Cost-based plan selection

Exercise 1.1

(refer to the handouts for the full exercise)

$$\sigma_{a=1 \text{ AND } b=2 \text{ AND } d=3}(R)$$

Give the best physical plan (index scan or table scan, possibly followed by a filter) and the cost of the selection.
Cost-based plan selection

Exercise 1.1

(refer to the handouts for the full exercise)

\[ \sigma_{a=1 \text{ AND } b=2 \text{ AND } d=3}(R) \]

Give the best physical plan (index scan or table scan, possibly followed by a filter) and the cost of the selection.

1. Cost of checking all conditions via a table scan + filter: \( B(R) = 1000 \) block I/Os.
2. Cost of an index-scan for condition \( a = 1 \), followed by a filter: \( B(R)/V(R, a) = 1000/20 = 50 \) block I/Os.
3. Cost of an index-scan for condition \( b = 2 \), followed by a filter: \( T(R)/V(R, b) = 5000/1000 = 5 \) block I/Os.
4. Cost of an index-scan for condition \( d = 3 \), followed by a filter: \( T(R)/V(R, d) = 5000/500 = 10 \) block I/Os.

Hence, we select plan (3).
Cost-based plan selection

Exercise 1.2

(refer to the handouts for the full exercise)

\[ \sigma_{a=1 \, \text{AND}} \, b=2 \, \text{AND} \, c \geq 3 \, (R) \]

Give the best physical plan (index scan or table scan, possibly followed by a filter) and the cost of the selection.

1. Cost of checking all conditions via a table scan + filter: \( B(R) = 1000 \) block I/Os.
2. Cost of an index-scan for condition \( a = 1 \), followed by a filter: \( B(R)/V(R, a) = 1000/20 = 50 \) block I/Os.
3. Cost of an index-scan for condition \( b = 2 \), followed by a filter: \( T(R)/V(R, b) = 5000/1000 = 5 \) block I/Os.
4. Cost of an index-scan for condition \( c \geq 3 \), followed by a filter: \( T(R)/3 = 5000/3 = 1667 \) block I/Os.

Hence, we select plan (3).
Cost-based plan selection

Exercise 1.3

(refer to the handouts for the full exercise)

\[ \sigma_{a=1 \text{ AND } b \leq 2 \text{ AND } c \geq 3}(R) \]

Give the best physical plan (index scan or table scan, possibly followed by a filter) and the cost of the selection.

1. Cost of checking all conditions via a table scan + filter: \( B(R) = 1000 \) block I/Os.
2. Cost of an index-scan for condition \( a = 1 \), followed by a filter: \( B(R)/V(R, a) = 1000/20 = 50 \) block I/Os.
3. Cost of an index-scan for condition \( b \leq 2 \), followed by a filter: \( T(R)/3 = 5000/3 = 1667 \) block I/Os.
4. Cost of an index-scan for condition \( c \geq 3 \), followed by a filter: \( T(R)/3 = 5000/3 = 1667 \) block I/Os.

Hence, we select plan (2).
Cost-based plan selection

Task

(refer to the handouts for the full exercise)

\[ \pi_{D.dname,F.budget}(\sigma_{E.hobby='yodeling'} \text{ AND } E.sal \geq 59000}(E) \bowtie \sigma_{D.floor=1}(D) \bowtie F) \]

Construct a sufficiently optimal physical query plan. Use disk I/Os as your optimization metric.
Cost-based plan selection

Task

\[ \pi_{D.dname, F.budget}(\sigma_{E.hobby='yodeling'} \text{ AND } E.sal \geq 59000(E) \Join \sigma_{D.floor=1}(D) \Join F) \]

Solution plan

Note: We use a greedy approach

1. Find best plan for each selection individually
   1. \( \sigma_{E.hobby='yodeling'} \text{ AND } E.sal \geq 59000(E) \)
   2. \( \sigma_{D.floor=1}(D) \)

2. Select the best pairwise join
   1. \( \sigma_{E.hobby='yodeling'} \text{ AND } E.sal \geq 59000(E) \Join \sigma_{D.floor=1}(D) \)
   2. \( \sigma_{E.hobby='yodeling'} \text{ AND } E.sal \geq 59000(E) \Join F \)
   3. \( \sigma_{D.floor=1}(D) \Join F \)

3. Join the previously selected first join with the third, remaining relation
Cost-based plan selection

Solution

Subexpression:

\[ \sigma_{E.hobby = 'yodeling'} \text{ AND } E.sal \geq 59000(E) \]

Possibilities:

1. Use clustered BTree index on E.sal, then filter on E.hobby
2. Scan the table and filter on both conditions
Cost-based plan selection

Solution

Subexpression:

\[ \sigma_{E.hobby='yodeling'} \text{ AND } E.sal \geq 59000(E) \]

**First possibility:** we use the clustered BTree index on E.sal to get the records such that \( E.sal \geq 59000 \), and a filter is applied to retain those with the correct hobby.

