CHRISTIAN DROZDOWICZ

StudentID: 832391

INFO20003

ASSIGNMENT 3

Question 1:

There are 100 tuples per page therefore:

Since Item = 160000 tuples; Let $\text{NPages(Item)} = \frac{160000}{100} = 1600\ Pages$

Since OrderItem = 200000 tuples; Let $\text{NPages(OrderItem)} = \frac{200000}{100} = 2000\ Pages$

And there are 802 Buffer pages in memory.

Question 1 Part a):

Consider Item as Outer relation

Cost(PNLJ)

$= NPages(Outer) + (NPages(Outer) * (NPages(Inner))$

$= NPages(Item) + (NPages(Item) * (NPages(OrderItem))$

$= 1600 + (1600\ * 2000)$

$= 1600 + 3200000$

$= 3201600\ (I/O)$

Question 1 Part b):

Consider Item as Outer relation

$$NBlocks(Outer) = NBlocks(Item) = \left\lceil \frac{\#\ of\ pages\ in\ Item}{Buffer - 2} \right\rceil = \left\lceil \frac{1600}{802 - 2} \right\rceil = \left\lceil \frac{1600}{800} \right\rceil = \lceil 2 \rceil = 2$$

Cost(BNLJ)

$= NPages(Outer) + (NBlocks(Outer) * (NPages(Inner))$

$= NPages(Item) + (NBlocks(Item) * (NPages(OrderItem))$

$= 1600 + (2\ * 2000)$

$= 1600 + 4000$

$= 5600\ (I/O)$

Question 1 Part c):

Consider 2 passes and let Item be Outer relation

Cost of sorting Item $= 2 * \# \, of \, passes * \# \, of \, pages \, of \, Item = 2 * 2 * 1600 = 6400 \, (I/O)$

Cost of sorting OrderItem $= 2 * \# \, of \, passes * \# \, of \, pages \, of \, OrderItem = 2 * 2 * 2000 = 8000 \, (I/O)$

Cost(SMJ)

$= Sort(Outer) + Sort(Inner) + NPages(Outer) + NPages(Inner)$

$= Sort(Item) + Sort(OrderItem) + NPages(Item) + NPages(OrderItem)$

$= 6400 + 8000 + 1600 + 2000$

$= 18000 \, (I/O)$

Question 1 Part d):

Let Item be Outer relation

Cost(HJ)

$= 2 * NPages(Outer) + 2 * NPages(Inner) + NPages(Outer) + NPages(Inner)$

$= 2 * NPages(Item) + 2 * NPages(OderItem) + NPages(Item) + NPages(OrderItem)$

$= 2 * 1600 + 2 * 2000 + 1600 + 2000$

$= 3200 + 4000 + 1600 + 2000$

$= 10800 \, (I/O)$

Question 1 Part e):

The optimal cost for this query would be achieved if each relation was read only once, this can be done by storing the entire smaller relation in memory and then just read in the larger relation page-by-page. That way for each tuple in the larger relation we search the smaller relation which is entirely in memory, and find the matching tuples.Since the smaller relation would have to be held entirely within the Buffer, as well as two extra pages: one page for reading in the larger relation, and another for an output buffer.

This can be calculated as:

Total Cost

$= NPages(Item) + NPages(OrderItem) = 1600 + 2000 = 3600(I/O)$

Then the minimum number of buffer pages for this cost would be

$= min\{NPgaes(Item), NPages(OrderItem)\} + 1 + 1$

$= min\{1600{,}2000\} + 1 + 1 = 1600 + 1 + 1$

$= 1602 \, Buffer \, Pages$

Question 2:

Student relation has 800 pages, each pages stores 80 tuples, wam = [0, 100], faculty has 10 items.

Question 2 Part a):

Calculate RF for wam:

$wam > 75$

The reduction factor is:

$$RF(wam) = \frac{High(col) - value}{High(col) - Low(col)} = \frac{100 - 75}{100 - 0} = \frac{25}{100} = 0.25$$

Calculate RF for faculty:

$faculty = 'Arts'$

The reduction factor is:

$$RF(faculty) = \frac{1}{NKeys(col)} = \frac{1}{NKeys(faculty)} = \frac{1}{10} = 0.1$$

Estimated Result size for whole query:

$NTuples(Student) = \#\ of\ Tuples\ per\ page * \#\ of\ pages = 80 * 800 = 64000\ tuples$

