THE UNIVERSITY OF HONG KONG

FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE

COMP7104/DASC7104 - Advanced Database Systems

Date: December 11, 2020 **Time: 6:30pm - 8:30pm**

There are 39 questions overall, divided into two sections as follows:

Section 1 – Multi-choice questions – 25 questions
1 point per question – 25/100 points in total

Section 2 – Detailed-answer questions – 14 questions
75/100 points in total

Answer ALL questions.

As the exam is open book, your Zoom webcam should be set to capture your eyes and the immediate environment only.

Answers can be presented either typed on a computer or handwritten.

If you type your answers during the examination, you can do so with any text editor, but the document submitted shall either be .doc, .docx, or .pdf (.zip archives are not allowed).

No calculators are necessary. You can leave arithmetic expressions unsolved, as long they are correct this will be considered a valid answer.

General indications

- 1 KB = 1024 bytes; i.e., we will be using powers of 2, not powers of 10
- You can leave arithmetic expressions unsolved, as long they are correct this will be considered a valid answer.

Section 1 – Multi-choice questions – 25 questions – 1 point per question

Write down the true answer(s) to each question. Each question may have zero, one, or several valid answers. For each question, only the identification of all and only the valid answers will allow you to obtain the assigned point.

Q1.1 Which of the following assertions about storage devices are true?
A) Flash memory is preferable to disk because flash allows for fine-grained writes
B) When looking for a 10 byte record (e.g., by record Id) on disk, more than 10 bytes of data may be read from that disk
C) Reading from an SSD disk is more costly than writing to an SSD disk

Q1.2 Which of the following assertions about storage devices are true?
A) In order to read something from the disk, DBMSs use a pointer "deref" (dereference) approach
B) Magnetic hard disks need to constantly re-organize the data stored on them (wear-levelling) to prevent failure
C) On flash (SSD) disks, sequential reads and random reads can be done with comparable speed, since these disks use NAND technology instead of traditional spinning magnetic disks

Q1.3 Which of the following assertions about DB files are true?
A) In a heap file, all pages must be filled to capacity except the last page.
B) Assuming integers take 4 bytes and pointers to pages take 4 bytes, a slot directory page that is 512 bytes can address 63 data pages
C) In a page containing fixed-length records without nullable fields, the size of the bitmap never changes.

For the next two questions, assume we have a heap file A implemented with a page directory. One page in the directory can hold 16 page entries. There are 54 pages in file A in total.

Q1.4 In average, how many IOs may be required to find a page with enough free space?
A) 1 I/Os
B) 2 I/Os
C) 3 I/Os

Q1.5 In the best case, how many IOs are required to write a record to a page with enough free space.

A) 2 I/Os

B) 3 I/Os

C) 4 I/Os

Q1.6. Which of the following assertions about DB files are true?

A) Fragmentation may be an issue for packed fixed-length record pages

B) Fragmentation may be an issue for slotted pages of variable-length records

C) For pages of fixed-length records, a slot directory has a better space complexity than a bitmap

Q1.7 The chaining of the leaves in a B+-tree index is useful for

A) The procedure of expanding leaves, when inserting new values in the index

B) The search procedure

C) Maintaining the overall consistency of the index tree

Q1.8 What is the disadvantage of an interval search with a B+-tree index?

A) It may cause random (non-sequential) reads at the leaves level

B) It may cause random (non-sequential) reads of blocks in the data file

C) It may cause a sequential scan of part of the data file

Q1.9 Which of the following assertions about DB indexing with B+-trees are true?

A) In a clustered index, the data pages are entirely sorted using the same index that build the B+ tree

B) In an alternative 1 index organization (by value), the entire records are directly stored in the leaf pages

C) In an alternative 3 index (by reference), index entries of the form (key; …) are stored in leaf pages, where each key is distinct

Q1.10 Which of the following assertions about DB indexing with B+-trees are true?

A) A B+ tree has always a balanced height

B) The keys on an internal node are always in sorted order

C) Alternative 3 is always better than alternative 2

D) The number of records in the table must be known before performing bulk loading

Q1.11 Which of the following assertions about buffer management in DBMSs are true?

