NATIONAL UNIVERSITY OF SINGAPORE

CS3223 – DATABASE SYSTEMS IMPLEMENTATION
(Semester 2: AY2017/18)

Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This assessment paper consists of **TWO** Sections and comprises **TWELVE (12)** printed pages. Section **A** has **FIVE (5)** questions, and Section **B** has **TWELVE (12)** questions. The maximum score is **FORTY** (40).
2. Answer **ALL** questions. Write your answers within the **space provided** in this booklet.
3. This is an **OPEN BOOK** assessment.
4. You are allowed to use an authorized calculator.
5. Please write your **Student Number** below. Do not write your name.

Student Number:												

EXAMINER'S USE ONLY	
Questions	**Marks**
Section A (28 marks)	
Q1 (8 marks)	
Q2 (3 marks)	
Q3 (8 marks)	
Q4 (6 marks)	
Q5 (3 marks)	
Section B (12 marks)	
Total (40 marks)	

Section A (28 marks)

Question 1. (8 marks) Consider the following database from a boat rental service:

Schema:

```
Boats(bid, type, price)
Sailors(sid, name, age)
LastRented (bid, sid, date)
```

Query:

```
SELECT S.name
FROM Boats B, Sailors S, LastRented L
WHERE B.bid = L.bid AND S.sid = L.sid
AND S.age ≤ 25 AND B.type = "tugboat"
AND B.price > 30000;
```

Data distribution:

```
Sailors.age ranges from 11 to 70
Boats.type has 10 distinct values
Boats.price ranges from 10,000 to 50,000
Boats.bid has 1,000 distinct values
Sailors.sid has 1,000 distinct values
```

The following information is given:

```
B+ tree on Boats.bid: unclustered, uses alternative 2 for data
entries (i.e., key-pointer pairs)
Index has 3 levels, i.e., takes 3 IO's to retrieve data entry
from the index
Boats: 40 bytes/tuple, 100 tuples/page, 10 pages
Sailors: 80 bytes/tuple, 50 tuples/page, 20 pages
LastRented: 20 bytes/tuple, 200 tuples/page, 100 pages
```

Access Methods:

```
Sailors: file scan
LastRented: file scan
Boats: B+ tree on bid
```

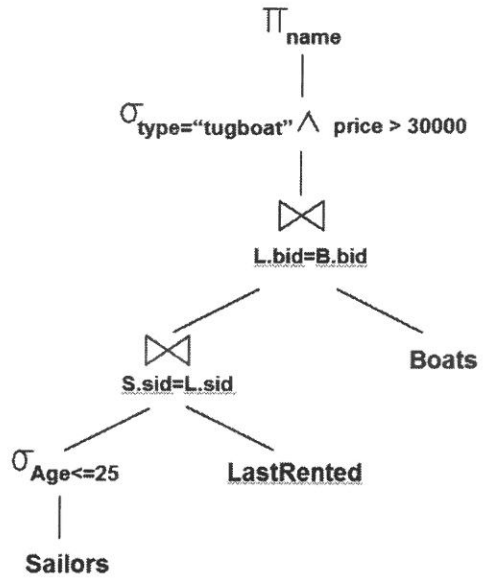
Join Algorithms:

```
Sailors with LastRented: page-oriented nested loops join
Intermediate result (of Sailors join with LastRented) with
Boats: index nested loops join
```

Operator Implementations

```
Selection (both instances): on the fly
Projection: on the fly
```

Consider the following *pipelined* query plan (i.e., the intermediate result is not materialized):



(A) What is the I/O cost for the subtree involving the join between Sailors and LastRented (output is pipelined to the next operator)?

(B) What is the number of output tuples for the first join operation between Sailors and LastRented?

(C) What is the cost of this plan (in terms of I/O)? There is no need to consider the output of the plan.

(D) For each of the following heuristics, state whether or not it will be beneficial in terms of reducing the number of I/Os. Justify your answers.

(i) Pushing down the selection on Boats.type and Boats.price below the join.

(ii) Projecting out Sailors.age before the join.

