Advanced Database Systems, second intermediate test – 4 June 2018 – solutions, Version 1.1

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,0,5) (begin,T2) (W,T2,B,0,20) (begin-ckp,{T1,T2}) (W,T1,A,5,10) (end-ckp) (begin,T3) (commit,T1) (W,T3,C,0,20) (commit,T3) (begin,T4) (W,T4,C,20,50) (commit,T4)

   a. Before starting this log, what was the content of A, B and C 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 and C in the buffer? What can be said about the content of A, B and C 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. After restart is finished, what is the content of A, B and C in the buffer?
   f. Undo and Redo are executed in the buffer or on the PS?
   g. After restart is finished, what is the content of A, B and C in the PS? What is different between this answer and that of question (b)?

---

a. Before starting this log, what was the content of A, B and C in the PS (Persistent Store)?
   (A=0,B=0,C=0)
b. Assume there was a crash at the end of the logging period. At crash time, what was the content of A, B and C in the buffer? What can be said about the content of A, B and C in the PS?
   In the buffer: (A=10,B=20,C=50). In the PS: (A=5 or 10, B=20, C=0 or 20 or 50)
c. At restart time, which transactions are put in the undo-list? Which in the redo-list?
   Undo-list=T2, redo-list=T1,T3,T4
d. List the operations that are redone, in the order in which are redone
   (W,T1,A,5,10) (W,T3,C,0,20) (W,T4,C,20,50)
e. After restart is finished, what is the content of A, B and C in the buffer?
   (A=10,B=0,C=50).
f. 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 and C in the PS? What is different between this answer and that of question (b)?
   (A=5 or 10, B=0 or 20, C=0 or 20 or 50)
   The only different is that now B may also be 0, because of the undo operation.

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

   $r_1[A], w_2[C], w_1[B], r_2[B], w_3[C], w_2[A], r_1[B], r_3[A]$

   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

---

a. Yes, is it equivalent to a history T1-T2-T3

b. 2PC History:

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

3. Consider the following tables:

Sales(Date,FKShop,FKCust,FKProd,UnitPrice,Q,TotPrice)
Shops(PKShop,Name,City,State,Area)
Customers(PKCust,Nome,FamName,City, State,Area)
Products(PKProd,Name,SubCategory,Category,Price)

With the following sizes:

Sales: NRec: 100.000.000, Npag: 1.000.000;
Shops: 500, 2; Customers: 10.000, 100; Products: 100.000, 10.000

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

```
SELECT P.PKProd, P.Name, P.Category, Sum(TotPrice)
FROM Sales S, Products P
WHERE S.FKProd=P.PKProd
GROUP BY P.PKProd, P.Name, P.Category
```

```
SELECT P.Category, Year(S.Date), Sum(TotPrice)
FROM Sales S, Customers C, Products P
WHERE S.FKCust=C.PKCust  AND   S.FKProd=P.PKProd
GROUP BY P.Category, C.Area
```

a. In the context of a traditional database, choose a physical organization to optimize both queries, and give a synthetic justification of your choice. Specify the primary organization of each table and which indices would you use. If the primary key of the Product and Customer relations must satisfy any property, specify this fact.
b. Show an access plan for the first query and compute its cost.
c. (Optional) How would the organization of (a) and the cost of (b) change if you added a condition AND Product.Category = 'Food', where 'Food' includes the 10% of the sales.

----

a. The first query can be executed in linear time if Sales is organized sequentially on FKProd and Products sequentially on PKProd: the join will just require a SortMerge of the two tables, and the result will be sorted on (Category, Name) if we assume that ordering Products on PKProd is the same as ordering them lexicographically on (Category, Name). If the Sales query where vertically partitioned the query would be even more efficient. For the second query, the join can be still executed in linear time since Customers fits main memory. However, the result will be sorted on (Category, Name), hence is not grouped in (Category,Area), hence must be sorted, which is quite expensive.

b. Access plan:

```
GroupBy(MergeJoin(TS(Products),TS(Sales),S.FKProd=P.PKProd),
        { P.PKProd, P.Name, P.Category},
        { Sum(TotPrice)}
       )
```

Cost: NPag(Products)+NPag(Sales) = 10.000+1.000.000 = 1.010.000

c. The condition Product.Category = 'Food' is best implemented by finding the first product with category 'Food' and scanning the two tables starting from that product. Since both tables are sorted on PKProd, whose order respect the Category order, such first product can be found using binary search with $\log_2(10.000)+\log_2(1.000.000)=13+20$ accesses. This number may be reduced by adding a sparse index on Product.Category, which we would use to find the first product and its PKProd, and a sparse index on Sales.FKProd.
Since both tables are clustered on Product.PKProd they are also clustered on Product.Category, hence the total cost is given by the cost of accessing the first element +
sf*(NPag(Products)+NPag(Sales)) = 33+0,1*(1.010.000) = 33+101.000 = 101.033

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

a. Column databases are better suited for OLAP applications than for OLTP applications. Why?
b. (Really really optional) List some features of column databases, or some techniques that are used by some of them, that are related with the recent trends in the evolution of hardware architectures. Please be brief and write clearly.

a. Column databases are based on three ideas, all of which are very useful for OLAP queries: (a) dividing the columns so that every query only accesses the columns it needs; (b) compressing the columns, so that every massive query transfers a smaller amount of data; (c) storing redundant Projections, so that every query finds the projection that is best suited; (d) optimizing the transfer of massive amount of data. (a) implies that the insertion of a single tuple must access many different pages (b) implies that every update must potentially compress and decompress the column (c) redundant storage implies that a single update must update more copies of the same item (d) massive access is essential in OLAP applications but has no relevance for OLTP applications.
b. The emphasis on linear scan is linked to the fact that modern disks have excellent linear scan speed. Use of compression is related to the increasing gap between disk speed and memory speed. The emphasis on fixed-size data structures is linked to SIMD parallelism of modern CPUs. The choice of moving from 'tuple at a time' to 'block at a time' is related to pipelined executing of instruction and to the importance of avoiding function calls, and to optimal use of instruction-cache. The choice of 'block-at-a-time' approach is also related to the need of doing optimal use of L3 cache.

