Advanced Database Systems, second intermediate test – 4 June 2019 – V1.0

Please feel free to answer this test in English, Italian, or any mixture

1. Consider the following log content. Assume that the DB was identical to the buffer before the beginning of this log, and consider a undo-redo protocol

   (begin,T1) (W,T1,A,0,20) (begin,T2) (W,T2,B,0,30) (begin-ckp,{T1,T2}) (W,T2,B,30,50) (W,T1,C,0,30) (end-ckp) (begin,T3) (commit,T2) (W,T3,B,50,70) (begin,T4) (W,T4,D,0,50) (commit,T4)

   a. Before starting this log, what was the content of A, B, C and D in the PS (Persistent Store)?
   b. Assume there was a crash at the end of the logging period. At crash time, what was the content of A, B, C, D in the buffer? What can be said about the content of A, B, C, D in the PS?
   c. At restart time, which transactions are put in the undo-list? Which in the redo-list?
   d. List the operations that are redone, in the order in which are redone
   e. List the operations that are undone, in the order in which are undone
   f. After restart is finished, what is the content of A, B, C, D in the buffer?
   g. Undo and Redo are executed in the buffer or on the PS?
   h. After restart is finished, what is the content of A, B, C, D in the PS? What is different between this answer and that of question (b)?

2. Assume that a system with no scheduler produces the following history, where we omit the commits:

   $r_1[B]$, $w_3[A]$, $r_2[B]$, $w_2[C]$, $w_3[B]$, $r_1[C]$, $r_2[A]$, $w_3[A]$

   a. Is this history c-serializable?
   b. Exhibit a history that may be produced by a strict 2PL scheduler if presented with the above operations in that order, assuming that each transaction commits immediately after its last operation. If a deadlock takes place, assume that the system just stops.

3. Consider the following tables:

   Sales(<u>Date</u>,<u>FKShop</u>,FKCust,FKProd,UnitPrice,Q,TotPrice)
   Shops(PKShop, ShopName, FKCity)
   Cities(PKCity, CityName, CountryName, Region)

With the following sizes:
Sales: NRec: 10.000.000, Npag: 100.000; Shops: NRec: 1000, Npag: 5; Cities: NRec: 200, Npag: 1

Assume a buffer of 1.000 pages. Consider the following queries:

```
SELECT C.CountryName, C.Region, Sum(S.TotPrice)
FROM Sales S, Shops Sh, Cities C
WHERE S.FKShop=Sh.PKShop AND Sh.FKCity =  C.PKCity AND S.Date >
1/1/2018
GROUP BY C.CountryName, C.Region

SELECT C.CountryName, C.Region, Year(S.Date), Sum(S.TotPrice)
FROM Sales S, Shops Sh, Cities C
WHERE S.FKShop=Sh.PKShop AND Sh.FKCity =  C.PKCity
GROUP BY C.CountryName, C.Region, Year(S.Date)
```

a. In the context of a traditional database, choose a physical organization to optimize both queries, without partitioning or denormalization, and give a synthetic justification of your choice. Specify the primary organization of each table and which indices would you use. Assume that the database contains 1000 different dates each with a similar amount of sales, and that the condition S.Date > 1/1/2018 has a selectivity factor of 10%.
b. Show an access plan for the first query assuming that all tables are stored serially and no index is used, and compute its cost. Assume each field occupies 4 bytes, and pages measure 4000 bytes, if needed.

4. Answer the following questions. Please keep the answers short and WRITE IN YOUR BEST HANDWRITING.

a. What is the difference between a distributed database and a parallel database
b. Assume that we have 5 copies of a piece of data. Give three different examples of Read and Write quorums to obtain a consistent access to that piece of data.

CONTINUES ON THE BACK WITH 5

5. Assume a database Occupations(OccId,FlatId*,CustomerId*,OwnerId*,From,To,Amount), Flats(FlatId,Street,StreetNo,City,Country), Customers(CustomerId,Name,Surname, Street,StreetNo,City,Country), Owners(OwnerId,Name,Surname, Street,StreetNo,City,Country), where the following table describes the sizes of the relations:

	NRec	NPag
Occupations	40,000,000	250,000
Flats	400,000	10,000
Customers	10,000,000	200,000
Owners	200,000	4,000

Each database fact (each occupation) describes the fact that a customer occupied a Flat From a given date To a given date and payed a given Amount. OwnerId* denotes the Owner of the Flat.
A parallel machine with 1000 nodes is used in order to store the database. Occupations, Customers and Owners are distributed on all nodes, with no replication, with a round-robin technique. Flats are hashed on FlatId.
Assume that we want to compute the following query:
SELECT total(o.Amount)
FROM Occupations o, Customers c
WHERE o.CustomerId = c.CustomerId
GROUP BY c.Country, o.From

a. Consider the following strategy:
    a. Redistribute Occupations and Customers over all nodes, with no projection, hashing both over CutomerId
    b. Locally compute the join
    c. Redistribute the result over all nodes by hashing on c.Country, o.From
    d. Locally compute the Group-By

