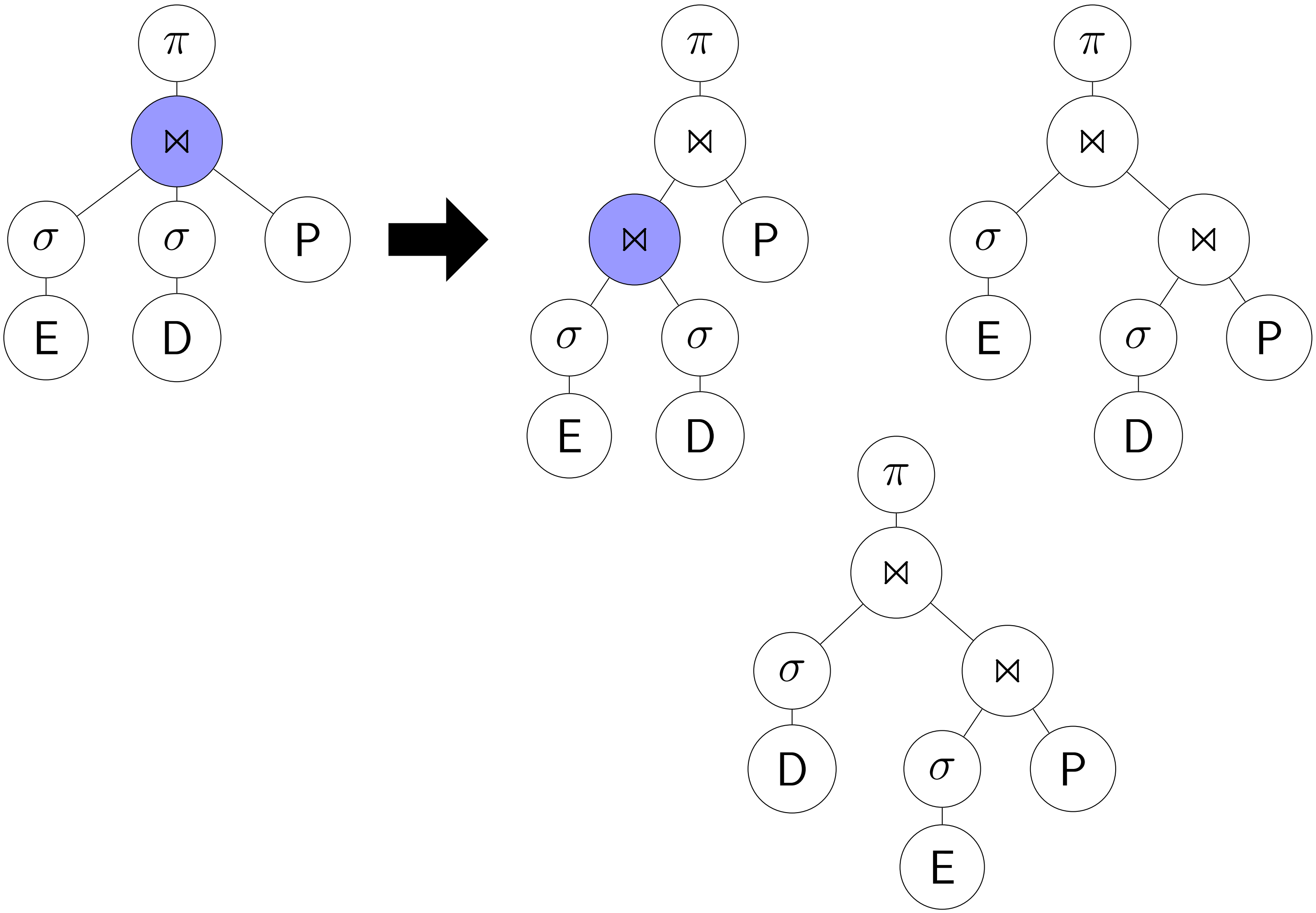
Cost-based plan selection

Solution



Cost-based plan selection

Solution



Cost-based plan selection

Solution

Now, we must determine an ordering for the joins. We consider all pairs of joins and keep the one with the smallest cost.

$$\underbrace{\sigma_{\texttt{e.sal}=\texttt{50000}}(E)}_{e_1} \text{ and } \underbrace{\sigma_{\texttt{d.budget}\geq\texttt{20000}}(D)}_{e_2}$$

The selection on each side requires one buffer to execute, leaving only 10 buffers for the join.