5. Consider relations R(<u>K</u>,A, B)[NPag=1.000.000, NReg=100*NPag] and S(<u>K</u>,A, B)[NPag=100.000, NReg=100*NPag], both having K as their Key. Assume a Shared Nothing architecture, where the relations are horizontally partitioned and stored in nodes [n0,…,n99], and records are distributed using the function h1 applied to R.A for relation R, and the function h2 applied to S.B for relation S.
   a. Describe a hash-based algorithm to compute the intersection of R and S, that leaves each piece of the result on the disk of the node where it has been computed. Assume that every node has a buffer of 100 pages. When describing the actions in a node you may refer to standard hash based algorithms, such as hash-join or hash-based intersection.
   b. Assume that disks are able to read or write at a rate of 1.000 pages/s, and compute the cost of the I/O operations of each single node, including the initial read, the local intersection operation, the final local write. For each node specify how much data is read and written, and how much time is needed for that, assuming that, at each node, different i/o operations cannot overlap
   c. Optional: Assume that each node communicates with the network at a rate of 40 MBytes/s, that every page measures 4Kbytes, that the network itself has unlimited capacity, ignore the set-up time of the communications, and assume that, whenever a node sends a block of data the receiving node is ready to receive it. Compute the time of the communication phase, remembering that each node must both send and receive data, and that when a node is sending it cannot be receiving.

---

a. Every node uses h1(S.A) to distribute its S tuples to the other nodes. Every node performs a hashjoin, on the K attribute, on the S tuples that it receives, that measure 1.000 pages, and its portion of the R table, which measures 10.000. (That is, it uses a new function h3(K) to distribute the S tuples into 100 buckets, it reads the local part of R and distributes it into 100 buckets in the same way, then it rereads every pair of corresponding R-S buckets, computes the intersection, and store the intersection on the disk).
b. (1) Every node reads its portion of S: 1.000 reads, i.e., 1 sec. (2) while its tuples si from S arrive, each node computes h3(si.K) for each tuple and creates 100 buckets, each of size

NPag(S)/NNodes/100, that is 100.000/100/100, that is 10 pages each, hence each bucket will fit main memory. This phase costs 1.000 writes for each node, since each node receives NPag(S)/NNodes records, hence 1 sec. (3) each node executes the first hashjoin phase for its part of the R relation as well, hence it creates 100 buckets for R, and this costs a full read and a full write of the local copy of R, that is 2*(1.000.000/NNodes) = 2*10.000 pages = 20 sec (4) each node executes the second phase of hashjoin and writes the result, hence it must read once both R and S, which costs 1.000+10.000 pages – and it must write the result, which is as big as S in the worst case, that is 1.000 pages. Total cost is 12.000 pages, that is, 12 secs. The total cost is hence 1+1+20+12=34 secs.

c. Every node sends 1.000 pages and receives 1.000 pages, that is, 8Mbytes. With a rate of 40 Mbytes/s, this requires 0,2 seconds.

6. Consider a distributed DBMS composed by two separate nodes, US and EU, which store the schema of exercise (1), where Area belongs to {'US','EU'). Assume that each node issues the following amount of operations per day.

Table	Queries per day	Updates per day
Sales	100	200
Shops	100	1
Customers	100	1
Products	100	5

Every query that is issued by a node n1 (US or EU) but also involves data that is *only* found on the other node incurs a cost QC, and every update that is issued by a node n1 (US or EU) but also involves data that is *also* found on the other node incurs a cost UC. The 90% of the US operations that involve a Sale regard a Sale whose shop has Area='US', and similarly for the operations on the EU node. The tables Customers and Products are uniformly accessed.

a. Define how the tables may be horizontally fragmented according to the above description. One fragment may include one entire table or a portion.
b. Specify how would you allocate the fragments on the two nodes, according to the usage model described, assuming QC and QU are similar. Ignore the issue of failure resilience.
c. (Optional) Compute the total communication cost per day according to the above description.

---

a. Shops should be fragmented according to the US/EU division. Sales should be fragmented according to the result of a semijoin with the fragmented Shops table. The other tables should not be fragmented: although Customers have an Area attribute, it is not involved in the condition of the queries that we consider.
b. Each Sale fragment should only be allocated on the corresponding node, since their access is mostly local and they are heavily updated. All other tables should be duplicated since queries are much more common than updates.
c. The communication cost for each node is the following

Table	Cost of queries	Cost of updates	How is stored
Sales	100*10%*QC	200*10%*QU	Partitioned
Shops	0	1*QU	Replicated
Customers	0	1*QU	Replicated
Products	0	5*QU	Replicated

Total	10*QC	27*QU

If Sales were duplicated and Shops was not, we would have the following lines:

Table	Cost of queries	Cost of updates	How is stored
Sales	0	200*QU	Replicated
Shops	100*10%*QC	1*10%*QU	Partitioned