$RF(wam) = 0.25$

$RF(faculty) = 0.1$

$$ResultSize(Query) = NTuples(Student) * \prod_{i=1,2} RF_i = 64000 * 0.25 * 0.1$$

$= 1600\ tuples$

Question 2 Part b):

Clustered B+ tree index on (faculty, wam), Let there be 120 index pages

Since we have a clustered B+ tree index on both faculty and wam there are two possible access paths for this query:

- The clustered B+ tree index on (faculty, wam), with cost:
  1. $= \left(NPages(Index) + NPages(Relation)\right) * \prod_{i=1,2} RF_i$
  2. $= (120 + 800) * 0.25 * 0.1$
  3. $= 23\ (I/O)$

- Full table scan, with cost $= 800(I/O)$

Since other indexes are not applicable here, the cheapest access path is to use a clustered B+ tree with cost $= 23(I/O)$

Question 2 Part c):

Unclustered B+ tree index on (wam), Let there be 60 index pages

Since we have an unclustered B+ tree index on wam there are two possible access paths for this query:

- The unclustered B+ tree index on (wam), with cost:
  1. $= \left(NPages(Index) + NTuples(Relation)\right) * \prod_{i=1,2} RF_i$
  2. $= (60 + (80 * 800)) * 0.25$
  3. $= 64060 * 0.25$
  4. $= 16015\ (I/O)$

- Full table scan, with cost $= 800(I/O)$

Since other indexes are not applicable here, the cheapest access path is to use the full table scan cost $= 800(I/O)$

Question 2 Part d):

Unclustered hash index on (faculty).

For hash index, the size doesn't matter as for each tuples the cost is 2.2: 1.2 is for the bucket check and 1 is to fetch the page from disk.

Since we have an unclustered hash index on faculty there are two possible access paths for this query:

- The unclustered hash index on (faculty), with cost:
  1. $= Hash\ lookup\ cost * NTuples(Relation) * \prod_{i=1,2} RF_i$
  2. $= 2.2 * (80 * 800) * 0.1$
  3. $= 2.2 * 64000 * 0.1$
  4. $= 14080\ (I/O)$

- Full table scan, with cost $= 800(I/O)$

Since other indexes are not applicable here, the cheapest access path is to use the full table scan cost $= 800(I/O)$

Question 2 Part e):

Unclustered hash index on (wam).

For hash index, the size doesn’t matter as for each tuples the cost is 2.2: 1.2 is for the bucket check and 1 is to fetch the page from disk.

Since we have an unclustered hash index on wam there is only one possible access paths for this query:

- Cannot have an unclustered hash index on (wam) in this query, because wam is a range and it is not viable to use a hash index when looking at a range (wam>75)
- We still have the full table scan, with cost $= 800(I/O)$

Since other indexes are not applicable here, and hash index on wam will not work for this query, the cheapest access path is to use the full table scan cost $= 800(I/O)$

Question 3:

Let Customer = c, Order = o, OrderItem =oi.

There are 400 different ItemName.

Quantity of OrderItems [0, 50].

80000 OrderItems, 8000, Orders, 2000 Customers.

100 tuples per page.

Clustered B+ tree index on Order.orderid with 20 index pages

Clustered B+ tree index on Order.quantity with 200 index pages.

Question 3 Part a):

Calculate RF for oi.quantity:

$oi.quantity < 10$

The reduction factor is:

$$RF(oi.quantity) = \frac{value - Low(col)}{High(col) - Low(col)} = \frac{10 - 0}{50 - 0} = \frac{10}{50}$$

$= 0.2$

Calculate RF for oi.itemname:

$oi.itemname = 'Rotring\ 600'$

The reduction factor is:

$$RF(oi.itemname) = \frac{1}{NKeys(col)} = \frac{1}{NKeys(oi.itemname)} = \frac{1}{400}$$

$= 0.0025$

Calculate RF for c.cusid = o.cusid:

$c.cusid = o.cusid$

The reduction factor is:

$$RF(c.cusid = o.cusid) = \frac{1}{Max(NKeys(c.cusid), NKeys(o.cusid))} = \frac{1}{Max(2000{,}2000)} = \frac{1}{2000}$$

$= 0.0005$

Calculate RF for o.orderid=oi.orderid:

$o.orderid = oi.orderid$

The reduction factor is:

$$RF(o.orderid = oi.orderid) = \frac{1}{Max(NKeys(o.orderid), NKeys(oi.orderid)}$$

$$= \frac{1}{Max(8000,8000)} = \frac{1}{8000}$$

$= 0.000125$

Estimated Result size for whole query:

$NTuples(oi) = 80000$

$NTuples(o) = 8000$

$NTuples(c) = 2000$

$RF(oi.quantity) = 0.2$

$RF(oi.itemname) = 0.0025$

$RF(c.cusid = o.cusid) = 0.0005$

$RF(o.orderid = oi.orderid) = 0.000125$

$ResultSize(Query) = NTuples(oi) * NTuples(o) * NTuples(c) * all\ relevant\ RF's$

$= 80000 * 8000 * 2000 * 0.2 * 0.0025 * 0.0005 * 0.000125$

$= 40\ tuples$

Question 3 Part b):

$NPages(c) = \frac{2000}{100} = 20, NPages(o) = \frac{8000}{100} = 80, NPages(oi) = \frac{80000}{100} = 800$

[1]

First NLJ Cost(c JOIN o)

- $= NPages(c) + (NPages(c) * NPages(o))$
- $= 20 + (20 * 80)$
- $= 1620(I/O)$

Result Size

- $= NTuples(c) * NTuples(o) * all\ relevant\ RF's$
- $= 2000 * 8000 * 0.0005$
- $= 8000\ Tuples$
- $= \frac{8000}{100} Pages = 80\ Pages$

Second NLJ Cost((c JOIN o) JOIN oi)

- $= 0 + (80 * 800)$
- $= 64000(I/O)$

Therefore Total Cost $= 65260(I/O)$

[2]

First HJ Cost(o JOIN oi)

- $= (3 * NPages(o)) + (3 * NPages(oi))$
- $= (3 * 80) + (3 * 800)$
- $= 1620(I/O)$

Result Size

- $= NTuples(o) * NTuples(oi) * all\ relevant\ RF's$
- $= 8000 * 80000 * 0.000125$
- $= 80000\ Tuples$
- $= \frac{80000}{100} Pages = 800\ Pages$

Second SMJ Cost((o JOIN oi) JOIN c)

Due to pipelining we obtain:

- $= \sout{NPages(o\ JOIN\ oi)} + NPages(c) + (2 * NPages(o) * 2) + (2 * NPages(c) * 2)$
- $= 20 + (4 * 800) + (4 * 20)$
- $= 3300(I/O)$

Therefore Total Cost $= 4920(I/O)$

[3]

First SMJ Cost((oi JOIN o) JOIN c)

Due to already sorted on index we obtain we obtain:

- $= Sort(oi) +$ ~~$Sort(o\ through\ index)$~~ $+ read(oi) + read(o\ through\ index)$
- $= (2 * NPages(oi) * 2) +$ ~~$(2 * NPages(o\ through\ index) * 2)$~~ $+ NPages(oi) + NPages(o\ though\ index)$
- $= (4 * 800) + 800 + 20$
- $= 4020(I/O)$

Result Size

- $= NTuples(oi) * NTuples(o) * all\ relevant\ RF's$
- $= 8000 * 80000 * 0.000125$
- $= 80000\ Tuples$
- $= \frac{80000}{100} Pages = 800\ Pages$

Second HJ Cost((oi JOIN o) JOIN c)

Due to pipelining we obtain:

- $= (\cancel{3} * NPages(oi\ JOIN\ o)) + (3 * NPages(c))$
- $= (2 * NPages(oi\ JOIN\ o)) + (3 * NPages(c))$
- $= (2 * 800) + (3 * 20)$
- $= 1660(I/O)$

Therefore Total Cost $= 5680(I/O)$

[4]

First NLJ Cost(oi JOIN o)

- $= NPages(o) + (NPages(I(oi)) * NPages(o))$
- $= 80 + (200 * 80)$
- $= 16080(I/O)$

Result Size

- $= NTuples(oi) * NTuples(o) * all\ relevant\ RF's$
- $= 80000 * 8000 * 0.2 * 0.000125$
- $= 16000\ Tuples$
- $= \frac{16000}{100} Pages = 160\ Pages$

Second HJ Cost((oi JOIN o) JOIN c)

Due to pipelining we obtain:

- $= (\cancel{3} * NPages(oi\ JOIN\ o)) + (3 * NPages(c))$
- $= (2 * NPages(oi\ JOIN\ o)) + (3 * NPages(c))$
- $= (2 * 160) + (3 * 20)$
- $= 380(I/O)$

Therefore Total Cost $= 16460(I/O)$