A) MRU prevents sequential flooding during sequential scans

B) We use the clock replacement policy over LRU in order to simplify the logic of the replacement policy

C) When using the LRU policy, the reference bits are used to give pages a "second chance" to stay in memory before they are replaced

Q1.12 Which of the following assertions about buffer management in DBMSs are true?

A. Each frame in the buffer pool is the size of a disk page

B. The buffer manager code (rather than the file/index management code) is responsible for turning on the dirty bit of a page

C. The dirty bit is used to track the popularity of the page

D. A sequential scan of the file will have the same hit rate using either MRU or LRU (starting with an empty buffer pool) when the file is smaller than the buffer pool.

Q1.13 Which of the following assertions about out-of-core sorting / hashing are true?

A) To finish the final pass of external hashing (conquer), we use a coarse-grained hash function, h_r, that is different from the hash functions used in the previous passes

B) De-duplicating a file using hashing is always more efficient than sorting in terms of I/O cost

C) Given any hashed file, there is some permutation of pages such that the resulting file is sorted

D) When the hashing algorithm does no recursive partitioning, the first pass serves as a divide phase, and the second pass serves as the conquer phase

Q1.14 Which of the following assertions about join algorithms are true?

A) Performing an index nested loop join with a clustered index on the inner relation always requires more I/Os than an index nested loop join with an unclustered index on the inner relation (for same tables)

B. Grace hash join is usually the best algorithm for joins in which the join condition includes an inequality (i.e. col1 < col2)

C. Performing key lookup over hash partitioned data requires fewer I/Os than if the data had been round-robin partitioned

Q1.15 Which of the following assertions about join algorithms are true?

A. Block (a.k.a. Chunk) Nested Loops join will always perform at least as well as Page Nested Loops join when it comes to minimizing I/Os

B. In choosing a join order for nested loops join to minimize I/Os, it is best to make the smaller relation the "outer" part of the loop

C. If we are joining two tables that are very different in size, by choosing a join order for index nested loops join to minimize I/Os, if both relations have indexes on their join column, it is best to query the index of the smaller relation

Q1.16 Which of the following assertions about SystemR/Selinger query optimization algorithm are true?

A) It never considers plans with cartesian products because they are suboptimal

B) It only explores right deep plans

C) It may produce a sub-optimal query plan, even when given a perfect cost estimator

D) The running time of the algorithm is at least exponential in the number of joins

Q1.17 Which of the following properties are true about the 2PL protocol?

A) 2PL must be used, or we cannot guarantee conflict serializability

B) 2PL must be used, or we cannot guarantee serializability

C) 2PL ensures that we do not have cascading aborts

Q1.18 Which of the following properties are true about the 2PL / strict 2PL protocol?

A) In strict 2PL, we can give up locks after aborting but before rollback is complete

B) Some conflict serializable schedules cannot be produced when using 2PL

C) Schedules that are conflict-serializable will not produce a cyclic dependency graph

D) Both strict 2PL and 2PL enforce conflict serializability

E) All schedules that are conflict serializable are view serializable

Q1.19 Which of the following statements are true about deadlock avoidance strategies in RDBMSs?

A) The wait-die avoidance strategy avoids all deadlocks

B) This wound-wait avoidance strategy avoids all deadlocks

C) Both avoidance strategies never abort transactions unnecessarily

Q1. 20 Which of the following assertions are true about the ARIES crash recovery algorithm?

A) CLRs are never undone during ARIES recovery

B) We remove a page from the dirty page table each time we write a CLR to the log

C) The transaction table and dirty page table in the End Checkpoint record are up-to-date as of the time when the End Checkpoint record was logged

D) ARIES never causes a database disk page to be overwritten by a page with the same PageLSN

Q1.21 Consider the following log. If the flushedLSN is 50, under WAL, which of the following scenario(s) is possible?

