Data Management – AA 2016/17 (06/02/2017) – exam $\boxed{\text{A}}$

Problem 1
Consider the following schedule

$$S = w_1(u)\, r_1(x)\, w_3(x)\, r_2(y)\, r_1(y)\, c_1\, r_4(u)\, w_2(y)\, w_3(y)\, c_3\, w_4(z)\ c_4\, r_2(z)\, c_2.$$

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 2
Assume that relation `R(A,B)` has 10.000 tuples, relation `Q(C,D,E,F)` has 400.000 tuples, attribute `D` has 2.000 values uniformly distributed on the tuples of `Q`, 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 A
from R
where not exists (select * from Q where Q.D = R.B)
```

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 `Q`: (1) heap file; (2) sorted file with sorting key `D`; (3) heap file with unclustering, dense sorted index with duplicates with search key `D` (strongly dense index); (4) sorted file with clustering, dense sorted index without duplicates with search key `D`; (5) sorted file with clustering, sparse sorted index with search key `D`.

Problem 3
The SQL table `R(A,B,C)` has 1.400.000 tuples stored in a heap file, the SQL table `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 `minus` clause in SQL computes the difference of two tables by eliminating duplicates):

```
select A,B from R order by A,B
minus
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 4
A schedule S is said to be *parsimonious* 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 parsimonious schedule is a 2PL schedule (with shared and exclusive locks).
2. Every parsimonious schedule is conflict serializable.
3. Every parsimonious schedule is view serializable.

Problem 5
Suppose that we have only 3 buffer frames available, and we have to compute the union (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 1

Consider the following schedule

$$S = w_1(u)\, r_1(x)\, w_3(x)\, r_2(y)\, r_1(y)\, c_1\, r_4(u)\, w_2(y)\, w_3(y)\, c_3\, w_4(z)\ c_4\, r_2(z)\, c_2.$$

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.

1) $S = w_1(u)\, r_1(x)\, w_3(x)\, r_2(y)\, r_1(y)\, c_1\, r_4(u)\, w_2(y)\, w_3(y)\, c_3\, w_4(z)\ c_4\, r_2(z)\, c_2.$

S'. $x\ell_1(u)\ w_1(u)\ s\ell_1(x)\ r_1(x)\ s\ell_1(y)\ u_1(x)\ x\ell_3(x)\ w_3(x)\ s\ell_2(y)\ r_2(y)$
$r_1(y)\ u_1(u)\ u_1(y)\ c_1\ s\ell_4(u)\ r_4(u)\ x\ell_2(y)\ w_2(y)\ s\ell_2(z)\ u_2(y)$
$x\ell_3(y)\ w_3(y)\ u_3(y)\ u_3(x)\ c_3 \ldots$

WE CANNOT DO $w_4(z)$ BECAUSE z IS LOCK WITH $r_2(z)$, BUT $r_2(z)$ COMES AFTER $w_4(z)$

2) $S = w_1(u)\, r_1(x)\, w_3(x)\, r_2(y)\, r_1(y)\, c_1\, r_4(u)\, w_2(y)\, w_3(y)\, c_3\, w_4(z)\ c_4\, r_2(z)\, c_2.$

$1 \rightarrow 4 \rightarrow 2$

READ-FROM: $\langle r_4(u), w_1(u) \rangle$, $\langle r_2(z), w_4(z) \rangle$

FINAL-WRITE: $w_4(z)$, $w_3(y)$, $w_3(x)$, $w_1(u)$

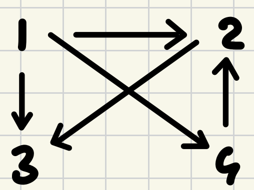
WE HAVE TO SEARCH A S' SERIAL SCHEDULE THAT HAS THE SAME READ FROM, FINAL WRITE OF S

