PAPER CODE NO.	EXAMINER: Dr. Rasmus Ibsen-Jensen	Tel. No.
COMP207	DEPARTMENT: Computer Science	

UNIVERSITY OF LIVERPOOL

FIRST SEMESTER EXAMINATIONS 2022/23

Mock Exam for Database Development

TIME ALLOWED : 3 Hours 30 Minutes

INSTRUCTIONS TO CANDIDATES

NAME OF CANDIDATE ..... SEAT NO .....
USUAL SIGNATURE .....

READ THE FOLLOWING CAREFULLY:

1. This exam paper consists of 30 questions. Each question comprises 5 statements, for which you should select the one most appropriate answer. The questions have the same weighting.
2. The exam mark is based on the overall number of correctly answered questions. The more questions you answer correctly the higher your mark, incorrectly answered questions do not count against you.
3. Enter your name and examination number IN PENCIL on the computer answer sheet according to the instructions on that sheet.
4. When you have completed this exam paper, read the instructions on the computer answer sheet carefully and transfer your answers from the exam paper. Use a HB pencil to mark the computer answer sheet and if you change your mind be sure to erase the mark you have made. You may then mark the alternative answer.
5. At the end of the examination, be absolutely sure to hand in BOTH this exam paper AND the computer answer sheet.
6. Calculators are NOT permitted.

THIS PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM

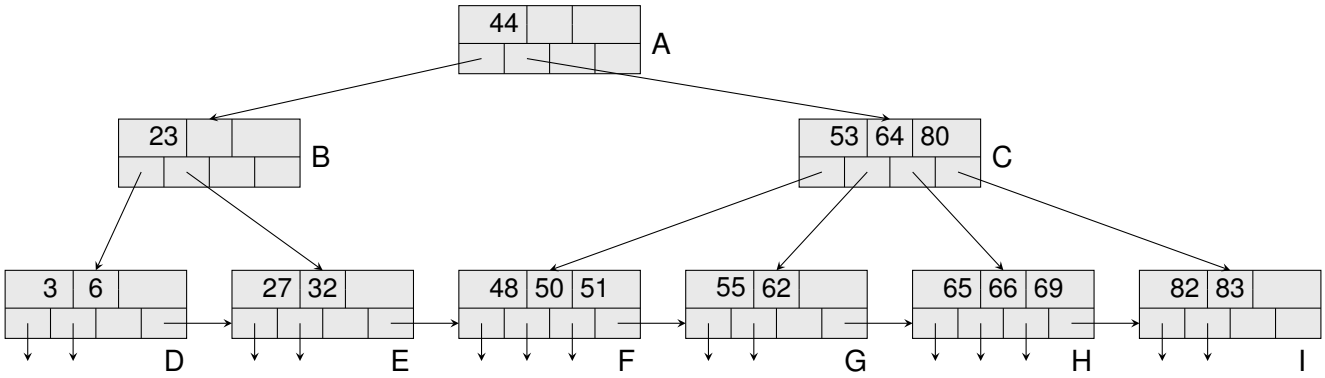

Figure 1: B+ tree for Question 1.

1. Consider the B+ tree in Figure 1. Because we use $n = 3$ for each node, we use $x = 2$ as the lower bound on the number of pointers a node can have. What happens if we delete 3? We find and delete 3 in node D and...

- ■ **A.** do nothing else.
- ☐ **B.** steal a pointer from node E and update the least common ancestor of E and D.
- ☐ **C.** merge with node E.
- ☐ **D.** merge with node E, move a pointer from node C to node B and update node A.
- ☐ **E.** split D.

2. A simple checkpoint is:

- ☐ **A.** A time point when all transactions commit.
- ☐ **B.** A time point when all transactions roll back.
- ☐ **C.** A time point when the DBMS is tested for efficiency
- ■ **D.** Point of synchronisation between database and log file. All buffers are flushed to secondary storage
- ☐ **E.** Point of synchronisation between database and log file. All buffers are deleted.

