Assignment 3
Database Systems : INFO 200003, Semester1, 2019

Name: Syed Ahammad Newaz Saif
Student ID: 684933

Question - 1 :

Given,

- Item has 160,000 tuples.
- OrderItem has 200,000 tuples.
- Both Item and Order Item have 100 tuples per page.
- NPages(Item) = 160,000/100 = 1600 pages.
- Npages(OrderItem) = 200,000/100 = 2000 pages.
- Block size = 802 buffer pages.

a) Page-oriented Nested Loops Join. Consider Item as the outer relation.

- Cost(PNLJ)
  - = NPages(Outer) + ((NPages(Outer))*(NPages(Inner)))
  - = NPages(Item) + ((NPages(Item))*(NPages(OrderItem)))
  - = 1600 + 1600 * 2000
  - = 3,201,600 I/Os

b) Block-oriented Nested Loops Join. Consider Item as the outer relation.

- NBlocks(Outer)
  - = (*NPages*(*Outer*))/(*Block size*−2)
  - = (1600) / (802−2)
  - = 2
- Cost(BNLJ)
  - = NPages(Outer) + ((NBlocks(Outer)) * (NPages(Inner)))
  - = 1600 + 2*2000
  - = 5,600 I/Os

c) Sort-Merge Join. Assume that Sort-Merge Join can be done in just 2 passes.

- Sort(R)
  - = 2 * Number Passes * NPages(R) Sort(Item)
  - = 2 * 2 * 1600 = 6,400 Sort(OrderItem)
  - = 2 * 2 * 2000
  - = 8,000

- Cost(Merging Item & OrderItem)
  = NPages(Item) + NPages(OrderItem)
  = 1600 * 2000 Cost(Merging Item & OrderItem)
  = 3,600

- Cost(SMJ)
  = Sort(Item) + Sort(OrderItem) + Cost(Merging Item & OrderItem)
  = 6400 + 8000 + 3600
  = 18,000 I/Os

d) Hash Join

- Cost(HJ)
  = 2 * NPages(Outer) + 2 * NPages(Inner) + NPages(Outer) + NPages(Inner)
  = 3 * NPages(Item) + 3 * NPages(OrderItem)
  = 3 * 1600 + 3 * 2000
  = 10,800 I/Os

e) What would be the lowest possible cost to perform this query, assuming that no indexes are built on any of the two relations, and assuming that sufficient buffer space is available? What would be the minimum buffer size required to achieve this cost? Explain briefly.

It seems that the lowest cost till now is the Block-oriented nested loop join from part (b) with 5,600 I/Os. Even lower cost is achievable by reading each relation once, so having a buffer size that stores the smallest relation. The lowest cost thereby becomes :

- Lowest cost
  = NPages(outer) + NPages(inner)
  = 1600 + 2000
  = 3600 I/Os
- Buffer size
  = 2 + min (NPages(inner), NPages(outer))
  = 1602 pages

Question - 2 :

Given,

- Schema - Student (studentid, firstname, lastname, faculty, level, wam)
- SQL statement -
  SELECT *
  FROM Student
  WHERE wam > 75 **AND** faculty = 'Arts';
- Student relation has 800 pages
- Each page stores 80 tuples
- Faculty attribute can take one of ten values (“Arts”, “Architecture”, “Business”, “Education”, “Fine Arts and Music”, “Law”, “Medicine”, “Science”, “Agricultural Science”, “Engineering”)
- wam (Weighted Average Mark) can have values between 0 and 100 ([0, 100])

a) Compute the estimated result size for the query, and the reduction factor of each filter.

- RF (faculty = ‘Arts’)
  = 1/10
  = 0.1

- RF(wam > 75)
  = ((*High*(*col*))− (*value*)) / ((*High*(*col*))− (*Low*(*col*)))
  = 100−75 100−0
  = 0.25

- Result size
  = (NTuples(Student)) * (RF(faculty)) *(RF(wam))
  = (64,000 * 0.1 * 0.25)
  = 1,600 tuples

b) Compute the estimated cost of the best plan assuming that a clustered B+ tree index on (faculty, wam) is the only index available. Suppose there are 120 index pages. Discuss and calculate alternative plans.

- Cost
  = RF(faculty) * RF(wam) * (NPages(R) + NPages(I))
  = 0.1 * 0.25 * (800 + 120)
  = 23 I/Os

  Alternative Plan :
  Full table scan
  = NPages(R)
  = 800 I/Os > 23 I/Os in clustered B+ tree

c) Compute the estimated cost of the best plan assuming that an unclustered B+ tree index on (wam) is the only index available. Suppose there are 60 index pages. Discuss and calculate alternative plans.