The number of tuples that satisfy the salary requirement is:

\[
\frac{60000 - 59000}{60000 - 10000} \times 50000 \text{ employees} = 1000 \text{ tuples}
\]

Hence, the index scan has a cost of 18 block I/Os (rounding up):

\[
\frac{1000 \text{ tuples}}{\frac{2048 \text{ bytes/block}}{35 \text{ bytes/tuple}}} = 18 \text{ blocks}
\]

The filtering can be performed on the fly without any supplemental I/O.
Cost-based plan selection

Solution

Subexpression

\[ \sigma_{E.hobby='yodeling'} \text{ AND } E.sal \geq 59000 \]

**Second possibility**: we forget about the index, and do the selection by scanning the table and filtering. This has a cost of \( B(E) \) I/Os (rounding up):

\[
B(E) = \frac{50000 \text{ tuples}}{\frac{2048 \text{ bytes/block}}{35 \text{ bytes/tuple}}} = 863 \text{ blocks}
\]
Cost-based plan selection

Solution

Subexpression:

\[ \sigma_{\text{E.hobby}=\text{'yodeling'} \ AND \ E.\text{sal} \geq 59000}(E) \]

Possibilities:

1. Use clustered BTree index on E.sal, then filter on E.hobby (18 blocks)
2. Scan the table and filter on both conditions (863 blocks)

Intermediate result:

- The first method is indeed better than the second one.
- The estimated number of tuples in the output of this subexpression is:

\[
\frac{60000 - 59000}{60000 - 10000} \times \frac{1}{200} \times 50000 \text{ tuples} = 5 \text{ tuples}
\]
Cost-based plan selection

Solution

Subexpression

\[ \sigma_{D \text{.floor}=1}(D) \]

Possibilities:

1. Use the index
2. Scan the table and filter on condition
Cost-based plan selection

Solution

Subexpression

\[ \sigma_{D, \text{floor}=1}(D) \]

First possibility: use the index. The number of tuples that satisfy the selection condition is:

\[ \frac{T(D)}{V(D, \text{floor})} = \frac{5000}{2} = 2500 \]

Since the index is not clustered, this approach has a cost of 2500 block I/Os.

Second possibility: a table scan followed by a filter. This costs \( B(D) \) block I/Os (rounding up).

\[ B(D) = \frac{5000 \text{ tuples}}{\left[ \frac{2048 \text{ bytes/block}}{40 \text{ bytes/tuple}} \right]} = 99 \text{ blocks} \]

The second possibility is indeed better than the first and is therefore preferred.

The estimated number of tuples in the output of this subexpression is 2500.
Cost-based plan selection

Solution

Now, we must determine an ordering for the joins.

1. \( \sigma_{E\text{.hobby}=\text{'yodeling'}} \text{ AND } E\text{.sal} \geq 59000 (E \bowtie \sigma_{D\text{.floor}=1} (D)) \)
2. \( \sigma_{E\text{.hobby}=\text{'yodeling'}} \text{ AND } E\text{.sal} \geq 59000 (E \bowtie F) \)
3. \( \sigma_{D\text{.floor}=1} (D) \bowtie F \)
Solution

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

\[ \sigma_{E.hobby='yodeling'} \text{ AND } E.sal \geq 59000(E) \quad \text{and} \quad \sigma_{D.floor=1}(D) \]

Note that there are only 8 buffers remaining, since we need 1 to execute the selection in \( e_1 \) and 1 for the selection in \( e_2 \).

The output of \( e_1 \) contains only 5 tuples, and can therefore be computed in 1 block. Since \( 1 = B(e_1) \leq M = 8 \), we can apply the one-pass join algorithm. Its cost is

\[
B(e_1) + B(e_2) = 1 + \frac{2500 \text{ tuples}}{2048 \text{ bytes/block}} + \frac{40 \text{ bytes/tuple}}{40 \text{ bytes/tuple}} = 51 \text{ block I/Os}
\]

Also, because there is no index on \( e_1 \) and \( e_2 \), we cannot apply index-based joins. The other join algorithms (nested loop, sort-join, hash-join) are always less efficient than the one-pass algorithm.
Cost-based plan selection

Solution

Second pair of joins:

\[
\sigma_{E.hobby='yodeling' \land E.sal\geq59000(E)} \quad \text{and} \quad F
\]

We have 9 buffers at our disposal, given that we need 1 buffers for the selection in \( e_1 \). Just as for the first join pair, we can apply the one-pass join since the output of \( e_1 \) fits in 1 block. The actual cost is:

\[
B(e_1) + B(F) = 1 + \frac{5000}{\lceil \frac{2048}{15} \rceil} = 38 \text{ I/O's}
\]

