THE UNIVERSITY OF HONG KONG

FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE

COMP7104 Advance Database Systems

Date: December 11, 2019 **Time: 6:30 – 8:30pm**

There are 45 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 – 20 questions**
75/100 points in total

Answer ALL questions.

Only approved calculators as announced by the Examinations Secretary can be used in this examination. It is candidates' responsibility to ensure that their calculator operates satisfactorily, and candidates must record the name and type of the calculator used on the front page of the examination script.

Calculator Brand:_____ Calculator Model:_____

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, if correct this will be considered a valid answer.

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

Unless indicated otherwise, there is exactly one valid answer per question. Only 100% valid answers allow you to obtain the assigned point for each question, all other answers will be scored 0.

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 querying for a 16 byte record, exactly 16 bytes of data is read from disk

C) Writing to an SSD drive is more costly than reading from an SSD drive

Q1.2 Which of the following assertions about storage devices are true?

A) In order to interact with all hard disks, computers use an explicit API

B) Hard drives need to constantly re-organize the data stored on them (wear-levelling) to prevent failure

C) For flash drives, read and writes are both equally fast since these drives use NAND technology instead of traditional spinning magnetic disks

Q1.3 Which of the following assertions about database 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 take 4 bytes, a slot directory that is 512 bytes can address 64 records in a page

C) In a page containing fixed-length records with no nullable fields, the size of the bitmap never changes

Q1.4 Which of the following are true about the benefits of using a record header for variable length records? (multiple answers possible)

A) Does not need a delimiter character to separate fields in the records

B) Always matches or beats space cost when compared to fixed-length record format

C) Can access any field without scanning the entire record

D) Has compact representation of null values

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.

Q1.5 In the worst case, how many IOs are required to find a page with free space?

A) 4 I/Os

B) 5 I/Os

C) 54 I/Os

Q1.6 In the worst case, how many IOs are required to write a record to a page with free space (assuming at least one free page exists)?

A) 4 I/Os

B) 6 I/Os

C) 7 I/Os

Q1.7 Which of the following assertions about indexing are true?

A) The maximum fanout of a B+ tree is equal to 2d where d is its order

B) When an insertion into a B+ tree causes a node to split, the height of the tree always increases

C) As the number of keys in a B+ tree increases, IO cost for searches grows logarithmically with the number of keys

D) Bulk-loading for B+ trees is efficient because we only keep the left most branch in memory for insertions and can disregard the rest of the tree.

Q1.8 Which of the following assertions about indexing are true?

A) The leaves of an ISAM are the only part of the structure that can be modified on an insert

B) Assuming we only have one copy of a table, we can only build an Alternative 1 clustered index on a single column of that table

C) A clustered B+ tree will always perform better than an unclustered B+ tree for equality searches

Q1.9 Which of the following assertions about indexing are true?

A) Indexes are used to optimize tables that are written to much more frequently than they are read from

B) For Alternative-2 trees, clustered indexes will not save any IOs over unclustered indexes when you are only doing equality search

C) Alternative-1 indexes will usually be able to store fewer records in each leaf than Alternative-2 indexes

D) With a series of carefully constructed inserts, an adversary can cause a B+ tree to be unbalanced, so that some lookups visit more nodes than others

Q1.10 Which of the following assertions about buffer management are true?

A) The buffer manager decides when to set the dirty bit of a page

B) The buffer manager decides when to pin a page from the buffer

C) The buffer manager decides when to unpin a page from the buffer

Q1.11 Assume page A is loaded into the buffer pool during a specific access pattern and is not evicted at any point during the access pattern. Page A has a pin count of 7 at the end of all accesses. Then A was accessed

A) Exactly 7 times during this access pattern

B) Possibly more than 7 times during this access pattern

C) Possibly less than 7 times during this access pattern

Q1.12 Which of the following assertions are true about out-of-core sorting / hashing?

A) In the out-of-core sorting algorithm, for every pass except the first one, we keep 1 buffer frame reserved as the output buffer

B) In the out-of-core hashing algorithm, for every pass except the first one, we keep 1 buffer frame reserved as the output buffer

C) Suppose the relation we are sorting is made of 100 data pages and we have 5 pages available for sorting. The second pass of our sorting algorithm will write to disk exactly 25 times

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

A) Increasing the number of buffer pages does not affect the number of I/Os performed in Pass 0 of an external sort

B) Double buffering reduces the time it takes to sort records within a single page

C) Increasing the number of buffer pages decreases the number of I/Os done in Pass 0 of an external hash

Q1.14 Which of the following assertions are true about Grace Hash Join?

A) Grace Hash Join will always perform better than Sort Merge Join

B) If one of the joining relations fits in RAM, Naïve Hash Join can outperform Grace Hash Join

C) A two-pass Grace Hash join requires memory equivalent to the square root of the larger input relation

Q1.15 Suppose we do a Page Nested Loop Join on two tables, P and Q. The same amount of pages are required to store P and Q, but Q has three times more records per page than P, on average. Which one of the following options is true?

