Data Management – AA 2016/17 (06/02/2017) – exam B

Problem 1
Suppose that we have only 3 buffer frames available, and we have to compute the difference (producing a result without duplicates) of two SQL tables (it is well known that an SQL table may contain duplicates). Describe in detail the algorithm you would use, and tell which is the cost of the algorithm in terms of number of page accesses.

Problem 2
A schedule S is said to be *strange* if it satisfies the following two conditions: (i) every transaction in S that does not contain any "read" action contains exactly one "write action", and every transaction that contains at least one "read" action does not contain any "write" action; (ii) no element of the database is read more than once in S, and no element of the database is written more than once in S. Prove or disprove the following propositions:

1. Every strange schedule is accepted by the timestamp-based scheduler.
2. Every strange schedule is conflict serializable.
3. Every strange schedule is view serializable.

Problem 3
Consider the following schedule

$$S = r_1(z)\, r_1(y)\, w_3(y)\, r_1(x)\, r_2(x)\, c_1\, w_4(z)\, w_2(x)\, w_3(x)\, c_3\, r_4(u)\ c_4\, w_2(u)\, c_2.$$

5.1 Tell whether S is a 2PL schedule (with shared and exclusive locks) or not, explaining the answer in detail.

5.2 Tell whether S is view-serializable or not, explaining the answer in detail.

5.3 Describe the behavior of the timestamp-based scheduler when processing the schedule S, assuming that, initially, rts(α)=wts(α)=0, and wts-c(α)=cb(α)=`true` for each element α of the database, and assuming that the subscript of each action denotes the timestamp of the transaction executing such action.

5.4 Tell whether S is ACR (Avoding Cascading Rollback) or not, explaining the answer in detail.

Problem 4
Relation `R(A,B,C)` has 1.400.000 tuples stored in a heap file, relation `Q(E,F,G,H,L)` has 2.400.000 tuples stored in a heap file, and there is a B^+-tree index on (`E,F`), where (`E,F`) is the key of `Q`. We know that each attribute or pointer occupies 20 Bytes, the size of each page is 600 Bytes, and the buffer has 400 free frames. Consider the following SQL query (we remind the student that the `union` clause in SQL computes the union of two tables by eliminating duplicates):

```
select A,B from R order by A,B
union
select E,F from Q order by E,F
```

Describe in detail the algorithm you would use to compute the answer to the above query, and tell which is the cost of the algorithm in terms of number of page accesses.

Problem 5
Assume that relation `Q(E,F)` has 10.000 tuples, relation `R(A,B,C,D)` has 400.000 tuples, attribute `B` has 2.000 values uniformly distributed on the tuples of `R`, each page of our system contains 400 Bytes, every attribute value or pointer requires 20 Bytes, and we have 252 free buffer frames. Consider the query

```
select E
from Q
where F not in (select B from R)
```

and tell which is the algorithm you would use and the corresponding cost (in terms of number of page accesses) for executing such query for each of the following methods for representing `R`: (1) heap file; (2) sorted file with sorting key `B`; (3) heap file with unclustering, dense sorted index with duplicates with search key `B` (strongly dense index); (4) sorted file with clustering, dense sorted index without duplicates (i.e., strongly dense) with search key `B`; (5) sorted file with clustering, sparse sorted index with search key `B`.

Problem 5

Assume that relation `Q(E,F)` has 10.000 tuples, relation `R(A,B,C,D)` has 400.000 tuples, attribute `B` has 2.000 values uniformly distributed on the tuples of `R`, each page of our system contains 400 Bytes, every attribute value or pointer requires 20 Bytes, and we have 252 free buffer frames. Consider the query

```
select E
from Q
where F not in (select B from R)
```

and tell which is the algorithm you would use and the corresponding cost (in terms of number of page accesses) for executing such query for each of the following methods for representing `R`: (1) heap file; (2) sorted file with sorting key `B`; (3) heap file with unclustering, dense sorted index with duplicates with search key `B` (strongly dense index); (4) sorted file with clustering, dense sorted index without duplicates (i.e., strongly dense) with search key `B`; (5) sorted file with clustering, sparse sorted index with search key `B`.

1) 400 / 20 = 20 VALUE IN A PAGE

Q: 20/2 = 10 TUPLES IN A PAGE → 10000 / 10 = 1000 PAGES

R: 20/4 = 5 TUPLES IN A PAGE → 400000 / 5 = 80 000 PAGES

PASS 1:

Q: 1000 / 252 = 4 RUNS

R: 80000 / 252 = 318 RUNS

PASS 2:

R: 318 / 247 = 2 RUNS

PASS 3:

WE COMPARE ALL VALUES OF F WITH THE VALUES OF B

COST = 3 B(Q) + 5 B(R) = 403 000 PAGE ACCESSES

NESTED LOOP.

WE LOAD Q 4 TIMES IN 260 FRAMES, 1 FRAME OF OUTPUT AND 1 FOR R.

COST = B(Q) + 4 B(R) = 321 000 PAGE ACCESSES

2) WE PERFORM A BINARY SEARCH

COST = $B(Q) + 10000 \cdot \log_2 80000 = 171000$ PAGE ACCESSES

3) WE HAVE A DATA ENTRY FOR EACH TUPLE

400 000 DATA ENTRIES

IN EACH PAGE THERE ARE 20 VALUES

IN EACH INDEX PAGE WE HAVE

$$400\,000 \cdot 2\ (<KEY, RID>) = 800\,000 \text{ VALUES}$$

EACH INDEX PAGE HAS 20 VALUES, SO:

$$800\,000 / 20 = 40\,000 \text{ INDEX PAGES}$$

WE PERFORM A BINARY SEARCH $\rightarrow \log_2 40000 = 16$

$$\text{COST} = B(Q) + 10000 \cdot \log_2 40000 = 161\,000 \text{ PAGE ACCESSES}$$

4) 2000 DATA ENTRIES

IN EACH PAGE THERE ARE 20 VALUES

IN EACH INDEX PAGE WE HAVE

$$2000 \cdot 2\ (<KEY, RID>) = 4000 \text{ VALUES}$$

EACH INDEX PAGE HAS 20 VALUES, SO:

$$4000 / 20 = 200 \text{ INDEX PAGES}$$

WE PERFORM A BINARY SEARCH $\rightarrow \log_2 200 = 8$

$$\text{COST} = B(Q) + 10000 \cdot \log_2 200 = 81\,000 \text{ PAGE ACCESSES}$$

5) WE HAVE A DATA ENTRY FOR EACH PAGE 80 000 DATA ENTRIES

IN EACH INDEX PAGE WE HAVE

$$80000 \cdot 2\ (<KEY, RID>) = 160000 \text{ VALUES}$$

EACH INDEX PAGE HAS 20 VALUES, SO:

$$160000 / 20 = 8000 \text{ INDEX PAGES}$$

WE PERFORM A BINARY SEARCH $\rightarrow \log_2 8000 = 13$

$$\text{COST} = B(Q) + 10000 \cdot (\log_2(8000) + 1) = 141000 \text{ PAGE ACCESSES}$$