It is also possible to use an index-join, since we have a clustered BTree on \( F.did \). This method has a cost of:

\[
B(e_1) + T(e_1) \times \left[ \frac{B(F)}{V(F, did)} \right] = 1 + 5 = 6 \text{ I/O's}
\]

Here, the index-join is therefore preferred.
Cost-based plan selection

Solution

Third join pair:

\[ \sigma_{D_{\text{floor}}=1}(D) \quad \text{and} \quad F \]

e_2

We have 9 buffers at our disposal, given that we need 1 buffer to perform the selection in \( e_2 \). It is not possible to use the one-pass join algorithm. The non-optimized version of the sort-merge join costs:

\[
2B(e_2) \lceil \log_M B(e_2) \rceil + 2B(F) \lceil \log_M B(F) \rceil + B(e_2) + B(F)
\]

\[
= 2 \times 50 \times 2 + 2 \times 37 \times 2 + 50 + 37
\]

\[
= 435 \text{ I/O's}
\]

We cannot use the optimization here, since there is not enough memory to perform the last merge of the merge-sort along with that of the sort-join:

10 necessary buffers

\[
\left\lceil \frac{B(e_2)}{M} \right\rceil + \left\lceil \frac{B(F)}{M} \right\rceil \not\leq M = 9 \text{ available buffers}
\]
Cost-based plan selection

Solution

In fact, we have a clustered B-tree index on \( F.d.i.d \). It ensues that \( F \) is already sorted on this join attribute. Given that we just have to sort \( e_2 \), the cost is:

\[
2B(e_2) \left\lceil \log_M B(e_2) \right\rceil + B(e_2) + B(F) \\
= 2 \times 50 \times 2 + 50 + 37 \\
= 287 \text{ I/Os}
\]

Here, we can perform the last merge of the merge-sort together with that of the sort-join:

\[
7 \text{ necessary buffers} = \left\lceil \frac{B(e_2)}{M} \right\rceil + 1 \leq M = 9 \text{ available buffers}
\]

The best cost we can achieve for our sort-merge join is therefore:

\[
2B(e_2)(\left\lceil \log_M B(e_2) \right\rceil - 1) + B(e_2) + B(F) = 187 \text{ I/Os}
\]
Cost-based plan selection

Solution

The cost of a hash-join is:

\[ 2B(e_2) \left\lceil \log_{M-1} B(F) - 1 \right\rceil + 2B(F) \left\lceil \log_{M-1} B(F) - 1 \right\rceil + B(e_2) + B(F) \]
\[ = 2 \times 50 \times 1 + 2 \times 37 \times 1 + 50 + 37 \]
\[ = 261 \text{ I/O's} \]

It is also possible to use an index-join, using the clustered Btree index on F.did. This method has a cost of:

\[ B(e_2) + T(e_2) \times \left[ \frac{B(F)}{V(F, \text{did})} \right] \]
\[ = 50 + 2500 \times \left[ \frac{37}{5000} \right] \]
\[ = 2550 \text{ I/O's} \]

(Notice that there is no index available on \( e_2 \), hence we cannot perform an index-join with \( e_2 \) as the inner relation)

Here, the optimized sort-merge join (using the sorted index) is therefore preferred.
Cost-based plan selection

Solution

The join-pair with the least cost is therefore:

\[ \sigma_{E.\text{hobby}=\text{`yodeling`} \text{ AND } E.\text{sal} \geq 59000\left(E\right)} \text{ and } F \]

Where an index-join on \( F.\text{did} \) is used. Therefore, only 2 buffers are necessary (why?).

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

\[
\frac{T(e_1) \times T(F)}{\max\left(V(e_1, \text{did}), V(F, \text{did})\right)} = \frac{5 \times 5000}{5000} = 5
\]
Cost-based plan selection

Solution

We still need to find the best way to join $e_3$ with $e_2$

$$\sigma_{E.\text{hobby}=\text{yodeling}} \text{ AND } E.\text{sal} \geq 59000 (E) \bowtie F \text{ and } \sigma_{D.\text{floor}=1} (D)$$

For the computation of $e_3$ we use 2 buffers for the index-join. Hence, only 8 buffers remain available.

The output of $e_3$ contains only 5 tuples. The size of a tuple of $e_3$ is evaluated to $15 + 35$ bytes. Thus, the output of $e_3$ fits in one block. Given that $1 = B(e_3) \leq M = 8$, a one-pass join is possible. The cost thereof is:

$$B(e_3) + B(e_2) = 1 + \left\lfloor \frac{2500}{2048} \right\rfloor = 51$$

There is no index on the intermediate result. An index-join is therefore not to be considered. The other join methods cost always more than the one-pass algorithm.

Hence, the one-pass algorithm is preferred to perform the join between $e_3$ and $e_2$. 

Solution of the exercises 20
Cost-based plan selection

Solution

The projection $\pi_{D.dname,F.budget}$ 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.