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

Problem 1

Let `R(A,B,C)` (with 26.000 pages) and `Q(D,E,F)` (with 10.000 pages) be two relations stored in a heap at processor P_0. We know that 200 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` of `R` contains 500 values uniformly distributed in the tuples of the relation, and the same holds for attribute `D` of `Q`. Finally, we know that there are 10 processors $P_1, \dots, P_{10}$ that we can use for processing queries, each with 80 free frames available. Consider the query that computes the equi-join between `R` and `Q` on the condition `C = D`, 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 $P_1, \dots, P_{10}$.

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 2

Consider the following schedule S:

$$B_1\ w_1(A)\ B_2\ w_2(D)\ B_3\ w_3(E)\ c_3\ B_4\ r_4(E)\ r_4(D)\ w_4(C)\ B_5\ r_5(C)\ c_4\ w_5(C)\ w_1(D)\ c_1\ w_2(A)\ c_2\ c_5$$

where the action B_i means "begin transaction T_i", the initial value of every item A, C, D, E is 100 and every write action increases the value of the element on which it operates by 100. 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 3

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

3.1 Give an example of three transactions, which obey 2PL and have the following properties: (*i*) there exists a schedule S on the three transactions such that when S is given in input to a 2PL 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 2PL 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, which obey 2PL and have the following properties: (*i*) There exists a schedule S on the n transactions such that when S is given in input to a 2PL 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 2PL scheduler.

Problem 4

Let `Person(id,age)` be a relation with 500 pages stored in a file sorted on ⟨`id,age`⟩, `City(name,region)` a relation with 200 pages sorted on ⟨`name,region`⟩ and `Visited(id,age,name,region)` a relation with 20.000 pages sorted on ⟨`id,age,name,region`⟩. With the goal of knowing, for each person, the cities that (s)he has not visited, we want to compute the set difference between the cartesian product of `Person` and `City` and the relation `Visited`. Tell which is the best algorithm that a query engine with 206 free buffer frames should use for this task, indicating also the cost of executing such algorithm.

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

Consider a database B about high schools, where: (*i*) for each student we are interested in the id, the age, the sex, the city of birth, the family (s)he is living with, and the school where (s)he has been enrolled in the various years, with the decades of the years; (*ii*) for each school we are interested in the type (classical, scientific, technical, etc.) and the city where the school is located; (*iii*) for each city we are interested in the province, the region and the number of inhabitants; (*iv*) for each family we are interested in the level of the income and the number of members. We want to build a data warehouse on all the above data for various analyses on student enrollments in the last 30 years. You are asked to show (1) the ER schema of the database B, (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 students, for each province of the school where they are enrolled, for each region of the city of residence of the students, and for each level of the income of the family of the students.

Problem 1

Let R(A,B,C) (with 26.000 pages) and Q(D,E,F) (with 10.000 pages) be two relations stored in a heap at processor P_0. We know that 200 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 of R contains 500 values uniformly distributed in the tuples of the relation, and the same holds for attribute D of Q. Finally, we know that there are 10 processors $P_1, \dots, P_{10}$ that we can use for processing queries, each with 80 free frames available. Consider the query that computes the equi-join between R and Q on the condition C = D, 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 $P_1, \dots, P_{10}$.

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:

$$\left.\begin{array}{l} R: 26000/200 = 130 \text{ RUNS} \\ Q: 10000/200 = 50 \text{ RUNS} \end{array}\right\} 180 \leq 199$$

PASS 2:

WE MERGE ALL RUNS OF R AND Q TO USE 2 FRAMES FOR R AND Q.

BECAUSE $\frac{26000}{500} = 52$ PAGES WITH THE SAME VALUE OF C

$\frac{10000}{500} = 20$ PAGES WITH THE SAME VALUE OF D

SO WE NEED 20 FRAMES FOR THIS JOIN

PASS 3:

WE READ ALL RUNS AND WE COMPARE THE VALUES OF C AND D, TO WRITE IN THE OUTPUT

COST = $5(B(R) + B(Q))$ = 180 000 PAGE ACCESSES

(1 FRAME R, 1 FRAME Q, 20 FRAMES JOIN, 1 FRAME OUTPUT)

2) $\frac{B(R)}{M-1} + \frac{B(Q)}{M-1} \leq M$

$$\text{EACH } P_i \text{ HAS } \left.\begin{array}{l} 26000/10 = 2600 \text{ PAGE OF R} \\ 10000/10 = 1000 \text{ PAGE OF Q} \end{array}\right\} 3600 \text{ PAGES}$$