LSN	Record	prevLSN
10	UPDATE: T1 writes P1	null
20	UPDATE: T2 writes P2	null
30	Begin Checkpoint	-
40	End Checkpoint	-
50	UPDATE: T2 writes P2	20
60	UPDATE: T1 writes P3	10

A) No dirty pages have been flushed to disk

B) The page updated at LSN 50 has been flushed to disk but the page updated at LSN 10 has not

C) The page updated at LSN 50 has been flushed to disk but the page updated at LSN 20 has not

D) All dirty pages have been flushed to disk.

Q1.22 Which of the following assertions are true about the 2PC distributed commit protocol?

A) Distributed deadlock occurs only if the waits-for graph at a node forms a cycle

B) In 2PC, we need all participant nodes to agree on abort (vote NO) to abort a transaction

C) A participant machine that does not flush its prepare records to the log before responding to the coordinator could violate the durability property

D) In 2PC phase 2, after the coordinator sends commit message to all participants, participants first respond with Ack, and then generate and flush commit record

Q1.23 Which of the following assertions are true about PDBMS query evaluation?

A) Pipeline parallelism scales up with the amount of data

B) Partition parallelism scales up with the amount of data

C) Performing key lookup over hash partitioned data requires fewer I/Os than if the data had been round-robin partitioned.

Q1.24 Which of the following assertions are true about PDBMS query evaluation?

A) Pipeline parallelism scales up to the pipeline depth

B) There is an extra disk I/O cost for parallel Grace Hash Join due to repartitioning (shuffling) the data across machines.

C) If we have little main memory, the parallel hash join is a pipeline breaker.

Q1.25 Which of the following drawbacks are true of 2PC-controlled replication?

A) A failed participant node can cause live nodes to abort

B) A failed coordinator node can freeze up the entire system indefinitely

C) Message latency for 2PC across wide-area networks can be high, and lead to slow transaction performance.

Section 2 – Detailed-answer questions – 14 questions – 75 points in total

Q2.1 – DB files (1.5 points) Assume you have a heap file implemented in a linked list structure (with a header page connected to a linked list of full pages and a linked list of pages with free space) with 5 full pages and 10 pages with free space. In the worst-case, how many pages would you have to read to see if there is a page with enough space to store some record?

Q2.2 – DB files (1.5 points) Same question as Q2.1, but now what if instead you had a page directory implementation of a heap file with 8 directory entries per header page (still 5 full pages, and 10 with free space)?

Q2.3 – DB files (4 points) Consider the following table schema:

```
Inventory (itemId integer PRIMARY KEY,
           quantity integer,
           description varchar2(20));
```

(a) (2 points) Given the schema above, at most how many variable-length records from this table can be stored in a 64 KB page? Assumptions: each page contains a 10-byte footer (including the pointer to free space) as well as a slot directory where it takes 4 bytes/record to store the pointer to the record in the page and 4 bytes/record to store the length/offset of that record; integers are 4 bytes each; records do not have record headers or pointers to the variable-length fields in the records.

(b) (2 points) Assuming we are using the same schema from the previous question, but now each record also has a record header of 4 bytes, and for each variable-length field in the record, it has a 32-bit pointer to the end of the field's value, at least how long in bits would each record be?

Q2.4. – Buffer Management (11 points) Assume you start off with 4 empty buffer frames. Assume the following page access pattern: A P R O P E R C O P P E R C O F F E E P O T

(a) (1.5 points) Using the LRU replacement policy, how many buffer pool hits are there?

(b) (1.5 points) Using the LRU replacement policy, what pages are in the buffer at the end?

(c) (1.5 points) Using the MRU replacement policy, how many buffer pool hits are there?

(d) (1.5 points) Using the MRU replacement policy, what pages are in the buffer at the end?

(e) (1.5 points) Using the Clock replacement policy, how many buffer pool hits are there?

(f) (1.5 points) Using the Clock replacement policy, what pages are in the buffer at the end?

(g) (2 points) How many reference bits are set at the end of the Clock algorithm?

Q2.5 - External hashing (6 points) Consider the following situations for external (out-of-core) hashing.

(a) (1.5 points) If we had 10 buffer pages to externally hash elements, what is the maximum number of pages we could externally hash and guarantee that we would not have to do recursive partitioning?