3. If a scheduler is *cascadeless*, then it is also:

(I) Strict
(II) Recoverable
(III) Serialisable

- ☐ **A.** Precisely (I) and (II)
- ■ **B.** Precisely (II)
- ☐ **C.** Precisely (I) and (III)
- ☐ **D.** All of the above

☐ **E.** None of the above

4. Consider the following schedule:

$$r_3(X);\ r_1(Y);\ w_2(Z);\ w_1(Y);\ r_2(X);\ w_1(Z);\ r_3(Y);\ w_3(Z)$$

This schedule is conflict-equivalent to which of the following serial schedules?

☐ **A.** $r_1(Y);\ w_1(Y);\ w_1(Z);\ w_2(Z);\ r_2(X);\ r_3(X);\ r_3(Y);\ w_3(Z)$

☐ **B.** $w_2(Z);\ r_2(X);\ r_3(X);\ r_3(Y);\ w_3(Z);\ r_1(Y);\ w_1(Y);\ w_1(Z)$

☐ **C.** $r_3(X);\ r_3(Y);\ w_3(Z);\ w_2(Z);\ r_2(X);\ r_1(Y);\ w_1(Y);\ w_1(Z)$

■ **D.** $w_2(Z);\ r_2(X);\ r_1(Y);\ w_1(Y);\ w_1(Z);\ r_3(X);\ r_3(Y);\ w_3(Z)$

☐ **E.** $r_3(X);\ r_3(Y);\ w_3(Z);\ r_1(Y);\ w_1(Y);\ w_1(Z);\ w_2(Z);\ r_2(X)$

5. If the wound-wait scheme is used for deadlock prevention, what happens if an younger transaction wants a lock on an item, but that lock is held by a older transaction?

☐ **A.** Both transactions are aborted.

☐ **B.** The older transaction is aborted.

☐ **C.** The younger transaction is aborted.

☐ **D.** The older transaction may wait for the younger transaction to finish.

■ **E.** The younger transaction may wait for the older transaction to finish.

6. An organisation uses a distributed database over two sites, A and B:

- Site A holds a relation *Passenger*(*passenger_id*, *first_name*, *last_name*). Values for each of the attributes, i.e. *passenger_id*, *first_name* and *last_name* require 30 byte each.
- Site B holds a relation *Flights*(*flight_id*, *time*, *seat*, *passenger_id*).
  Each value of attributes *flight_id* and *passenger_id* require 30 byte, each value of *seat* require 15 byte, and each value for attribute *time* requires 25 byte.

Assume the following:

- $|\pi_{\text{passenger\_id}}(\text{Flights})| = 100000$
- $|\pi_{\text{passenger\_id}}(\sigma_{\text{time='10/12/2018 at 10:00'}}(\text{Flights}))| = 1000$
- $|\text{Passenger} \ltimes \sigma_{\text{time='10/12/2018 at 10:00'}}(\text{Flights})| = 1000$
- $|\pi_{\text{flight\_id}}(\sigma_{\text{time='10/12/2018 at 10:00'}}(\text{Flights}) \ltimes \text{Passenger})| = 10$
- $|\text{Passenger}| = 100000$

To execute the query $\pi_{\text{first\_name,last\_name,flight\_id,time}}(\text{Passenger} \bowtie \sigma_{\text{time='10/12/2018 at 10:00'}}(\text{Flights}))$ at site B, how many bytes have to be transferred between A and B at a minimum?

☐ **A.** 300 + 600 = 90 bytes

☐ **B.** 30000 + 60000 = 90000 bytes

■ **C.** 30000 + 90000 = 120000 bytes

☐ **D.** 100000 + 60000 = 160000 bytes

☐ **E.** 9000000 bytes

7. What does fragmentation transparency refer to?

☐ **A.** The database may not use fragmentation

☐ **B.** The database must use fragmentation

■ **C.** How the database is divided up over a distributed database does not matter for how to write queries for it

☐ **D.** Certain relations in the database are invisible

☐ **E.** None of the above

8. A scheduler is *serial* if:

☐ **A.** There are no cycle in the precedence graph.