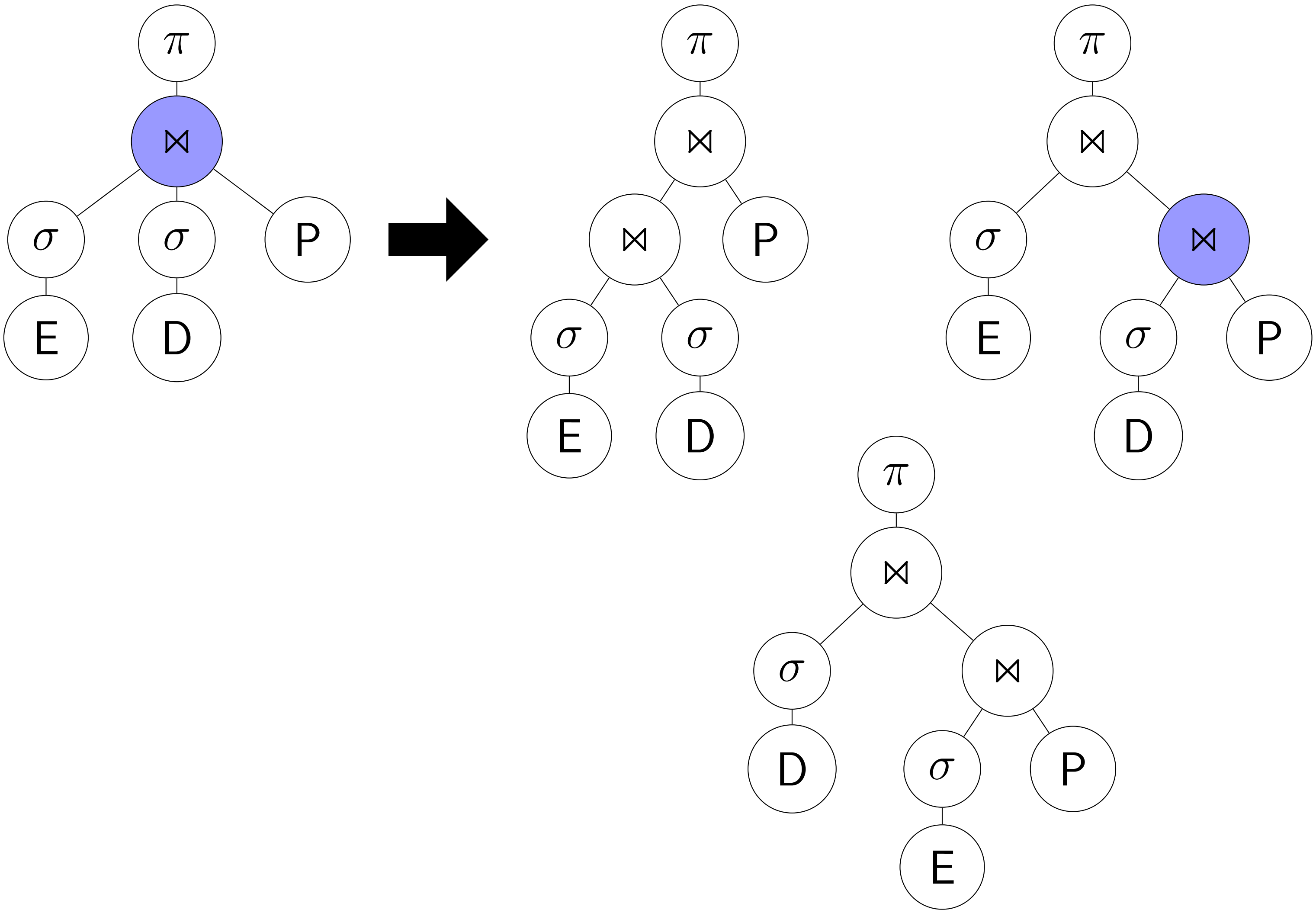
The output of e_1 contains only 1 tuples, and can therefore be computed in 1 block. Since $1 = B(e_1) \leq M = 10$, we can apply the one-pass join algorithm. Its cost is