- Cost
  = RF(wam) * (NTuples(R) + NPages(I))
  = 0.25 * (64000 + 60)
  = 16,015 I/Os

  Alternative Plan :
  Full table scan
  = NPages(R)
  = 800 I/Os < 16,015 I/Os in clustered B+ tree index on wam so it is more cost-effective solution

d) Compute the estimated cost of the best plan assuming that an unclustered hash index on (faculty) is the only index available. Discuss and calculate alternative plans.

- Cost
  = NTuples(R) * RF(faculty) * 2.2
  = 64000 * 0.1 * 2.2
  = 14,080 I/Os

  Alternative Plan :
  Full table scan
  = NPages(R)
  = 800 I/Os < 14,080 I/Os is more cost-effective solution

e) Compute the estimated cost of the best plan assuming that an unclustered hash index on (wam) is the only index available. Discuss and calculate alternative plans.

For wam greater than 75, there will be no index returned so an unclustered hash index cannot be used on the range of the weight average marks which will cost more. In order to make the plan more cost-effective, we can use a the following alternative plan:

- Full table scan
  = NPages(R)
  = 800 I/Os

Question - 3:

- **Schema -**
  Customer (cusid: integer, postcode: char(4), name: char(20)) Order (orderid: integer, orderdate: date, cusid: integer, departmentid: integer) OrderItem (orderitemid: integer, handling: char(20), orderid: integer, itemname: char(15), quantity: integer)
- **Query** -
  SELECT c.postcode, o.departmentid
  FROM Customer c, Order o, OrderItem oi
  WHERE c.cusid = o.cusid **AND** o.orderid = oi.orderid **AND** oi.quantity < 10 **AND**
  oi.itemname = 'Rotring 600';

a) Compute the estimated result size and the reduction factors (selectivity) of this query.

- C:
  - NPages(c)
    = 2000/100
    = 20
  - NTuplesPerPage(c)
    = 100
  - NKeys(c)
    = 2000
- O:
  - NPages(o)
    = 80
  - NTuplesPerPage(o)
    = 100
  - NKeys(o) = 8000
- OI:

- NPages(oi) = 80000/100 = 800
- NTuplesPerPage(oi) = 100
- NKeys(oi) = 400

- ITEM RANGE = [0, 50]

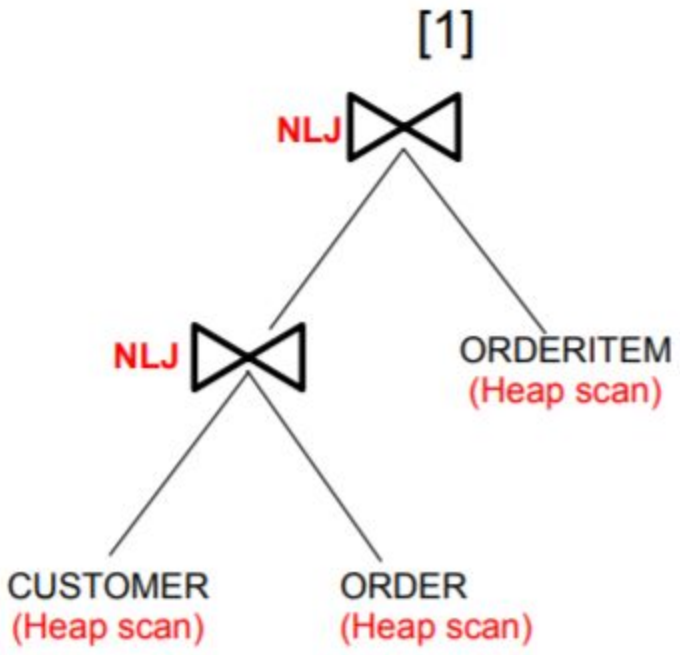
Reduction factors

- RF(c.cusid = o.cusid)
  = 1 /(*Max* (*NKeys*(*c*),*NKeys*(*o*)))
  = 1/2000
  = 0.0005