S': $T_1\ T_4\ T_2\ T_3$ HAS THE SAME READ-FROM, FINAL-WRITE

SO S IS VIEW

IT CAN BE PROVED ALSO THROUGH THE $P(S)$

$S = w_1(u)\, r_1(x)\, w_3(x)\, r_2(y)\, r_1(y)\, c_1\, r_4(u)\, w_2(y)\, w_3(y)\, c_3\, w_4(z)\ c_4\, r_2(z)\, c_2.$



CONFLICT $\rightarrow$ VIEW

4) $S = w_1(u)\, r_1(x)\, w_3(x)\, r_2(y)\, r_1(y)\, c_1\, r_4(u)\, w_2(y)\, w_3(y)\, c_3\, w_4(z)\, c_4\, r_2(z)\, c_2.$

READ-FROM: $<r_4(u), w_1(u)>$, $<r_2(z), w_4(z)>$

c_1 BEFORE r_4 , c_4 BEFORE r_2 $\quad$ S IS ACR

3) $S = w_1(u)\, r_1(x)\, w_3(x)\, r_2(y)\, r_1(y)\, c_1\, r_4(u)\, w_2(y)\, w_3(y)\, c_3\, w_4(z)\, c_4\, r_2(z)\, c_2.$

$w_1(u)$	OK	$wTS(u) = 1$, $cb(u) =$ FALSE
$r_1(x)$	OK	$rTS(x) = 1$
$w_3(x)$	OK	$wTS(x) = 3$, $cb(x) =$ FALSE
$r_2(y)$	OK	$rTS(y) = 2$
$r_1(y)$	OK	
c_1	OK	$wTS\text{-}c(u) = 1$, $cb(u) =$ TRUE
$r_4(u)$	OK	$rTS(u) = 4$
$w_2(y)$	OK	$wTS(y) = 2$, $cb(y) =$ FALSE
$w_3(y)$	OK	WAITING
$w_4(z)$	OK	$wTS(z) = 4$, $cb(z) =$ FALSE
c_4	OK	$wTS\text{-}c(z) = 4$, $cb(z) =$ TRUE
$r_2(z)$	READ TOO LATE	ROLLBACK
$w_3(y)$	OK	$wTS(y) = 3$
c_3		$wTS\text{-}c(y) = 3$, $cb(z) =$ TRUE

Problem 2

Assume that relation `R(A,B)` has 10.000 tuples, relation `Q(C,D,E,F)` has 400.000 tuples, attribute `D` has 2.000 values uniformly distributed on the tuples of `Q`, 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 A
from R
where not exists (select * from Q where Q.D = R.B)
```

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 `Q`: (1) heap file; (2) sorted file with sorting key `D`; (3) heap file with unclustering, dense sorted index with duplicates with search key `D` (strongly dense index); (4) sorted file with clustering, dense sorted index without duplicates with search key `D`; (5) sorted file with clustering, sparse sorted index with search key `D`.

R: 10 000 · 2 = 20 000 VALUES
20 000 · 20 BYTES = 400 000 BYTES
400 000 / 400 = 1000 PAGES

Q: 400 000 · 4 = 1 600 000 VALUES
1600 000 · 20 = 32 000 000 BYTES
32 000 000 /400 = 80 000 PAGES

1) ~~PASS 1:~~

~~R: 1000 /252 = 4 RUNS~~
~~Q: 80000 / 252 = 318 RUNS~~

~~PASS 2.~~

~~Q: 318 / 252 = 2 RUNS~~

~~PASS 3:~~

~~WE READ ALL RUNS AND EXECUTE THE QUERY~~

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

BLOCK NESTED LOOP

WE LOAD R IN 4 TIMES, 1 FRAME FOR Q, 1 FOR OUTPUT

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

2) WE READ ONCE B(R), AND PERFORM A BINARY SEARCH ON Q

COST = $B(R) + 10000 \cdot \log_2 B(Q) = 171\,000$

MAYBE WITH TWO PASS COST = 3 B(R) + B(Q) = 83000

3) WE HAVE A DATA ENTRY FOR EACH TUPLE

THERE ARE $400 / 20 = 20$ VALUES IN A PAGE

WE HAVE 400000 DATA ENTRY

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

$800\,000 / 20 = 40\,000$ INDEX PAGES $\rightarrow \log_2 40000 = 16$

$$COST = B(R) + 10\,000 \cdot \log_2 40000 = 161\,000$$

4) WE HAVE 2000 DATA ENTRIES

THERE ARE ALWAYS 20 VALUES IN A PAGE

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

$4000 / 20 = 200$ INDEX PAGES

$$COST = B(R) + 10\,000 \cdot \log_2 200 = 81\,000$$

5) WE HAVE A DATA ENTRY FOR EACH PAGE

20 VALUES IN A PAGE   80000 ENTRIES

$80\,000 \cdot 2$ (<KEY, RID>) $= 160000$ VALUES

$160\,000 / 20 = 8000$ INDEX PAGES

$$COST = B(R) + 10\,000 \cdot (\log_2(8000) + 1) = 41\,000$$

Problem 3

The SQL table `R(A,B,C)` has 1.400.000 tuples stored in a heap file, the SQL table `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 `minus` clause in SQL computes the difference of two tables by eliminating duplicates):