$$B(e_1) + B(e_2) = 1 + 26 = 27 \text{ I/O's}$$

An index-join cannot be used on e_2 since it is not a base relation. All other join methods always cost more than one-pass join. Hence the one-pass join is preferred.

Cost-based plan selection

Solution



Cost-based plan selection

Solution

The second join pair is:

$$\underbrace{\sigma_{\texttt{D.budget}\geq 20000}(D)}_{e_2} \text{ and } P$$

We have 11 buffers at our disposal, given that we need 1 buffer to perform the selection in e_2. It is not possible to use a one-pass join, since $26 = B(e_2) \geq M = 11$ and $500 = B(P) \geq M = 11$.

A block-based nested-loop join costs:

$$B(e_2) + \left\lceil \frac{B(e_2)}{M-1} \right\rceil \times B(P) = 26 + \left\lceil \frac{26}{10} \right\rceil \times 500 = 1526 \text{ I/Os}$$

Cost-based plan selection

Solution

The second join pair is:

$$\underbrace{\sigma_{\texttt{D.budget}\geq 20000}(D)}_{e_2} \text{ and } P$$

We have enough memory to perform an optimized sort-merge join:

$$8 = \left\lceil \frac{B(e_2)}{M^{\lceil \log_M B(e_2) \rceil - 1}} \right\rceil + \left\lceil \frac{B(P)}{M^{\lceil \log_M B(P) \rceil - 1}} \right\rceil \leq M = 11 \text{ available buffers}$$

This optimized sort-merge join has a cost of:

$$\begin{aligned} & 2B(e_2)\left\lceil \log_M B(e_2) \right\rceil + 2B(P)\left\lceil \log_M B(P) \right\rceil - B(e_2) - B(P) \\ = \; & 2 \times 26 \times 2 + 2 \times 500 \times 3 - 26 - 500 \\ = \; & 2578 \text{ I/O's} \end{aligned}$$

Cost-based plan selection

Solution

Assuming that the clustered index on `P.pid` is a *BTree*, it ensues that P is already sorted on this join attribute. Given that we then only need to sort e_2, the cost of a non-optimized sort-merge join is:

$$2B(e_2)\left\lceil \log_M B(e_2) \right\rceil + B(e_2) + B(P)$$

Futhermore, we can optimize the last merge:

$$4 \text{ necessary buffers} = \left\lceil \frac{B(e_2)}{M} \right\rceil + 1 \leq M = 11 \text{ available buffers}$$

The cost thereof is:

$$\begin{aligned} & 2B(e_2)(\lceil \log_M B(e_2) \rceil - 1) + B(e_2) + B(P) \\ = \ & 2 \times 26 \times 1 + 26 + 500 \\ = \ & 578 \text{ I/Os} \end{aligned}$$

Cost-based plan selection

Solution

The cost of an hash-join is:

$$
\begin{aligned}
& 2B(e_2)\lceil \log_{M-1} B(e_2) - 1 \rceil + 2B(P)\lceil \log_{M-1} B(e_2) - 1 \rceil + B(e_2) + B(P) \\
= & \ 2 \times 26 \times 1 + 2 \times 500 \times 1 + 26 + 500 \\
= & \ 1578 \text{ I/O's}
\end{aligned}
$$

