Data Management – exam of 08/06/2023 **(B)**

Problem 1

Let `Trainer(id,nation)` be a relation with 500 pages stored in a file sorted on ⟨`id,nation`⟩, `Team(name,city)` a relation with 100 pages sorted on ⟨`name,city`⟩ and `Coached(id,nation,name,city)` a relation with 35.000 pages sorted on ⟨`id,nation,name,city`⟩. With the goal of knowing, for each trainer, the teams that (s)he has not coached, we want to compute the set difference between the cartesian product of `Trainer` and `Team` and the relation `Coached`. Tell which is the best algorithm that a query engine with 107 free buffer frames should use for this task, indicating also the cost of executing such algorithm.

Problem 2

Answer the following two questions, providing a suitable motivation for each answer.

3.1 Give an example of three transactions which have the following properties: (*i*) there exists a schedule S on the three transactions such that when S is given in input to a timestamp-based scheduler, a deadlock occurs; (*ii*) for each pair of the three transactions and for any schedule S on such pair, no deadlock occurs when S is given in input to a timestamp-based scheduler.

3.2 Try to generalize the above example to the case of n transactions. More precisely, try to come up with an example of n transactions having the following properties: (*i*) There exists a schedule S on the n transactions such that when S is given in input to a timestamp-based scheduler, a deadlock occurs. (*ii*) For each $n-1$ transactions of the n chosen transactions and for any schedule S on such $n-1$ transactions, no deadlock occurs when S is given in input to a timestamp-based scheduler.

Problem 3

Let `R(A,B,C,D)` (with 52.000 pages) and `S(E,F,G,H)` (with 20.000 pages) be two relations stored in a heap at processor P_0. We know that 300 free buffer frames are available at P_0 and that 100 tuples of each relation fit in one page. Also, we know that attribute `C` contains 500 values uniformly distributed in the tuples of `R`, and the same holds for attribute `H` in `S`. Finally, we know that there are 10 processors $P_1, \ldots, P_{10}$ that we can use for processing queries, each with 80 free frames available. Consider the query that computes the equi-join between `R` and `S` on the condition `C = H`, and compare the following two cases:

1.1 The query is executed at processor P_0.

1.2 The query is executed in parallel after sending the tuples of the two relations to processors $P_1, \ldots, P_{10}$.

For each of the above cases, illustrate the algorithm you would use and discuss its cost in terms of number of page accesses (case 1.1) and elapsed time (case 1.2).

Problem 4

Consider the following schedule S:

$$B_1\ w_1(X)\ B_2\ w_2(Y)\ B_3\ w_3(Z)\ c_3\ B_4\ r_4(Z)\ r_4(Y)\ w_4(W)\ B_5\ r_5(W)\ c_4\ w_5(W)\ c_5\ w_1(Y)\ c_1\ w_2(X)\ c_2$$

where the action B_i means "begin transaction T_i", the initial value of every item X, Y, Z, W is 200 and every write action increases the value of the element on which it operates by 200. Suppose that S is executed by PostgreSQL, and describe what happens when the scheduler analyzes each action (illustrating also which are the values read and written by all the "read" and "write" actions) in both the following two cases: (1) all the transactions are defined with the isolation level "read committed"; (2) all the transactions are defined with the isolation level "repeatable read".

Problem 5 (only for students enrolled in an A.Y. before 2021/22 who do **not** do the project)

Consider a database B about musicians and concerts, where: (*i*) for each musician we are interested in the id, the age, the sex, the city of birth, the city (s)he is living in, the music academy where (s)he graduated and the concerts to which (s)he participated; (*ii*) for each concert we are interested in the average price of the ticket, in the month and the year when it was held, with the decade of the year, and the city where the concert was held; (*iii*) for each city we are interested in the province, the region and the number of inhabitants; (*iv*) for each music academy we are interested in the name and the founder. We want to build a data warehouse on all the above data for various analyses on participation of musicians to concerts in the last 50 years. You are asked to show (1) the ER schema of the database, (2) the DFM schema of the data warehouse, (3) the corresponding star schema (optionally, with the queries used to populate the star schema tables), and (4) based on such tables, the SQL query that computes the number of musicians who participated in concerts, for each region of the city of birth of the musicians, and for each music academy where the musicians graduated.

Problem 1

Let `Trainer(id,nation)` be a relation with 500 pages stored in a file sorted on ⟨`id,nation`⟩, `Team(name,city)` a relation with 100 pages sorted on ⟨`name,city`⟩ and `Coached(id,nation,name,city)` a relation with 35.000 pages sorted on ⟨`id,nation,name,city`⟩. With the goal of knowing, for each trainer, the teams that (s)he has not coached, we want to compute the set difference between the cartesian product of `Trainer` and `Team` and the relation `Coached`. Tell which is the best algorithm that a query engine with 107 free buffer frames should use for this task, indicating also the cost of executing such algorithm.

WE PERFORM THE CARTESIAN PRODUCT IN THE BUFFER WITHOUT SAVING IT, AND FOR EACH TUPLE WE COMPARE IT WITH THE TUPLE IN COACHED TO DELETE DUPLICATION.

WE PUT TEAM IN 100 FRAMES, 1 FRAME FOR TRAINER, 1 FRAME FOR TUPLES OF CARTESIAN PROD, 1 FRAME FOR COACHED, 1 FRAME FOR OUTPUT

COST = B(TEAM) + B(TRAINER) + B(COACHED) = 35 600 PAGE ACCESSES

Problem 2

Answer the following two questions, providing a suitable motivation for each answer.

