CMPT 454 Assignment 3 Solution

Question 1

For each of the operations described below you are to calculate the number of disk reads (and writes) to perform the operation in the most efficient way. Do not include the cost to write out the result of the operation. Briefly explain the process of each operation and its components.

For example, *use the index on a, read x nodes of the index and y records of the file*.

Assume the root node of tree indexes and directory of extensible hash indexes are *not* held in main memory at the start of the operation. [2 marks each, unless noted otherwise]

Product = {*pid*, *pname*, *ptype*, *pnumber*, *description*, *manufacturer*, *mcountry*, *mcity*, *price*}

Table Statistics (note for all V the table is the Product table)

B(Product)	T(Product)	V(*pid*)	V(*pname*)	V(*ptype*)	V(*pnumber*)
25,000	250,000	250,000	50,000	500	1,000
V(*mcountry*)	**V(*mcity*)**	**V(*manufacturer*)**	**V(*price*)**	**V(*description*)**	
40	1,000	10,000	50,000	125,000	
V({*ptype*, *pnumber*})		**V({*mcountry*, *mcity*})**			
250,000		2,500			

Indexes

- Primary dense B+ tree index on {*ptype*, *pnumber*} where *ptype* is the prefix of the search key. The index is height 4 (includes leaf and root level) and interior and leaf nodes contain 50 search keys on average.
- B+ tree index on {*mcountry*, *mcity*} where *mcountry* is the prefix of the search key. The index is height 5 (includes leaf and root level) and interior and leaf nodes contain 30 search keys on average.
- B+ tree index on *pname*. The index is height 4 (includes leaf and root level) and interior and leaf nodes contain 60 search keys on average.
- Extensible hash index on *pid*. The directory resides on two disk blocks and each bucket contains 100 search keys on average.
- Linear hash index on *manufacturer*. Each bucket contains 40 search keys on average – there are no overflow blocks.

Marking – 2 marks for each question (except l). In each case there is 1 mark for the noted explanation, give ½ mark if some of the details are missing. Acceptable values for the numeric answer are noted in each question.

a) $\sigma_{(ptype = 'ABC' \wedge pnumber = 13)}$ (Product) – use the {*ptype*, *pNumber*} index. There is 1 matching record Cost: 4 (height) + 1 (record) = 5. 1 mark for 5

b) $\sigma_{(ptype = 'ABC')}$ (Product) – use the {*ptype*, *pNumber*} index. There are (250,000 / 500) 500 matching records on 50 or 51 blocks. Cost: 4 (height) + 50 or 51 (blocks of records) = 54 or 55. 1 mark for 54 or 55

c) $\sigma_{(pid = 123456 \vee pid = 678324)}$ (Product) – use the *pid* index twice, the directory only needs to be read for the first selection. Cost: 4 (directory and 2 buckets) + 2 (records) = 6. 1 mark for 6, ½ mark for 8

d) $\sigma_{(mcountry = 'UK' \wedge mcity = 'Sheffield')}$ (Product) – use the {*mcountry*, *mcity*} index. There are (250,000 / 2,500) 100 matching records. Cost: 5 (height) + 4 (additional leaves) + 100 (records) = 109. 1 mark for 108, ½ mark for 107 or 110

e) $\sigma_{(mcity = 'Detroit')}$ (Product) – mcity is not a prefix of the {*mcountry*, *mcity*} index, so the index cannot be used so a file scan is required. Cost: 25,000. 1 mark for 25,000

f) $\sigma_{(pname = 'foo345' \vee manufacturer = 'acme')}$ (Product) – use the *pname* and *manufacturer* indexes; either independently or by taking the union of the *rid*s. There are (250,000 / 50,000) 5 matching *pname* records and (250,000 / 10,000) 25 matching *manufacturer* records. Cost: 4 (or 5) (*pname* index) + 5 (*pname* records) + 1 (*man*. index) + 25 (*man*. records) = 35 (36). 1 mark for 35 or 36, ½ mark if at least half of the calculation is correct

g) $\sigma_{(mcountry = 'Canada' \wedge price > 25.00 \wedge description = 'sweet widget')}$ (Product) – no index on price or description so use the {*mcountry*, *mcity*} index. There are (250,000 / 40) 6,250 matching records. Cost: 4 + (6,250 / 30) 209 (leaf nodes) + 6,250 = 6,463. 1 mark for 6,462 or 6,463, ½ mark for 6,460, 6,461 or 6,464

h) $\sigma_{(pname = 'bar111' \vee price = 73.80 \vee ptype = 'TLR')}$ (Product) – disjunction and no index on prices so a file scan is required. Cost: 25,000. 1 mark for 25,000

i) $\sigma_{(\ (pname = 'foo17' \vee price = 19.99) \wedge\ (ptype = 'HJK' \vee price = 19.99)\ \wedge\ (ptype = 'HJK' \vee pid = 432911)\ )}$ (Product) – the first two conjuncts include a disjunction on price so cannot use the indexes. Use the {*ptype*, *pNumber*} and *pid* indexes; either independently or by taking the union of the *rid*s. There are (250,000 / 500) 500 matching *ptype* records on 50 or 51 blocks and 1 matching *pid* record. Cost: 4 (*ptype* index) + 50 or 51 (blocks of *ptype* records) + 3 (*pid* index) + 1 (*pid* record) = 59 (58). 1 mark for 58 or 59, ½ mark if at least half of the calculation is correct