(b) (1.5 points) If we had 20 buffer pages to externally hash elements, what is the minimum number of pages we could externally hash to guarantee that we would have to use recursive partitioning?

(c) (1.5 points) If we had 15 buffer pages to externally hash elements, and a hash function that partitions elements perfectly evenly, what is the maximum number of pages we could externally hash and guarantee that we would not have to do recursive partitioning?

(d) (1.5 points) Assume that we have 64 KB of memory with 4 KB pages, what would be the cost in I/Os to externally hash a 128-page file? Assume that only one partition of 20 pages needs to be recursively partitioned once.

Q2.6 - External sorting (8 points) For general external merge sort in 2 passes, fill in the blanks with the appropriate numerical expressions, where we have a 150 pages file, B=23 pages of RAM (buffer pages).

(a) (1 point) How many input buffers will be used in pass 0?

(b) (1 point) How many output buffers will be used in pass 0?

(c) (1 point) How many sorted runs will be produced in pass 0?

(d) (1 point) What is the maximum number of pages that can be in the sorted runs produce in pass 0?

(e) (1 point) How many input buffers can be used in pass 1?

(f) (1 point) How many output buffers can be used in pass 1?

(g) (1 point) How many sorted runs will be produced in pass 1?

(h) (1 point) How many pages will be in the sorted run(s) produced pass 1?

Q2.7 – Join algorithms (3 points) Given a relation R of 1000 pages, a relation S of 200 pages, a buffer of size B=50.

(a) (1.5 points) What is the cost of joining R and S using sort-merge join? (Consider the cost of read and write for sorting; this is ordinary sort-merge, not optimized sort-merge, so the sort and merging steps are fully independent).

(b) (1.5 points) What is the minimum number of buffer pages required for the cost of the above question to remain the same?

Q2.8 – Query optimization (6 points). We consider the following schema:

```
Laptops(lid INTEGER PRIMARY KEY,
        brand VARCHAR(50),
        price INTEGER);

Users(uid INTEGER PRIMARY KEY,
      name VARCHAR(50),
      age INTEGER);

LastUsed(lid INTEGER REFERENCES Laptops(lid),
uid INTEGER REFERENCES Users(uid),
date DATE,
PRIMARY KEY (lid, uid));
```

We make the following assumptions about the distribution of data:

- $10 \leq$ Users.age < 85
- $1000 \leq$ Laptops.price $< 5,000$
- Users.uid has 1000 distinct values
- Laptops.lid has 1000 distinct values
- Laptops.brand has 10 distinct values
- Users.age is independent from Laptops.brand

Consider the following query:

```
SELECT U.name
FROM Laptops L, Users U, LastUsed LU
WHERE L.lid = LU.lid AND U.uid = LU.uid
AND (U.age < 25 OR L.brand = 'PC')
AND L.price >= 3000;
```

Compute the selectivity for each of the following predicates from the WHERE clause. Write only your final answer, as a simple fraction.

(a) (1.5 points) L.lid = LU.lid

(b) (1.5 points) U.uid = LU.uid

(c) (1.5 points) L.price >= 3000

(d) (1.5 points) U.age < 25 OR L.brand = 'PC'

Q2.9 (4 points) Consider the following schema:

```
Product(pid INTEGER PRIMARY KEY,
        name varchar2(20),
        price INTEGER);
Company(cid INTEGER,
        pid INTEGER REFERENCES Product(pid),
        name varchar2(20),
        PRIMARY KEY (cid, pid));
```

- Product contains 20,000 tuples, and each record is 20 bytes long.
- Company contains 1,000 tuples, and each record is 25 bytes long.
- Each page can hold 5,000 bytes.
- The buffer pool is 102 pages large.
- The fill factor for all hash tables is 0.8.
- There are no indexes.

Considering all the join algorithms covered so far in this class, what is the minimum estimated I/O cost to execute the following query? Exclude the final write from your solution.

SELECT P.name, C.name FROM Product P, Company C WHERE P.pid = C.pid

Q2.10 - Concurrency control (6 points) Consider the following schedule (time flows top-to-bottom, each line corresponds to one clock "tick").