- RF(o.orderid = oi.orderid)
  = 1/ 2000
  = 0.0005

- RF(oi.quantity < 10)
  = (*value*−*Low*(*Col*)) / ((*High*(*col*))−(*Low*(*col*)))
  = 10 − 0 50−10
  = 0.25
- RF(oi.itemname = 'Rotring 600')
  = 1/ 400
  = 0.0025
- Expected result size
  = ((NTuples(c)) * (NTuples(o)) * (NTuples(oi))* (Reduction factors))
  = (2000 * 8000 * 80000 * 0.0005 * 0.0005 * 0.25 *0.0025)
  = 200 Tuples

b) Compute the cost plans. Assume that sorting of any relation can be done in 2 passes. NLJ is a Page-oriented Nested Loops Join. Assume cusid is the candidate key of Customer relation, ordered is the candidate key of the Order relation. Assume 100 tuples of a resulting join between Customer and Order fit in a page. Same for Order and OrderItem. If selection over filtering predicates is not marked in the plan, assume it will happen on-the-fly after all joins are performed, as the last operation in the plan.



[1]

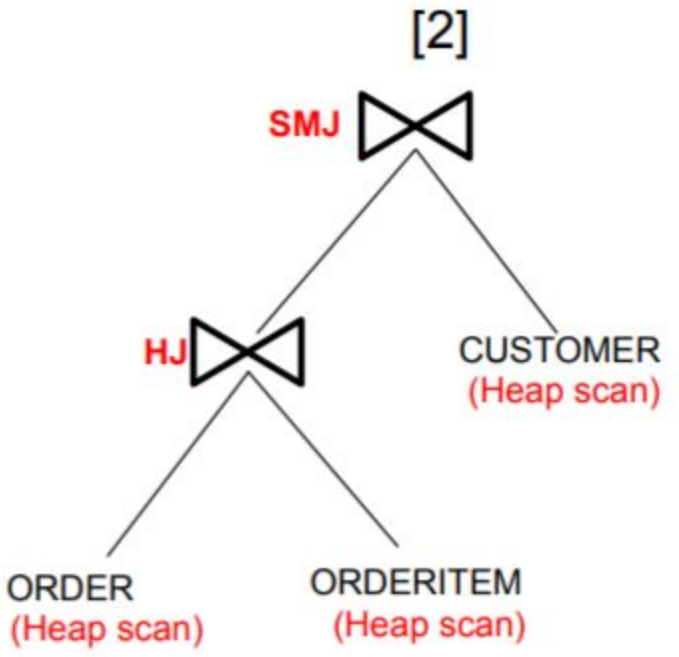
- NTuples for Customer JOIN Order
  = ((NTuples(Customer)) * (NTuples(Order))) / (*NKeys*(*I*))
  = (2000 * 8000) / (2000)
  = 8000 Tuples

- **Size of Customer JOIN Order**
  - **NPages(c X o)**
    = 8000/100
    = 80 pages

- Cost(c X o) NLJ
  = NPages(Outer) + NPages(Outer)*NPages(Inner)
  = 20 + 20*80 = 1620 I/Os

- Cost((c X o) X oi) NLJ
  = NPages(c X o) * NPages(oi)
  = 80 * 800
  = 64,000 I/Os

- Total cost
  = 1620 + 64000
  = 65,620 I/Os



[2]

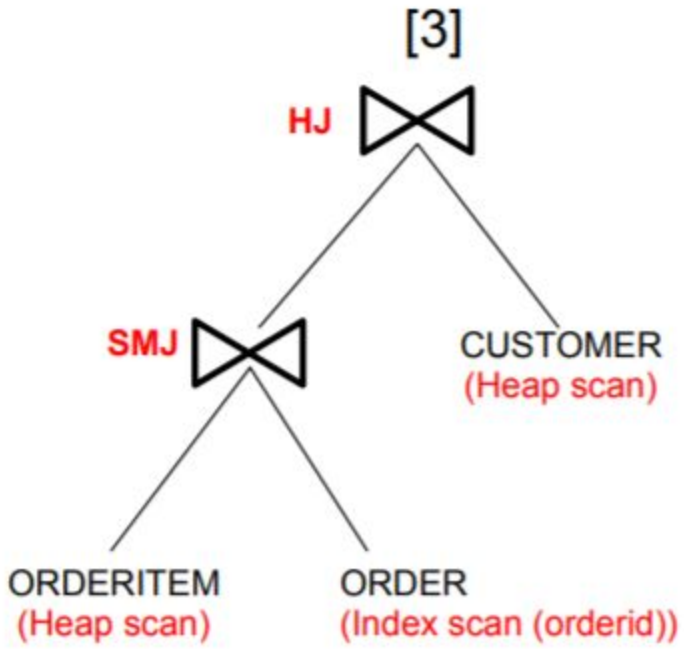
- NTuples for Order JOIN OrderItem
  = ((NTuples(OrderItem)) * (NTuples(Order))) / (*NKeys*(*I*))
  = (80000 * 8000) / (2000)
  = 80000 Tuples

