John Rivera

Homework 8

Database Systems

1.

a) "Block-nested loop join for R ⋈ S with M=101 blocks. For join ordering, choose the lowest cost one and only show the result of that join."

Lowest cost ordering => outer relation has less pages

$$2000 + \left\lceil \frac{2000}{100} \right\rceil * 6000 = \mathbf{122000}$$

b) "External sorting of S using M=60 blocks."

Step 1: read PAGES(S) into memory (fits only 60 at a time)
=> Cost is 6000
=> Creates 6000/60 = 100 sorted groups
Write groups to disk => 6000 pages

Total Step 1 Cost = 12000

Step 2: 100 sorted groups does not fit in M => must further divide
⇨ Read 60 groups into memory, then 40 groups => 6000
⇨ Creates 2 sorted groups
⇨ Write them out to disk => cost of 6000

Total Step 2 Cost = 12000

Step 3: 2 sorted groups fir in M
⇨ Read 6000 pages and output
⇨ Cost = 6000

TOTAL COST = 30,000

c) "External sorting of S using M=100 blocks."

Step 1: PAGES(S)/M = 6000/100 = 60 sorted groups
⇨ Cost = 1200

Step 2: 60 < 100 => fits in memory
Merge
Cost = 6000
TOTAL COST= 18,000

d) “Hash join for R ⋈ S with M=101 blocks”

First Hash the relations to same buckets

R -> $\frac{PAGES(R)}{M-1} = \frac{2000}{100} = 20\ pages\ per\ bucket$

S -> $\frac{6000}{100} = 60\ pages\ per\ bucket$

Cost of Hashing = $2 * PAGES(R) + 2 * PAGES(S) = 16{,}000$

To join, must read each bucket into memory

- ⇨ Each bucket contains 60 + 20 = 80 pages
- ⇨ 80 pages fits into 100 memory -> can join right away
- ⇨ So simply read buckets into memory one at a time and output the join
  - ○ Read cost = 2000 + 6000 = 8,000

TOTAL COST = 24,000

e) “Sort merge join for R ⋈ S with M=101 blocks”

First, Sort R

Step 1: 2000/100 = 20 groups -> read and write => Cost = 4,000

Step 2: merge

- ○ Read and write back to disk to free up memory for the next step
  - ▪ Cost = 4,000

Cost for Sorting R = 8000

Second, sort S

Step 1: 6000/100 = 60 groups -> read + write => cost = 12,000

Don’t finish individually sorting S

Combine sort and join

Allocate 1 memory for reading R

Allocate 60 memory for reading 1 page from each sorted group(can use sorted order even though not fully ordered on disk)

Output Joins

Read + Output PAGES(R) and PAGES(S) once each

Cost = 2000 + 6000 = 8000

TOTAL COST = 8000 + 12000 + 8000 = 28,000

2.

a) “Plan 1: sequential scan over R”

TOTAL COST = PAGES(R) = 2,000

b) “Plan 2: using index I1”

800 nodes at leaf level
TUPLES(R) = 100,000
Tuples per node = 100,000/800 = 125 tuples per node

Index Scan Cost = 1 root + 1 internal + (1 – 2) = 3 – 4

- Tuples(R.C > 10 and R.D = 25) = 50 tuples => fit in 1 or 2 leaf nodes

Data Cost = 50

- 50 tuples in 50 pages at most

TOTAL COST = 53 – 54

c) “Plan 3: using index I2”

1,500 nodes at leaf level
In this case, we have to scan all the nodes where D = 25

- TUPLES(R.D = 25) = 1,000

Index Scan Cost = $1 + 1 + \left(\frac{1{,}000}{100{,}000}\right) * 1{,}500 =$ 17 – 18

Both A and B (what is returned) are index attributes so no need to look in disk

TOTAL COST = 17 – 18

d) “Plan 4: using index I3”

300 nodes in leaf level
In this case we have to scan nodes where R.C > 10 and check in page to see if it fits criteria

- TUPLES(R.C > 10) = 20,000 tuples

Index Scan Cost = $1 + 1 + \left(\frac{20{,}000}{100{,}000}\right) * 300 =$ 62 – 63