```
select A,B from R order by A,B
minus
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.

R. 1 400 000 · 3 = 4 200 000 VALUES
4 200 000 · 20 = 84 000 000 BYTES
84 000 000 / 600 = 140 000 PAGES

Q: 2 400 000 5 = 12 000 000 VALUES
12 000 000 · 20 = 240 000 000
240 000 000 / 600 = 400 000 PAGES

THE INDEX IS DENSE, A DATA ENTRY FOR EACH TUPLE

600 / 20 = 30 VALUES IN A PAGE

30 / 3 (<KEY, RID>) = 10 $\xrightarrow{67\%}$ 6 ENTRIES FOR EACH LEAF

F = 0.75 · 10 = 7

N = 2400 000 / 6 = 400 000

WE PERFORM A PROJECTION ON A AND B

1400 000 / (600 / 40) = 93334

140000 / 399 (1 FRAME FOR PROJ) = 359 SUBLIST FOR 93334 PAGES

COST = B(R) + 2 · 93334 + N = 726668

Problem 4

A schedule S is said to be *parsimonious* 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 parsimonious schedule is a 2PL schedule (with shared and exclusive locks).
2. Every parsimonious schedule is conflict serializable.
3. Every parsimonious schedule is view serializable.

1) $T_1: r_1(x)\ r_1(y)$
$T_2: w_2(y)$
$T_3: w_3(x)$

$S: r_1(y)\ w_2(y)\ w_3(x)\ r_1(x)$ ✗

$sl_1(y)\ r_1(y)\ xl_1(x)\ u_1(y)\ xl_2(y)\ w_2(y)$

2) SINCE NO ELEMENT IS WRITTEN / READ TWICE, IT MEANS THAT FOR AN ELEMENT x THERE IS ONLY A CONFLICT ONE-WAY.

A CONFLICT CAN BE HAPPENED ONLY WITH A $W_i(x)$, AND A T_i PARSIMONIOUS CAN HAVE ONLY A WRITE OR ONLY READ ACTIONS.

THE PRECEDENCE GRAPH IS ALWAYS ACYCLIC → CONFLICT-SER

3) SINCE EVERY PARSIMONIOUS SCHEDULE IS CONFLICT-SER, THEN IT'S ALSO VIEW-SER.

Problem 5

Suppose that we have only 3 buffer frames available, and we have to compute the union (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.

3 BUFFER FRAMES SET UNION

WE SORT R AND S TILL THEY FIT IN ONE FRAME AND WE DELETE DUPLICATES

$$2 \cdot B(R) \cdot (\log_2 B(R) + 1)$$

$$2 \cdot B(S) \cdot (\log_2 B(S) + 1)$$

WE READ B(R) AND B(S) TO PERFORM THE UNION

$$COST = (2 \cdot B(R) \cdot (\log_2 B(R) + 1)) + (2 \cdot B(S) \cdot (\log_2 B(S) + 1)) + B(R) + B(S)$$