Question 2. (3 marks) Consider the Dynamic Programming query optimization algorithm that we discussed in class where we maintain only one optimal plan for each subplan (i.e., for each subset of relations, we maintained only one optimal plan for that subset). Suppose for a particular query the scheme generates the globally optimal solution (in the bushy search space) under its cost model. However, at runtime, the query plan may actually not be optimal. Explain why?

Question 3. (8 marks) Suppose we run the following 6 transactions using the validation-based protocol. There are no other transactions in the system. Suppose the database has only the set of given 8 objects: a, b, c, d, e, f, g, h. The values read set and write set of transaction T_6 is not given. The following table lists the transactions involved, together with their read and write sets:

Transaction	Read Set	Write Set
T_1	{a, b}	{b, c}
T_2	{a, b, c}	{h}
T_3	{b}	{d. e}
T_4	{c}	{f, g}
T_5	{a}	{d, f}
T_6	{XXX}	{YYY}

The following sequence of events takes place.

a) T_1, T_2, T_3, T_4 start (in this order)
b) T_3 initiates validation
c) T_5, T_6 start (in this order)
d) T_1 initiates validation
e) T_5 initiates validation
f) T_4 initiates validation
g) T_2 initiates validation
h) T_1, T_2, T_3 finish (if they were not aborted earlier)
i) T_6 initiates validation
j) T_4, T_5, T_6 finish (if they were not aborted earlier)

In the table below, write down the outcome of each transaction, either COMMITTED if it was successful at validation, or ABORTED if it was not. For transactions that are aborted, state the R-W and W-W conflicts that arise. We are also given that transaction T_6 commits successfully. What are the largest possible sets of values of XXX and YYY for this to happen?

Transaction	Outcome	Conflicts (if any)
T_1		
T_2		
T_3		
T_4		
T_5		
T_6	**Committed**	XXX = YYY =

Question 4. (6 marks) Consider the following transaction log from the start of the run of a database system that is capable of running undo/redo logging with checkpointing. Note that the log record is enclosed by < and >. The number besides each log record is just an identifier to refer to the log record; it is not part of the log records.

1. < START T1 >;
2. < T1, A, 49, 20 >;
3. < START T2 >;
4. < T1, B, 250, 20 >;
5. < START CKPT(T1, T2) >;
6. < T1, A, 75, 49 >;
7. < T2, C, 35, 20 >;
8. < T2, D, 45, 20 >;
9. < COMMIT T1 >;
10. < END CKPT >;
11. < START T3 >;
12. < T3, E, 55, 20 >;
13. < START CKPT(T2, T3, T4) >;
14. < T2, D, 46, 45 >;
15. < START T4 >;
16. < T4, F, 100, 20 >;
17. < T4, G, 111, 20 >;
18. < END CKPT >;
19. < T2, C, 65, 35 >;
20. < COMMIT T2 >;
21. < COMMIT T3 >;
22. < T4, F, 150, 100 >;
23. < COMMIT T4 >

Assume the log entries are in the format < Tid, Variable, Newvalue, Oldvalue >. Further assume that all the Oldvalue of objects A, B, C, D, E, F, G are initially on disk.

(A) Suppose the system crashes just **after** entry 7, i.e., < T2, C, 35, 20 >, is written to disk. What are the transactions to be undone and redone?

Undo:

Redo:

(B) Suppose the system crashes just **after** entry 12, i.e., < T3, E, 55, 20 >, is written to disk. What are the values of A, C, E on disk before and after the recovery? If the value is uncertain, then indicate all possible values.

	Before recovery	After recovery
A		
C		
E		

(C) Suppose the system crashes just **after** entry 21, i.e., <COMMIT T3>, is written to disk. For each of the transactions to be undone, list the identifier of the earliest record to be undone. For example, if you think that T1 has to be undone, and the earliest record is <T1, A, 75, 49>, then write <T1, 6>. Likewise, for each of the transactions to be redone, list the identifier of the earliest record to be redone.

Undo:

Redo:

Question 5. (3 marks) In our lecture, we have discussed 4 possible designs for log-based recovery mechanisms, Intuitively, NO UNDO/NO REDO seems to be very attractive as it simply means there is no need to do any recovery! Explain why such a scheme is not used in practice.