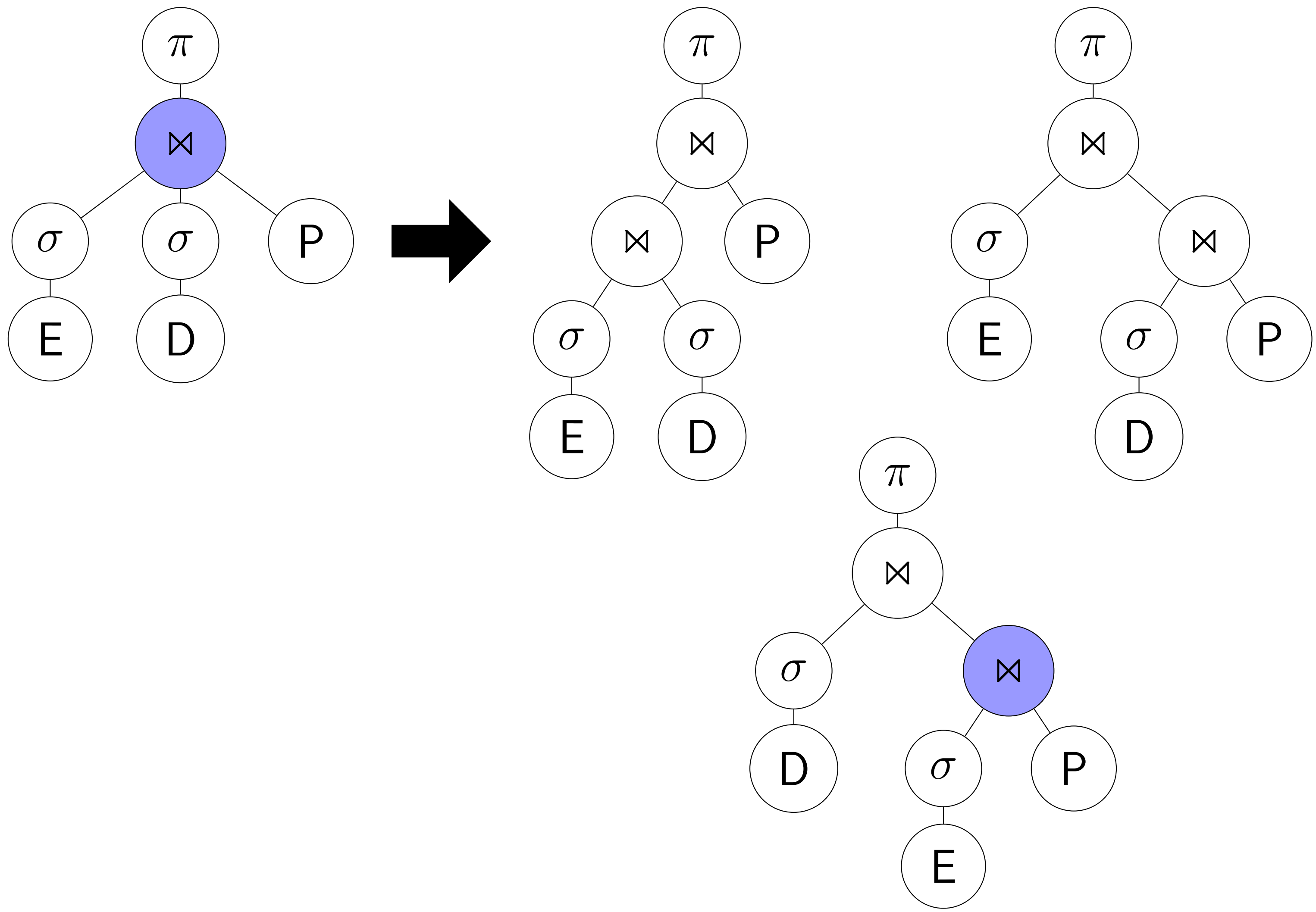
It is also possible to use an index-join, using the clustered index on `P.pid`. This method has a cost of:

$$
B(e_2) + T(e_2) \times \left\lceil \frac{B(P)}{V(P, \texttt{pid})} \right\rceil = 26 + 2501 \times 1 = 2527 \text{ I/O's}
$$