☐ **B.** All unlocks of locks that can write happens after commit

☐ **C.** Reads is only done to variables after the last transaction that wrote to that variable has committed

■ **D.** For each pair of transactions, all the operations from one happens before all operations of the other or vice versa

☐ **E.** None of the above

9. Consider the scenario of a bank that uses a database with a relation Accounts(account_no, balance) to keep track of the balance for the different accounts. Assume that two transactions T_1 and T_2 execute on the database in the following way:

Time	Event
2	Transaction T_1 adds 100 *$* to account 123 and commits
3	Transaction T_2 checks how much money is on account 123
3	Transaction T_2 removes 100 *$* from account 123
4	⚡
5	Transaction T_2 adds 100 *$* to account 456 and commits

The "⚡" at time 4 indicates a power failure. Assuming the DBMS does not prevent these transactions from executing and there is no changes to the database once it is restarted, which of the ACID properties, besides Consistency, would be violated?

■ **A.** Only Atomicity

☐ **B.** Atomicity and Isolation

☐ **C.** Only Isolation

☐ **D.** Isolation and Durability

☐ **E.** Only Durability

10. Inheritance are part of which type of databases:

- ☐ **A.** No-SQL databases
- ☐ **B.** Relational databases
- ■ **C.** Object-Oriented and Object-Relational databases
- ☐ **D.** Distributed databases
- ☐ **E.** None of the above are correct

11. What is the definition of *shading* in distributed databases?

- ☐ **A.** The database has divided a relation over a network of computers such that each computer stores a disjoint subset of the columns
- ☐ **B.** The database has divided the columns over a network of computers such that each computer stores a subset of columns, together with the primary key for the relation
- ☐ **C.** Each part of the database is stored on multiple computers in the network
- ■ **D.** The database has divided a relation over a network of computers such that each computer stores a subset of the rows
- ☐ **E.** None of the above.

12. Given the following XQuery, what is the equivalent SQL statement?

```
FOR $v IN $doc//student
WHERE $v/year = 2
RETURN $v/name
```

- ☐ **A.** `SELECT student FROM name WHERE year = 2`
- ☐ **B.** `SELECT name, year FROM student WHERE year = 2`
- ☐ **C.** `SELECT student FROM year WHERE name = 2`
- ■ **D.** `SELECT name FROM student WHERE year = 2`
- ☐ **E.** `SELECT year FROM student WHERE name = 2`

13. Consider a schedule over some number of transactions. However, before starting, the computer it is running on has a power failure and none of the schedule is done after restart. Which of the ACID properties are broken?

- ☐ **A.** Atomicity
- ☐ **B.** Consistency
- ☐ **C.** Isolation
- ☐ **D.** Durability
- ■ **E.** None of them

14. Assume a database with schema

Movie(id, name, genre, rating)

and the following characteristics:

Movie contains 1200 tuples, exactly 200 of them with a value of 'Action' for the 'genre' attribute. The entire relation requires 240 blocks on disk.

What is the size estimate of the selection $\sigma_{\text{genre='Action'}}$(Movie) and how many disk blocks are needed to store the result of the selection?

- ☐ **A.** Size estimate: 1200; number of blocks: 240
- ■ **B.** Size estimate: 200; number of blocks: 40
- ☐ **C.** Size estimate: 1200; number of blocks: 80
- ☐ **D.** Size estimate: 100; number of blocks: 40
- ☐ **E.** Size estimate: 200; number of blocks: 20

15. Which of the following is a characteristic of a star schema

- ☐ **A.** They have few rows
- ☐ **B.** They are typically used if there are many users with simple queries
- ☐ **C.** For each pair of schemas, at least one contains the primary key of the other
- ■ **D.** There is a special schema such that each other schema has its primary key in that schema
- ☐ **E.** None of the above.