Assume that pages have a size of 4K, you need 10 ms to read/write a page, and 1 ms to send or to receive one page over the network. Assume every machine has a buffer size of 1.000 pages that can be used to perform database operations. Assume a model where you sum cost of reading + cost of sending + cost of receiving + cost of writing, ignoring any parallelism between these four phases.

a. Compute the time that is needed in order to complete the algorithm described. If you miss any information, just ask me.
b. Suggest some possible optimizations to the strategy of point a – just suggest, do not compute anything
c. Suggest a different data distribution strategy in order to optimize this query

Advanced Database Systems, second intermediate test – 4 June 2019 – V1.0 - solutions

Please feel free to answer this test in English, Italian, or any mixture

1. Consider the following log content. Assume that the DB was identical to the buffer before the beginning of this log, and consider a undo-redo protocol

   (begin,T1) (W,T1,A,0,20) (begin,T2) (W,T2,B,0,30) (begin-ckp,{T1,T2}) (W,T2,B,30,50) (W,T1,C,0,30) (end-ckp) (begin,T3) (commit,T2) (W,T3,B,50,70) (begin,T4) (W,T4,D,0,50) (commit,T4)

   a. Before starting this log, what was the content of A, B, C and D in the PS (Persistent Store)?

A=0, B=0, C=0, D=0

   b. Assume there was a crash at the end of the logging period. At crash time, what was the content of A, B, C, D in the buffer? What can be said about the content of A, B, C, D in the PS?

In the buffer: A=20, B=70, C=30, D=50
In the PS: A=20, B=30 o 50 o 70, C=0 o 30, D=0 o 50

   c. At restart time, which transactions are put in the undo-list? Which in the redo-list?

Undo list: T1, T3. Redo list : T2, T4

   d. List the operations that are redone, in the order in which are redone

(W,T2,B,30,50) (W,T4,D,0,50)

   e. List the operations that are undone, in the order in which are undone

(W,T3,B,50,70) (W,T1,C,0,30) (W,T1,A,0,20)

   f. After restart is finished, what is the content of A, B, C, D in the buffer?

A = 0, B=50, C=0, D=50,

   g. Undo and Redo are executed in the buffer or on the PS?

In the buffer

   h. After restart is finished, what is the content of A, B, C, D in the PS? What is different between this answer and that of question (b)?

In the PS: A=0 or 20, B=30 or 50 or 70, C=0 or 30, D=0 or 50.
The only difference is that A may be 0 as well, because of the undo operation.

2. Assume that a system with no scheduler produces the following history, where we omit the commits:

$r_1[B], w_3[A], r_2[B], w_2[C], w_3[B], r_1[C], r_2[A], w_3[A]$

a. Is this history c-serializable?

The conflict graph: T3 -> T2, T1 -> T3, T2-> T3, T2-> T1
We have a cycle, hence the history is not c-serializable

b. Exhibit a history that may be produced by a strict 2PL scheduler if presented with the above operations in that order, assuming that each transaction commits immediately after its last operation

2PC History:

T1	T2	T3
r(B)		
		w(A)
	r(B)	
	w(C)	
		wl(B)*
rl(C)*		
	rl(A)*	

Deadlock

3. Consider the following tables:

Sales(<u>Date</u>,<u>FKShop</u>,FKCust,FKProd,UnitPrice,Q,TotPrice)
Shops(PKShop, ShopName, FKCity)
Cities(PKCity, CityName, CountryName, Region)

With the following sizes:

Sales: NRec: 10.000.000, Npag: 100.000;
Shops: NRec: 1000, Npag: 5; Cities: NRec: 200, Npag: 1

Assume a buffer of 1.000 pages. Consider the following queries:

```
SELECT C.CountryName, C.Region, Sum(S.TotPrice)
FROM Sales S, Shops Sh, Cities C
WHERE S.FKShop=Sh.PKShop AND Sh.FKCity =  C.PKCity AND S.Date > 1/1/2018
GROUP BY C.CountryName, C.Region

SELECT C.CountryName, C.Region, Year(S.Date), Sum(S.TotPrice)
```

```
FROM Sales S, Shops Sh, Cities C
WHERE S.FKShop=Sh.PKShop AND Sh.FKCity =  C.PKCity
GROUP BY C.CountryName, C.Region, Year(S.Date)
```

a. In the context of a traditional database, choose a physical organization to optimize both queries, without partitioning or denormalization, and give a synthetic justification of your choice. Specify the primary organization of each table and which indices would you use. Assume that the database contains 1000 different dates each with a similar amount of sales, and that the condition S.Date > 1/1/2018 has a selectivity factor of 10%.