	T1	T2	T3	T4
1	R(A)			
2		R(A)		
3			R(C)	
4			W(C)	
5		R(B)		
6		W(B)		
7	R(B)			
8				R(B)
9	R(C)			
10				R(C)
11				W(B)
12		COM		
13			COM	
14	COM			
15				COM

(a) (2 points) Draw the conflict graph for this schedule.

(b) (2 points) Which of the two locking disciplines learned in class could have produced this schedule?

(c) (2 points) Which of the following schedules below are conflict equivalent to the schedule above?

A. T3, T1, T2, T4

B. T2, T3, T1, T4

C. T4, T3, T1, T2

D. T3, T2, T1, T4

Q2.11 – Crash recovery (7 points) Consider the following log, recovered after a crash. T1, T2, and T3 are the only transactions. All pages were flushed to disk at and including LSN 50, so the log record has been truncated to start at LSN 60. For the following questions, assume we have not flushed any pages to disk after LSN 50, that is, all pages on disk have pageLSN < 60.

LSN	Record	prevLSN
...	...	...
60	UPDATE: T1 writes P2	null
70	UPDATE: T2 writes P3	40
80	Begin Checkpoint	-
90	End Checkpoint	-
100	UPDATE: T2 writes P1	70
110	UPDATE: T1 writes P3	60
120	UPDATE: T3 writes P1	null
130	T1 ABORT	110
140	CLR: T1 LSN 110, undoNextLSN: 60	130
150	T2 COMMIT	100
	CRASH	

(a) (2 points) Give the transaction table (order by TID) and dirty page table (order by PID) as recorded in the EndCheckpoint record.

(b) (2 points) Now, fill out the transaction table (order by TID) and dirty page table (order by PID) for the same log after the analysis phase.

(c) (1.5 points) After the UNDO phase is complete, which of the record(s) below are added to log?

A. a CLR for T1's update at LSN 60

B. a CLR for T2's update at LSN 100

C. a CLR for T2's update at LSN 70

D. a CLR for T3's update at LSN 120

(d) (1.5 points) Consider a slightly different scenario, in which the records at LSN 90 and 100 are in opposite order (that is, the UPDATE occurs before the End Checkpoint). Of the following, choose the most appropriate answer:

A. The dirty page table stored in the checkpoint must be different than in the original scenario

B. The dirty page table stored in the checkpoint could be different than in the original scenario

C. The dirty page table stored in the checkpoint must be the same as in the original scenario

Q2.12 – PDBMS (7 points) For the following questions, assume that you have access to the following relations:

Players (name, team, position, salary, agent)

Coaches (name, team, salary)

Important parameters for this question are summarized in the table below:

variable	symbol	value
Number of machines	M	4
Size of Page	s	4 KB
Pages in RAM per machine for joins	B	8 pages
Size of `Players`	[P]	128 pages
Size of `Coaches`	[C]	4 pages
Time for each I/O	t	5 ms

Assume we have 4 machines, each with 8 pages in memory for joins. We will need to measure time, so assume that an I/O takes 5ms. For the following questions, when we ask for execution time we are only concerned with the time associated with I/Os (e.g. assume CPU and other costs are negligible). Assume that we can send individual tuples over the network with no overhead in terms of network cost.

The first two questions will deal with the following query:

```
SELECT p.name, c.name
FROM Players p, Coaches c
WHERE c.team = p.team;
```

(a) (2 points) Assuming the Players and Coaches relations are both round-robin partitioned across 4 machines, what is the largest amount of data we ship across the network, in KB, of the query above, assuming we want to execute a sort-merge join?

(b) (2 points) Assuming the Players relation starts out hash-partitioned on the position key across the 4 machines, and that 75% of Players gets mapped to one machine, how long in ms will it take to complete a parallel scan of Players?

Suppose we now add another table with the following schema:

Fans (name, team)

which has 40,000 pages and is round-robin partitioned across 4 new machines with only this data.

The next two questions deal with the following query:

```
SELECT f.name, c.name
FROM Coaches c, Fans f
WHERE f.team = c.team;
```

(c) (1 point) What is the name of the join strategy that provides lowest possible network cost (amount of data shipped) to execute the query above?