16. Locks are useful for ensuring some of the ACID properties. Which?

- ☐ **A.** Atomicity and Consistency
- ☐ **B.** Atomicity and Isolation
- ☐ **C.** Atomicity and Durability
- ■ **D.** Consistency and Isolation
- ☐ **E.** Consistency and Durability

name	movies
Anna	BI, BS, BU
Ben	HP1, HP2, HP3
Chloe	BI, HP1
Dave	BI, HP1, HP2
Emma	BI, BS, HP1
Fred	BI, BU
Gwen	BS, HP1, HP2
Henry	BS, HP1, HP2

Table 1: Table of movie viewers. Each row corresponds to the movies watched by the individual.

17. Table 1 contains records of movie watching. Each row states which movies where watched by the person that has the name contained in the first column. What is the *support* for $J = \{BI, HP1\}$?

- ☐ **A.** $\frac{0}{8}$
- ☐ **B.** $\frac{1}{8}$
- ☐ **C.** $\frac{2}{8}$
- ■ **D.** $\frac{3}{8}$
- ☐ **E.** $\frac{4}{8}$

18. With reference to Table 1, which of the following association rules (AR) have a confidence of 100%?

- ☐ **A.** BI, BU $\rightarrow$ BS
- ☐ **B.** BS, HP1$\rightarrow$ HP2
- ■ **C.** BS, HP2$\rightarrow$ HP1
- ☐ **D.** HP1, HP2$\rightarrow$ HP3
- ☐ **E.** BI, HP1$\rightarrow$ HP2

```
<?xml version="1.0" encoding="UTF-8"?>
<breakfast_menu>
    <food>
            <name>Belgian Waffles</name>
            <price currency="pounds">3</price>
            <price currency="euros">4</price>
            <weight>300g</weight>
    </food>
    <food>
        <name>Strawberry Belgian Waffles</name>
        <price currency="pounds">5</price>
        <price currency="euros">6</price>
        <weight>450g</weight>
    </food>
    <food>
            <name>Blueberry Belgian Waffles</name>
        <price currency="pounds">4</price>
        <price currency="euros">5</price>
        <weight>400g</weight>
    </food>
    <food>
        <name>Cereal</name>
        <price currency="pounds">1</price>
        <weight>200g</weight>
    </food>
    <food>
        <name>Full-English Breakfast</name>
            <price currency="pounds">5</price>
            <price currency="euros">6</price>
    </food>
</breakfast_menu>
```

Figure 2: XML document `breakfast_menu.xml` for Questions 19–20.

19. Given the XML document in Figure 2 on the preceding page, what does the XPath query

```
(//food/price)[1]
```

do on this XML document?

- ■ **A.** It selects the first `price` element below a `food` element.
- ☐ **B.** It selects each `price` element below a `food` element.
- ☐ **C.** It selects the first `price` element of each `food` element.
- ☐ **D.** It selects all `price` elements below a `food` element that contains number 1.
- ☐ **E.** It selects the last `price` element below a `food` element.
- ☐ **F.** It selects all `price[1]` elements below a `food` element.

20. Given the XML document in Figure 2 on the facing page, what is the correct XQuery to output all the names of food items that has some price above 3?

- ☐ **A.**
```
let $x in $doc/breakfast_menu/food/name
where $x/../price >3
return $x
```
- ☐ **B.**
```
for $x in $doc/breakfast_menu/food/name
where $x/../@price > 3
return $x
```
- ☐ **C.**
```
for $x in $doc/breakfast_menu/food/name
where $x/../price >3
output $x/actor
```
- ■ **D.**
```
for $x in $doc/breakfast_menu/food
where $x/price > 3
return $x/name
```
- ☐ **E.**
```
let $x in $doc/breakfast_menu/food
where $x/price > 3
return $x/name
```