Hence, we assume that the index on `P.pid` is a BTree, and sorting P is not necessary. In that case the optimized sort-merge join that only sorts e_2 is preferred.

Cost-based plan selection

Solution



Cost-based plan selection

Solution

The third join pair is:

$$\underbrace{\sigma_{\texttt{E.sal=50000}}(E)}_{e_1} \text{ and } P$$

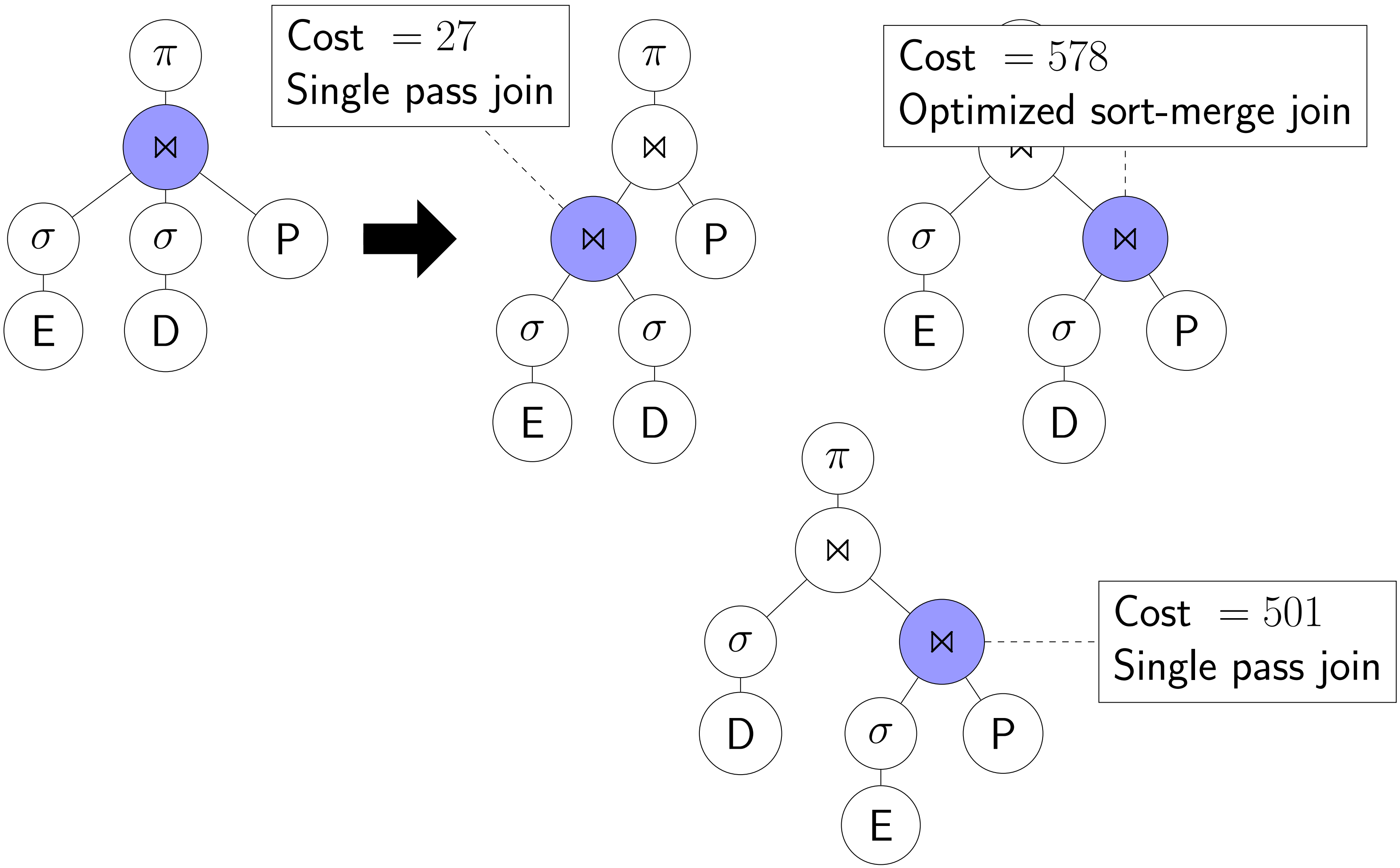
Note that this join is a full cartesian product. A one-pass join is available at the following cost:

$$B(e_1) + B(P) = 1 + 500 = 501 \text{ I/O's}$$

No index can help up for this join, and the one-pass join algorithm gives the best cost.

Cost-based plan selection

Solution



Cost-based plan selection

Solution

The join-pair with the least cost is therefore:

$$\underbrace{\sigma_{\texttt{e.sal}=50000}(E)}_{e_1} \text{ and } \underbrace{\sigma_{\texttt{D.budget}\geq 20000}(D)}_{e_2}$$

Where an one-pass join on `E.did` is used. Therefore, only 2 buffers are necessary (why?).

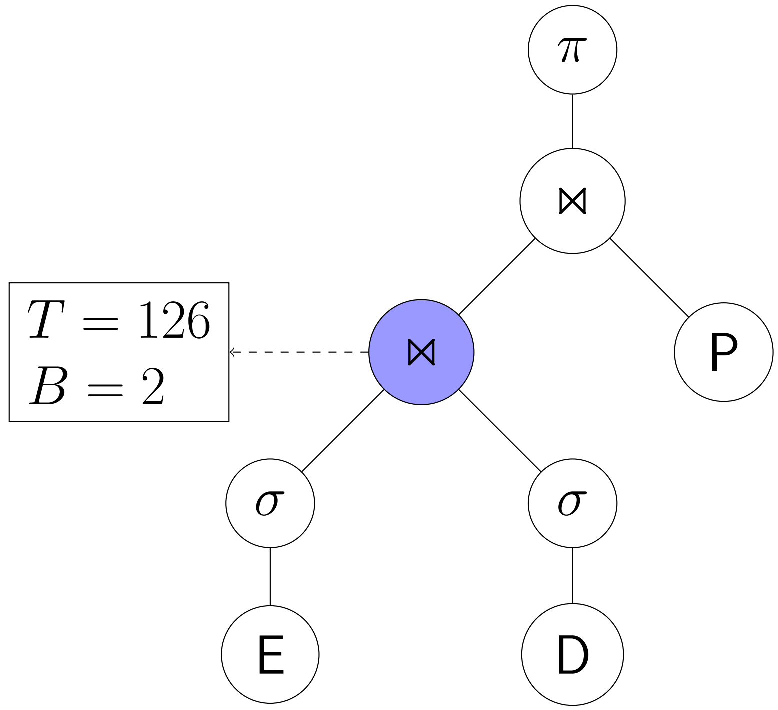
The estimated number of tuples in the output of this join is:

$$\frac{T(e_1) \times T(e_2)}{\max(V(e_1, \texttt{did}), V(e_2, \texttt{did}))} = \frac{1 \times 2501}{20} = 126$$