WE PARTITIONATE ALL PAGES WITH AN HASH BASED ALGORITHM

$\frac{2600}{79} + \frac{1000}{79} = 46 \leq 79$

20 PAGES FOR THE JOIN ENTER HERE, SO WE USE THE MERGE-SORT JOIN

COST = $\overset{P_0}{\overline{B(R) + B(Q)}} + 2(P(R) + P(Q)$ = 43200 PAGE ACCESSES

ELAPSED TIME

Problem 2

Consider the following schedule S:

$$B_1\ w_1(A)\ B_2\ w_2(D)\ B_3\ w_3(E)\ c_3\ B_4\ r_4(E)\ r_4(D)\ w_4(C)\ B_5\ r_5(C)\ c_4\ w_5(C)\ w_1(D)\ c_1\ w_2(A)\ c_2\ c_5$$

where the action B_i means "begin transaction T_i", the initial value of every item A, C, D, E is 100 and every write action increases the value of the element on which it operates by 100. 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".

1)

T_1: BEGIN T_1
$w_1(A) = 200$

T_2: BEGIN T_2
$w_2(D) = 200$

T_3: BEGIN T_3
$w_3(E) = 200$
COMMIT

T_4: BEGIN T_4
$r_4(E) = 200$
$r_4(D) = 100$
$w_4(C) = 200$

T_5: BEGIN T_5
$r_5(C) = 100$

T_4. COMMIT

T_5: $w_5(C) = 300$ LOST UPDATE

T_1: $w_1(D) = 200$
COMMIT

DEAD LOCK

T_2 DO ROLLBACK

T_2: $w_2(A) = 300$
COMMIT

T_5: COMMIT

1)

T_1: BEGIN T_1
$w_1(A) = 200$

T_2: BEGIN T_2
$w_2(D) = 200$

T_3: BEGIN T_3
$w_3(E) = 200$
COMMIT

T_4: BEGIN T_4
$r_4(E) = 200$
$r_4(D) = 100$
$w_4(C) = 200$

T_5: BEGIN T_5
$r_5(C) = 100$

T_4. COMMIT

T_5: $w_5(C) = 200$ CRASH

T_1: $w_1(D) = 200$
COMMIT

DEAD LOCK

T_2 DO ROLLBACK

T_2: $w_2(A) = 200$
COMMIT

IN THE DEADLOCK T_1 IS WAITING, AND T_2 DO ROLLBACK

Problem 3

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

3.1 Give an example of three transactions, which obey 2PL and have the following properties: (*i*) there exists a schedule S on the three transactions such that when S is given in input to a 2PL 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 2PL 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, which obey 2PL and have the following properties: (*i*) There exists a schedule S on the n transactions such that when S is given in input to a 2PL 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 2PL scheduler.

1) T_1: $w_1(x)\ w_1(y)$

T_2: $w_2(y)\ w_2(z)$

T_3: $w_3(z)\ w_3(x)$

S_{123}: $w_1(x)\ w_2(y)\ w_3(z)\ w_1(y)\ w_2(z)\ w_3(x)$

T_1 WAITS FOR T_2, T_2 WAITS FOR T_3, T_3 WAITS FOR T_1

S_{12}: $xl_1(x)\ w_1(x)\ xl_2(y)\ w_2(y)\ xl_2(z)\ u_2(y)\ xl_1(y)\ w_1(y)\ u_1(y)\ u_1(x)\ \overset{\checkmark}{w_2(z)}\ u_1(z)$

S_{23}: SAME AS ABOVE

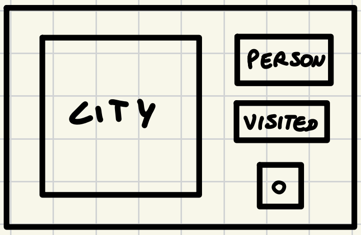
S_{13}: SAME AS ABOVE

2) S: $w_1(x_1)\ w_2(x_2) \ldots w_{n-1}(x_{n-1})\ w_n(x_n)\ w_1(x_2)\ w_2(x_3) \ldots w_{n-1}(x_n)\ w_n(x_1)$

Problem 4

Let `Person(id,age)` be a relation with 500 pages stored in a file sorted on ⟨`id,age`⟩, `City(name,region)` a relation with 200 pages sorted on ⟨`name,region`⟩ and `Visited(id,age,name,region)` a relation with 20.000 pages sorted on ⟨`id,age,name,region`⟩. With the goal of knowing, for each person, the cities that (s)he has not visited, we want to compute the set difference between the cartesian product of `Person` and `City` and the relation `Visited`. Tell which is the best algorithm that a query engine with 206 free buffer frames should use for this task, indicating also the cost of executing such algorithm.

WE USE 200 FRAMES FOR CITY, 1 FOR PERSON, 1 FOR THE CARTESIAN PRODUCT, 1 FOR OUTPUT



COST = B(CITY) + B(CITY) + B(VISITED) = 20700