For every C > 10, we must look in page to see if R.D = 25

- If disk pages not sorted and we reread pages => 20,000 pages at worse
- If we don’t reread pages => 2,000 pages at worse

TOTAL COST = 2,062 – 2,063 or 20,062 – 20,063

e) “Plan 5: using index I4”

250 nodes at leaf level
In this case we scan the nodes of D=25 and go into the pages to see if C>10

Index Scan Cost = $1 + 1 + \left(\frac{1{,}000}{100{,}000}\right) * 250 =$ 5 – 6

Go into Pages of every D=25 tuple and see if C > 10

- Data Cost = 1,000

TOTAL COST = 1,005 – 1,006

f) **“**Plan 6: using index I3 and I4 both.”

Index Scan Cost For I3 = 63 – 63
Index Scan Cost for I4 = 5 – 6

Intersection and then search pages for tuples in the intersection

- Data Cost = 50

TOTAL COST = (62 – 63) + (5 – 6) + 50 = 117 – 119

3.

Q1)TUPLES(Games) * SEL(id = 21) = 10,000 * (1/40) = **250**

Q2) 30,000 * (1/10,000) = **3**

Q3) 30,000 * (1/3,000) = **10**

Q4) 30,000 * (1/10,000) * (1/3,000) = .001 = **0**

Q5) 10,000 * 30,000 * (1/max(10000, 10000)) * (1/3,000) = **10**

Q6) 740,000 * (1/2) = **370,000**

Q7) 740,000 * 1/3,000 * 1/10,000 = 0.024667 = **0**

Q8) 740,000 * (1 – ((1 – 1/30,000) (1 – 1/10,000))) = 320.64 = **321**

Q9) 740,000 * 30,000 * 1/max(3000, 3000) = **7,400,000**

Q10) 740,000 * 30,000 * (1/max(3000, 3000)) * (1/max(10000, 10000)) = **740**

4.

PLAN 1:

COST OF BLNJ = $PAGES(R) + PAGES(S) * \left\lceil \frac{PAGES(R)}{M-1} \right\rceil = 100 + 800 * \left\lceil \frac{100}{50} \right\rceil = 1700$

COST OF SELECTION = 0

SORT BY R.A, R.C:

Takes input from Selection. Selection outputs 175 pages

Step 1: So it takes 175 pages from selection => writes 175 pages as 3 groups

- COST = 175

Step 2: 3 < M = 50, so read sorted groups from disk and sort and output

- COST = 175

COST OF SORT = 350

COST OF PROJECT = 0

TOTAL COST OF PLAN 1 = 1700 + 350 = 2050

PLAN 2:

INDEX SCAN COST = $1 + \left(\frac{1}{4}\right) * 100 + 0 = 26$

- Data cost is 0 because all of the needed attributes are in index

BNLJ:

- Takes input from Index scan
- Index scan outputs 25 to it. We have 25 + 1 memory available for use
- So for Block nested loop join, we only need to read S once, sort and output
- COST OF BNLJ = 350

Rest of steps are equal to Plan 1
COST OF SORT BY = 350

TOTAL COST OF PLAN 2 = 350 + 800 + 26 = 1176

PLAN 3:

SELECT R.B > 20 SEQUENTIAL SCAN COST = PAGES(R) = 100

SORT BY R.A, R.C:

- SELECT R.B > 20 outputs 25 tuples, 25 < 50 -> can sort in one step w/o using disk
- COST = 0

SORT BY S.D:

Step 1: Read PAGES(S) into memory, sort, output sorted groups to disk
This will create 800/50 = 16 sorted group
Step 1 Cost = 2 * PAGES(R) = 1600

Step 2: Read groups to memory, sort, output to pipeline
Step 2 cost = 800

Total sort by S.D cost = 1600 + 800 = 2400

SMJ:

Takes input from sort steps, one page from each sort at a time as they happen (combining merge and sort steps), and then joins and outputs them
COST = 0

TOTAL COST OF PLAN 3 = 2400 + 100 = 2500