These records are 60 bytes long and can be stored in 2 blocks

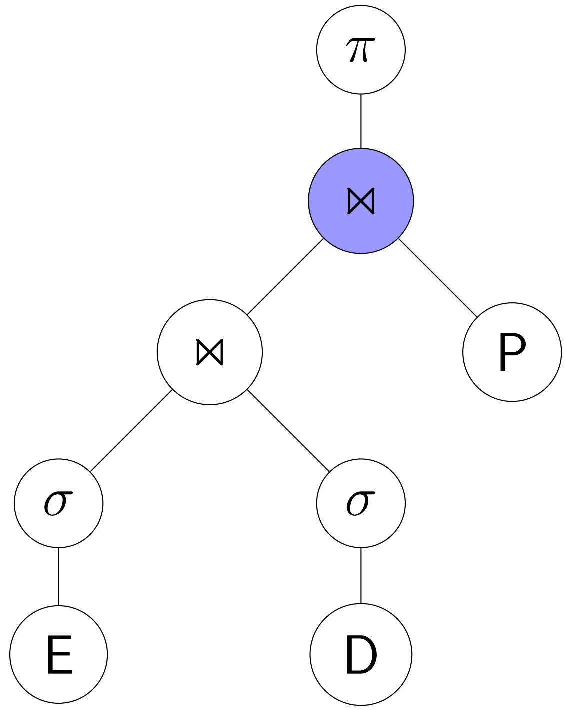
Cost-based plan selection

Solution



Cost-based plan selection

Solution



Cost-based plan selection

Solution

We still need to find the best way to join the whole expression

$$\underbrace{\sigma_{\texttt{e.sal}=50000}(E) \bowtie \sigma_{\texttt{D.budget}\geq 20000}(D)}_{e_3} \text{ and } P$$

We expect to have $12 - 3 = 9$ main memory buffers available.

The output of e_3 fits in 2 blocks. Given that $2 = B(e_3) \leq M = 8$, a one-pass join is possible. The cost thereof is:

$$B(e_3) + B(P) = 2 + 500 = 502$$

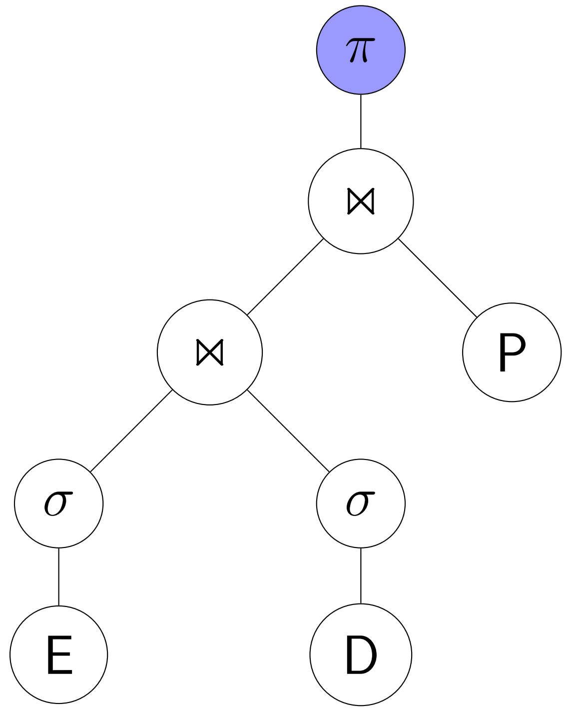
This join can also be performed by means of an index-join, using the clustered index on `P.pid`.

$$B(e_3) + T(e_3) \times \left\lceil \frac{B(P)}{V(P, \texttt{pid})} \right\rceil = 2 + 125 \times 1 = 127 \text{ I/O's}$$

Hence, the index-join is preferred.

Cost-based plan selection

Solution



Cost-based plan selection

Solution

The projection $\pi_{\texttt{E.eid},\texttt{D.did},\texttt{P.pid}}$ can be performed on the fly at the same time as the last join.

Notice that we did not need to materialize any of the intermediate results.

Cost-based plan selection

Solution