Shops and Cities fit main memory, hence the organization is not important, and the join can be done at zero cost. Hence, in the first query the dominating cost is that of reading the entire Sales table (the sorting phase is less important since it deals with 10% of the records), while in the second query the dominating cost is that of the sort operation needed by the group by. The best solution would be to sort sales according to Date. In this way, the first query could just read the 10% of Sales that it needs, and then sort it. For the second query, it would read the data sorted by date. For each date, the corresponding sales occupy 100.000/1.000 = 100 pages, hence they fit main memory, and further grouping can be computed in main memory at zero cost. We could avoid even this extra step by using an encoding of PKShop that respects the CountryName order, and keeping Sales sorted on (Date,FKShop), but this is not crucial.

b. Show an access plan for the first query assuming that all tables are stored serially and no index is used, and compute its cost. Assume each field occupies 4 bytes, and pages measure 4000 bytes, if needed.

```
GroupBy({C.CountryName, C.Region},{ Sum(S.TotPrice)})
                        |
Sort(C.CountryName, C.Region), Join, Join, GroupBy, Project  all of them in main memory
                        |
Project(S.Date, S.TotPrice, C.CountryName, C.Region)
                        |
    Join with Cities in main memory
                        |
     Join with Shops in main memory
                        |
     Filter(S.Date > 1/1/2018)
                         |
             TableScan(Sales)
```

C(TableScan)=100.000
ERec(TableScan)=10.000.000
ERec(Filter) = ERec(Join Shop) = ERec(Join Cities) = 1.000.000
EPag(Project) = 1.000.000*16/4000 = 4.000.
C(Sort) = C(TableScan) + NPag(Project)*2 = 100.000+2*4.000 = 108.000

Every other operation takes place in main memory, hence the total cost is 108.000 .

4. Answer the following questions. Please keep the answers short and WRITE IN YOUR BEST HANDWRITING.

   a. What is the difference between a distributed database and a parallel database

There are thre main differences: (1) in a distributed data the communications are more expensive than in a parallel database (2) in the failure models, specifically *partitioning* is a crucial problem in distributed databases, while is usually not considered in the design of a parallel database (3) heterogeneity: parallel databases are usually homogeneous, while distributed system may be extremely heterogeneous, up to the point where different nodes may by under the administration of a different authority of may even not be totally trusted

   b. Assume that we have 5 copies of a piece of data. Give three different examples of Read and Write quorums to obtain a consistent access to that piece of data.

Write all / read one: write quorum 5, read quorum 1. Write all minus 1: wq 4, rq 2. Write half+1: wq 3, rq 3.

5. Assume a database Occupations(OccId,FlatId*,CustomerId*,OwnerId*,From,To,Amount), Flats(FlatId,Street,StreetNo,City,Country), Customers(CustomerId,Name,Surname, Street,StreetNo,City,Country), Owners(OwnerId,Name,Surname, Street,StreetNo,City,Country), where the following table describes the sizes of the relations:

	NRec	NPag
Occupations	40,000,000	250,000
Flats	400,000	10,000
Customers	10,000,000	200,000
Owners	200,000	4,000

Each database fact (each occupation) describes the fact that a customer occupied a Flat From a given date To a given date and payed a given Amount. OwnerId* denotes the Owner of the Flat. A parallel machine with 1000 nodes is used in order to store the database. Occupations, Customers and Owners are distributed on all nodes, with no replication, with a round-robin technique. Flats are hashed on FlatId.
Assume that we want to compute the following query:

```
SELECT total(o.Amount)
FROM Occupations o, Customers c
WHERE o.CustomerId = c.CustomerId
GROUP BY c.Country, o.From
```

a. Consider the following strategy:

a. Redistribute Occupations and Customers over all nodes, with no projection, hashing both over CutomerId
b. Locally compute the join
c. Redistribute the result over all nodes by hashing on c.Country, o.From
d. Locally compute the Group-By

Assume that pages have a size of 4K, you need 10 ms to read/write a page, and 1 ms to send or to receive one page over the network. Assume every machine has a buffer size of 1.000 pages that can be used to perform database operations. Assume a model where you sum cost of reading + cost of sending + cost of receiving + cost of writing, ignoring any parallelism between these four phases. Compute the time that is needed in order to complete the algorithm described. If you miss any information, just ask me.

a. Redistribute Occupations and Customers over all nodes, with no projection, hashing both over CustomerId

Every nodes send 250 pages and 200 pages, and receives the same amount of data.

Cost of reading: (250+200)*10ns
Cost of sending: (250+200)*1ns
Cost of receiving: (250+200)*1ns
Cost of writing: 0, fits main memory

b. Locally compute the join

The join is computer in main memory at zero cost

c. Redistribute the result over all nodes by hashing on c.Country, o.From

b costs 0, and for c:

Cost of sending: (250+200)*1ns
Cost of receiving: (250+200)*1ns

d. Locally compute the Group-By

Again, this costs 0

b. Suggest some possible optimizations to the strategy of point a – just suggest, do not compute anything

At each step, we may project data over the interesting fields only, that would greatly reduce the amount of data to be communicated. A semijoin optimization would not make any sense here, given the absence of conditions and the fast speed of the network.

c. Suggest a different data distribution strategy in order to optimize this query

Since the dominating time here is reading time, this query would be much faster if Occupations and Customers were stored according to a vertical fragmentation that separates the useful fields o.Amount, o.CustomerId, o.From from the others, and the same for c.CustomerId and c.Country. Less importantly, we could co-locate Occupations and Customers so that the join does not need any data transmission.