3.1 Give an example of three transactions which have the following properties: (i) there exists a schedule S on the three transactions such that when S is given in input to a timestamp-based scheduler, a deadlock occurs; (ii) for each pair of the three transactions and for any schedule S on such pair, no deadlock occurs when S is given in input to a timestamp-based scheduler.

3.2 Try to generalize the above example to the case of n transactions. More precisely, try to come up with an example of n transactions having the following properties: (i) There exists a schedule S on the n transactions such that when S is given in input to a timestamp-based scheduler, a deadlock occurs. (ii) For each $n-1$ transactions of the n chosen transactions and for any schedule S on such $n-1$ transactions, no deadlock occurs when S is given in input to a timestamp-based scheduler.

1) T_1 : $w_1(A)$ $w_1(B)$

T_2 : $w_2(B)$ $w_2(C)$

T_3 : $w_3(C)$ $w_3(A)$

S_{123}: $w_1(A)$ $w_2(B)$ $w_1(B)$ $w_3(C)$ $w_2(C)$ $w_3(A)$ DEADLOCK

S_{12}: $w_1(A)$ $w_2(B)$ $w_1(B)$ $w_2(C)$

S_{23}: $w_2(B)$ $w_3(C)$ $w_2(C)$ $w_3(A)$

S_{13}: $w_1(A)$ $w_3(C)$ $w_1(B)$ $w_3(A)$

2) S: $w_1(x_1)$ $w_2(x_2)$ $w_1(x_2)$ ... $w_n(x_n)$ $w_{n+1}(x_{n+1})$ $w_n(x_{n+1})$ ... $w_{n+1}(x_1)$

Problem 3

Let R(A,B,C,D) (with 52.000 pages) and S(E,F,G,H) (with 20.000 pages) be two relations stored in a heap at processor P_0. We know that 300 free buffer frames are available at P_0 and that 100 tuples of each relation fit in one page. Also, we know that attribute C contains 500 values uniformly distributed in the tuples of R, and the same holds for attribute H in S. Finally, we know that there are 10 processors $P_1, \ldots, P_{10}$ that we can use for processing queries, each with 80 free frames available. Consider the query that computes the equi-join between R and S on the condition C = H, and compare the following two cases:

1.1 The query is executed at processor P_0.

1.2 The query is executed in parallel after sending the tuples of the two relations to processors $P_1, \ldots, P_{10}$.

For each of the above cases, illustrate the algorithm you would use and discuss its cost in terms of number of page accesses (case 1.1) and elapsed time (case 1.2).

1) PASS 1:

WE SORT R AND S

$$\left.\begin{array}{l} 52000/300 = 174 \text{ RUNS FOR R} \\ 20000/300 = 67 \text{ RUNS FOR S} \end{array}\right\} 241 \text{ RUNS}$$

PASS 2:

WE READ THE RUNS OF R AND S, AND WE PERFORM EQUIJOIN ON C=H.

COST = 3 (B(R) + B(S)) = 216 000 PAGE ACCESSES

2) WE USE A GOOD HASH FUNCTION TO PARTITIONATE R AND S BASED ON C AND H VALUES, IN 10 PROCESSORS

5200 PAGES OF R IN EACH P_i $\rightarrow$ 50 DIFFERENT VALUES OF C

2000 PAGES OF S IN EACH P_i $\rightarrow$ 50 DIFFERENT VALUES OF H

5200 · 100 = 520000 TUPLES $\rightarrow$ 520000 / 50 = 10 400 TUPLES OF R WITH SAME VALUES OF C

2000 · 100 = 200000 TUPLES $\rightarrow$ 200000 / 50 = 4000 TUPLES OF S WITH SAME VALUES OF H

COST =

Problem 4

Consider the following schedule S:

$$B_1\ w_1(X)\ B_2\ w_2(Y)\ B_3\ w_3(Z)\ c_3\ B_4\ r_4(Z)\ r_4(Y)\ w_4(W)\ B_5\ r_5(W)\ c_4\ w_5(W)\ c_5\ w_1(Y)\ c_1\ w_2(X)\ c_2$$

where the action B_i means "begin transaction T_i", the initial value of every item X, Y, Z, W is 200 and every write action increases the value of the element on which it operates by 200. Suppose that S is executed by PostgreSQL, and describe what happens when the scheduler analyzes each action (illustrating also which are the values read and written by all the "read" and "write" actions) in both the following two cases: (1) all the transactions are defined with the isolation level "read committed"; (2) all the transactions are defined with the isolation level "repeatable read".

T_1: BEGIN T_1

$w_1(x) = 400$

T_2: BEGIN T_2

$w_2(y) = 400$

CRASH ALL THINGS

T_3. BEGIN T_3

$w_3(z) = 400$

COMMIT

T_4: BEGIN T_4

$r_4(z) = 400$
$r_4(y) = 200$
$w_4(w) = 400$

T_5: BEGIN T_5

$r_5(w) = 200$

T_4: COMMIT

T_5: $w_5(w) = 600$ $\leftarrow$ LOST UPDATE
COMMIT

T_1: $w_1(y) = 400$
COMMIT

DEADLOCK

T_2: CRASH

T_1 : BEGIN T_1

$w_1(x) = 400$

T_2: BEGIN T_2

$w_2(y) = 400$

T_3. BEGIN T_3

$w_3(z) = 400$

COMMIT

T_4: BEGIN T_4

$r_4(z) = 400$
$r_4(y) = 200$
$w_4(w) = 400$

T_5: BEGIN T_5

$r_5(w) = 200$

T_4: COMMIT

T_5: ~~$w_5(w) = 400$~~
COMMIT $\rightarrow$ ROLL BACK

T_1: $w_1(y) = 400$
COMMIT

T_2: CRASH