(d) (2 points) What is the amount of data shipped in KB to execute that join strategy?

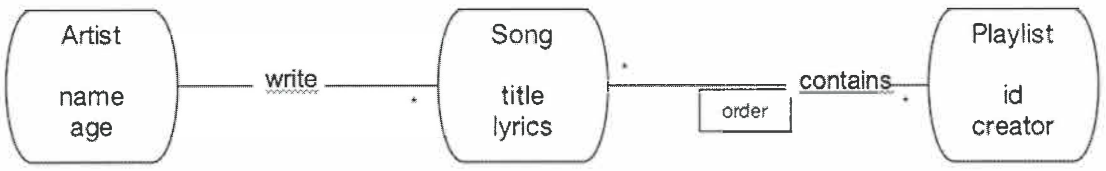
Q2.13. – Distributed transactions (4 points) Our database runs on 4 machines and uses 2PC. Machine 1 is the Coordinator, while Machines 2, 3, and 4 are Participants. Suppose our machines are connected such that the time it takes to send a message from Machine i to Machine j is 100xmax(i,j) milliseconds. Assume these communication latencies are symmetric: it takes the same amount of time to send from i to j as it takes to send from j to i. For example, sending a message between Machine 2 and Machine 4 takes 400=100x4 milliseconds in either direction, while sending a message between Machine 1 and Machine 2 takes 200=100x2 milliseconds in either direction, etc.

Assume that the transaction will commit (i.e., all subordinates vote YES), and that everything is instantaneous except for the time spent sending messages between two machines.

(a) (2 points) How much time passes from when Machine 1 sends its first message to when Machine 1 sends its second message, and what are these two messages?

(b) (2 points) How much time passes from when Machine 2 sends its first message to when Machine 2 sends its second message, and what are these two messages?

Q2.14 – NoSQL Cassandra (6 points) We are modeling in the Apache Cassandra NoSQL system a database for an online music service, with the following conceptual description (the star symbol indicates that several entities of one entity type can be in specific relationship with an entity of another type; for example, an artist may write several songs, while each song is written by exactly one artist):



More precisely, we have therefore songs, each written by an artist, and playlists consisting of an ordered list of songs. A relational-style database for this application could be defined as follows:

```
Artist (id int not null, name varchar(50), age int, primary key(id))
Song (id int not null, title varchar(50), lyrics text, primary key(id))
Playlist (id int not null, creator varchar(50), primary key(id))
SongInPlaylist (id_playlist int not null, id_song int not null, position int,
                primary key(id_playlist, id_song))
```

(a) (1 point) Remember that in Apache Cassandra, we will choose the schema depending on the access patterns, the queries that we expect to run on the data. Here are two such access patterns: "find all song titles" and "find all song titles of a particular playlist". Which of these two access patterns may be problematic over the previous relational-style schema in a Big Data context (if used in Cassandra)?

Assume now that we create for this application the following table in Cassandra:

```
Playlists (id_playlist int, creator text, song_order int, song_id int, title text, lyrics text, name text,
age int, PRIMARY KEY (id_playlist, song_order ) ).
```

(b) (1 point) How many insertions may be necessary in the worst-case in order to add a song in a playlist (compare the initial relational-style schema with the Playlists one) ?

(c) (1 point) Can we "find all song titles of a particular playlist" with the Playlists table, and if yes write the CQL query for it?

(d) (1 point) How many rows must be updated in the Playlists table when the age of an artist changes, compared with the initial relational-style schema?

(e) (1 point) Note that the identifier of the Playlists table is compound (id_playlist, song_order). In Apache Cassandra, a compound primary key consists of the partition key and the clustering key. The partition key determines which node stores the data. Rows for a partition key are then stored in order based on the clustering key. In this case, which of the following assertions are true?

A) A song is store on a single node
B) The songs of a playlist are all stored on a single node
C) The songs stored by a node are sorted by their Id
D) The songs of a playlist are stored one after the other

(f) (1 point) Explain with your own words how each of the two access patterns could be evaluated over the Playlists table in Apache Cassandra.

END OF PAPER