Explain Analyse da operação 1

**halloween\_discount**

!!! podemos colocar as alterações aqui ou simplesmente usar os novos queries (ou não fazer alteração nenhuma, não é mt imporrtante e os tempos de execução em geral são piores)

**(14)** –indica a linha do código

**SELECT SUM(OL.quantity)**

**FROM orderlines OL, orders O, products P**

**WHERE O.orderid = OL.orderid**

**AND P.prod\_id = OL.prod\_id**

**AND P.category = 11**

**AND O.orderid = id\_order; --teve de se substituir por um valor concreto**

Segundo o EXPLAIN ANALYSE, a query efetua dois *index nested loops*. O primeiro recorre aos índices ix\_orderlines\_orderid e products\_pkey, e o segundo efetua um *index only scan* usando orders\_pkey.

Tem-se os seguintes tempos, dados pela média de 10 valores, e incerteza dada pelo desvio padrão:

Planning time = 0,4285 ± 0,1043 ms

Execution time = 0,1278 ± 0,0484 ms

Conclui-se que não seria necessário adicionar outros índices, mas reorganizaram-se as condições na cláusula WHERE, de modo a facilitar a compreensão:

**SELECT SUM(OL.quantity)**

**FROM orderlines OL, orders O, products P**

**WHERE O.orderid = id\_order**

**AND OL.orderid = O.orderid**

**AND P.prod\_id = OL.prod\_id**

**AND P.category = 11;**

Com esta alteração, obtiveram-se tempos de planeamento e execução idênticos:

Planning time = 0,4522 ± 0,0959 ms

Execution time = 0,1322 ± 0,0373 ms

**(40)**

**SELECT SUM(P.price \* OL.quantity)**

**FROM orderlines OL, orders O, products P**

**WHERE O.orderid = OL.orderid**

**AND P.prod\_id = OL.prod\_id**

**AND O.orderid=id\_order --teve de se substituir por um valor concreto**

**AND P.category = 11;**

Segundo o EXPLAIN ANALYSE, esta query utiliza o mesmo procedimento que a anterior, pois as condições da cláusula WHERE são as mesmas.

Os tempos médios obtidos foram:

Planning time = 0,4492 ± 0,0806 ms

Execution time = 0,1373 ± 0,0369 ms

Novamente mudou-se a ordem das condições, de modo a facilitar a compreensão:

**SELECT SUM(P.price \* OL.quantity)**

**FROM orderlines OL, orders O, products P**

**WHERE O.orderid = id\_order**

**AND OL.orderid = O.orderid**

**AND P.prod\_id = OL.prod\_id**

**AND P.category = 11;**

E os tempos são idênticos:

Planning time = 0,4556 ± 0,1336 ms

Execution time = 0,1440 ± 0,0279 ms

**(48)**

**SELECT SUM(P.price \* OL.quantity)**

**FROM orderlines OL, orders O, products P**

**WHERE O.orderid = OL.orderid**

**AND P.prod\_id = OL.prod\_id**

**AND O.orderid=id\_order**

**AND P.category != 11;**

Segundo o EXPLAIN ANALYSE, a query efetua dois *index nested loops*. O primeiro recorre aos índices orders\_pkey (index only scan) e ix\_orderlines\_orderid, e o segundo efetua um *index only scan* usando products\_pkey.

Os tempos médios obtidos foram:

Planning time = 0,3018 ± 0,0515 ms

Execution time = 0,0904 ± 0,0117 ms

Usando o query com as condições reorganizadas:

**SELECT SUM(P.price \* OL.quantity)**

**FROM orderlines OL, orders O, products P**

**WHERE O.orderid = id\_order**

**AND Ol.orderid=O.orderid**

**AND P.prod\_id = OL.prod\_id**

**AND P.category != 11;**

Obteve-se:

Planning time = 0,3624 ± 0,0783 ms

Execution time = 0,1288 ± 0,0289 ms

**(68)**

**UPDATE orders**

**SET totalamount = ntotalamount --teve de se substituir por um valor concreto**

**WHERE orderid = id\_order; --teve de se substituir por um valor concreto**

Segundo o comando EXPLAIN ANALYSE, este query efetua um *index scan*, recorrendo ao índice *orders\_pkey*, pelo que não é necessário introduzir outro índice.

Os tempos médios obtidos foram:

Planning time = 0,1207 ± 0,0322 ms

Execution time = 0,1217 ± 0,0268 ms

**is\_it\_halloween**

**(86)**

**SELECT O.orderdate**

**FROM orders O**

**WHERE O.orderid = id\_order**

Segundo o comando EXPLAIN ANALYSE, este query usa o índice *orders\_pkey* para efetuar um *index scan,* portanto não é necessário criar mais índices.

Os tempos médios obtidos são:

Planning time = 0,1053 ± 0,0284 ms

Execution time = 0,0637 ± 0,0145 ms