- **Size of Customer JOIN Order**
  - **NPages(c X oi)**
    = 80000/100
    = 800 pages

- Cost(o X oi) HJ
  = ((3*NPages(Outer)) + (3*NPages(Inner)))
  = ((3*80) + (3*800))
  = 2640 I/Os

- Cost((o X oi) X c) SMJ
  = ((2 * NPasses * NPages(o X oi))+ NPages(o X oi) + NPages(c))
  = ((2 * 2 * 800) + 800 + 20)
  = 4,020 I/Os

- Total cost
  = 2640 + 4020
  = 6,660 I/Os



[3]

- NTuples for Order JOIN OrderItem
  = ((NTuples(OrderItem)) * (NTuples(Order))) / (*NKeys*(*I*))
  = (80000 * 8000) / (2000)
  = 80000 Tuples

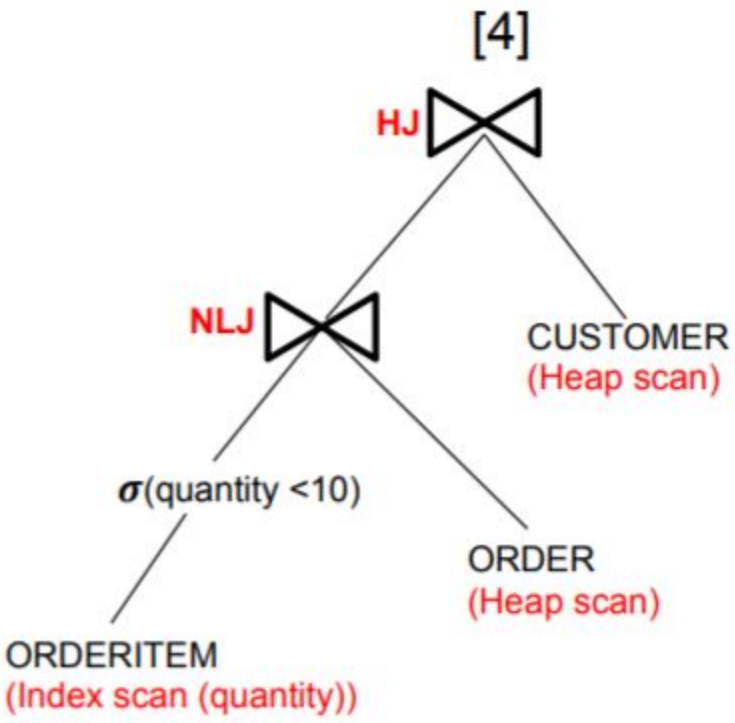
- **Size of Customer JOIN Order**
  - **NPages(oi X c)**
    = 80000/100
    = 800 pages

- Cost(oi X o) SMJ
  = sort(oi) + sort(o) + cost(merging oi with o)
  = ((2 * NPasses * NPages(oi)) + 0 (already sorted) + NPages(oi) + NPages(o))
  = (2 * 2 * 800) + 800 + 20

= 4,020 I/Os

- Cost((o X oi) X c) HJ
  = (3*NPages(Outer) + 3*NPages(Inner))
  = (3*NPages(o X oi) + 3*NPages(c))
  = ((3*800) + (3*20))
  = 2,460 I/Os

- Total cost
  = 4020 + 2460
  = 6,480 I/Os



[4]

- NTuples for Order JOIN OrderItem
  = ((NTuples(OrderItem)) * (NTuples(Order))) / (*NKeys*(*I*))
  = (80000 * 8000) / (2000)

= 80000 Tuples

- **Size of Customer JOIN Order**
  - **NPages(oi X o)**
    = 80000/100
    = 800 pages

- Cost(o X oi) NLJ
  = (NPages(Outer) + (NPages(Outer)*NPages(Inner)))
  = (80 + (80*800))
  = 64,080 I/Os

- Cost((o X oi) X c) HJ
  = ((3 * NPages(Outer)) + (3 * NPages(Inner)))
  = ((3 * 800) + (3 * 20))
  = 2460 I/Os

- Total cost
  = 64080 + 2460
  = 66,540 I/Os