j) Sort the Product table assuming there are 100 main memory frames available – block size is 25,000, after first sort pass there are 250 sorted runs of size 100, after first merge pass there are 3 sorted runs, so can sort in three passes. Cost: 25,000 * 5 = 125,000. 1 mark for 125,000, ½ mark for 150,000

k) $\pi_{(ptype,\ description)}$ (Product) – assume there are 50 main memory frames available, and that duplicates are *not* to be removed – scan the file and remove unwanted columns. Cost: 25,000. 1 mark for 25,000

l) $\pi_{(ptype,\ description)}$ (Product) – assume there are 50 main memory frames available, and that duplicates are to be removed using sort projection; also assume that duplicates are only encountered in the final stage of the process [4 marks] – [2 marks for explanation, 2 marks for calculation] – scan file, remove unwanted attributes and sort results. Each sorted record is size 100 scan fit 40 on a block (takes up 6,250 blocks). After first pass there are 125 sorted runs of size 50 (more accurately size 48 or 49). After the first merge pass there are 3 sorted

runs. So only requires two merge passes. Cost = 25,000 (R) + 6,250 (W) + 12,500 (RW) + 6,250 (R) = 50,000. 2 marks for 50,000, 1 mark for 56,250

Note that *ptype* is 4 bytes, *description* is 96 bytes and pages are 4,096 bytes

Question 2

Answer the following questions about performing a natural join between the *Patient* and *Visit* tables. The only attribute the two tables have in common is *msp*, which is the primary key of *Patient* and a foreign key in *Visit*. Relevant information is shown to the right. [15]

	Visit	Patient
T(R)	1,800,000	180,000
B(R)	180,000	18,000
V(R, msp)	180,000	180,000

a) How many records will the joined relation contain? [1] – 1,800,000.

b) Approximately how many records of the joined relation fit on a single block? [1] – Each joined record is approximately twice the size so 5.

c) What is the main memory requirement (in frames) to perform the join in two passes using the *sort-join* algorithm? Briefly explain your calculation. [2] – To achieve this in two passes there must be sufficient frames to maintain an input buffer for each sorted run after the initial pass (the sort pass) and one frame for an output buffer. The requirement is therefore *sqrt*(B(*Patient*) + B(*Visit*)) = *sqrt* (180,000 + 18,000) = 445+1 (output) = 446. 1 mark for 446 (½ for 445 or 447) and 1 mark for the explanation

d) If *Patient* was already sorted on *msp* would your answer to part (**c**) change? Explain why or why not? [2] – Yes, because we only need one input buffer for *Patient* (as it already consists of a single sorted run), so the answer is *sqrt* (180,000) + 2 = 427. 1 mark for 427 (½ for 425, 3426 or 428) and 1 mark for the explanation

e) What is the main memory requirement (in frames) to perform the join in two passes using the *hash-join* algorithm? Briefly explain your calculation. [2] – The memory requirement is for one partition of the smaller relation, an input buffer for the larger relation, and an output buffer. So, the answer is *sqrt* (18,000) + 2 = 137. 1 mark for 136 or 137 (½ for 135 or 138) and 1 mark for the explanation

f) Assume that there are approximately 1,200 frames available for performing the join; what is the cost of performing the join using the *block nested loop join* algorithm? Briefly explain your calculation. [3] – For the block nested loop join fill as much of main memory as possible with blocks of the smaller relation and scan the larger relation. 1,200 blocks for *Patient* input. Scan *Visit* 15 times for a cost of 18,000 + 15(180,000) = 2,718,000. 1 mark for the answer and 1 mark for the explanation

g) Assume that there is a secondary extensible hash index on *msp* in *Patient*. If the directory page of the index is retained in main memory, what is the cost of performing the join using the *index nested loop join* algorithm? Briefly explain your calculation. [3] – To perform an index nested loop join, scan the outer relation (the one with no index) while using the index on the inner relation. The cost is the cost of scanning the outer relation (180,000) + the cost of using the index for each record in the outer relation so read bucket of the index for each

record of *Visit* which joins to just one record of *Patient*) for a total cost of 180,000 + 1,800,000 * 2 = 3,780,000. 1 mark for the answer and 1 mark for the explanation

h) Assume that there is a primary B+ tree index of height 3 on *msp* in *Visit*. If the root node of the index is retained in main memory, what is the cost of performing the join using the *index nested loop join* algorithm? Briefly explain your calculation. [3] – Cost of scanning the outer relation (18,000) + the cost of using the index for each record in the outer relation (180,000 * 2 for the non-root levels of the index where each record of *Patient* joins to 10 records of *Visit* on 2 blocks of *Visit* each) for a total cost of 18,000 + 180,000 * 4 = 738,000. 1 mark for the answer (½ for 310,000) and 1 mark for the explanation

i) Assume that there are just over 4,000 frames available for performing the join; what is the cost of performing the join using the *hybrid hash join* algorithm where one partition of the outer relation is to be retained in main memory? Briefly explain your calculation. [3] – The smaller relation should be partitioned into 5 partitions of size 3.600, one of which is retained throughout the partitioning stage for both *Patient* and *Visit*. The retained partition of *Patient*, and the matching records of *Visit* only have to be read once rather than being read and written (partitioning step) and then read again (probing step). Cost is therefore 18,000 + 180,000 + 2*((180,000 + 18,000) *4/5) = 514,800. 1 mark for the answer and 1 mark for the explanation

Assessment

The assignment is out of 46. Marks are assigned as follows:

- Question 1 – 26
- Question 2 – 20

John Edgar (johnwill@sfu.ca)