A) $P \bowtie Q$ does fewer I/Os than $Q \bowtie P$

B) $Q \bowtie P$ does fewer I/Os than $P \bowtie Q$

C) $P \bowtie Q$ does the same number of I/Os as $Q \bowtie P$

Q1.16 Which one of the following options is true about the IO cost of a self-join (i.e., joining a table with itself)? Assume $B >> 3$ (is much larger than 3), where B is the pages of memory available (SNLJ= Simple Nested Loop Join, PNLJ=Page Nested Loop Join, BNLJ=Block Nested Loop Join)

A) $IO(SNLJ) > IO(PNLJ) \geq IO(BNLJ)$

B) $IO(SNLJ) \leq IO(PNLJ) \leq IO(BNLJ)$

C) IO(SNLJ) ≥ IO(PNLJ) ≥ IO(BNLJ)

D) IO(PNLJ) ≤ IO(SNLJ) ≤ IO(BNLJ)

In the following two questions, assume we are joining two tables R and S, and we have B buffers of memory for our join. Do not assume any additional information about the tables that is not explicitly provided.

Q1.17 We are performing an equijoin. [R] < B, [S] > B^3. Which join algorithm will provide the correct answer with the fewest I/Os:

A) Block Nested Loops Join

B) Grace Hash Join

C) Sort-Merge Join

Q1.18 We are performing an equijoin. B < [R] < $(B-1)^2$, and B < [S] < $(B-1)^2$. Both R and S have only 2 distinct values in the join key column. Which join algorithm will provide the correct answer with the fewest I/Os:

A) Block Nested Loops Join

B) Grace Hash Join

C) Sort-Merge Join

Q1.19 Which of the following assertions about the System R / Selinger algorithm are true?

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

B) It only explores left deep plans

C) It does not keep track of interesting orders as they do not reduce I/O cost

Q1.20 Which of the following assertions about the System R / Selinger algorithm are true?

A) It will produce an optimal query plan, given a perfect cost estimator

B) The running time of the algorithm is at least exponential in the number of tables

C) If we only consider BNLJ and do not consider interesting orders while joining N tables, the number of left-deep query plans is O (N!)

Q1.21 Which of the following assertions are true about transaction management in RDBMSs?

A) If only the last two operations out of many in a committed transaction do not take effect, it is a violation of the Atomicity principle

B) Interleaving transactions is always faster than doing them one at a time

C) All conflict-serializable are also serial schedules

Q1.22 Which of the following properties are true about the strict 2PL protocol?

A) strict 2PL ensures that we do not have cascading aborts

B) strict 2PL ensures that we do not have deadlocks

C) strict 2PL requires that transactions acquire locks only at the start of the transaction, and release all locks only at the end of the transaction

Q1.23 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) All 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

Q1.24 Which of the following assertions are true about partitioning and query evaluation in PDBMSs?

A) If we partition data using the round robin scheme and then perform some operations on the data, every machine is guaranteed to have the same number of records +/- one record

B) Range partitioning keeps data stored in sorted order on each machine

C) A join over two tables that requires an asymmetric shuffle will incur fewer network I/Os than one that requires a symmetric shuffle.

D) If data is partitioned using round robin, when searching for a record, we have to perform broadcast lookup to only a subset of the machines.

Q1.25 Which of the following are valid reasons to replicate data in NoSQL database ? (multiple answers possible)

A) To increase the odds that some node will be able to respond to any request at any given time

B) To allow multiple nodes to split up the work for a large volume of requests.

C) To reduce latency by allowing clients to fetch data from a nearby server

D) To handle ever-increasing database sizes

E) To recover from failures

Section 2 – Detailed-answer questions – 20 questions

Q2.1 (7 points) Consider the following relation:

```
Dogs (
dogId INTEGER PRIMARY KEY, -- cannot be NULL!
age INTEGER NOT NULL,
name VARCHAR(20) NOT NULL,
color VARCHAR(10) NOT NULL);
```

You may assume that:

INTEGER is 4 bytes long

VARCHAR(n) can be between 0 and n bytes long

(a) (1 point) As the records are variable length, we will need a record header in the record. How big is the record header (explain)? You may assume pointers are 4 bytes long, the record header only contains pointers.

(b) (1 point) Including the record header, what is the smallest possible record size (in bytes) in this schema? Explain.

(c) (1 point) Including the record header, what is the largest possible record size (in bytes) in this schema? Explain.

(d) (2 point) Now let's look at pages. Suppose we are storing these records using a slotted page layout with variable length records. The page footer contains an integer storing the record count and a pointer to free space, as well as a slot directory storing, for each record, a pointer and length. What is the maximum number of records that we can fit on an 8KB page? (Recall that one KB is 1024 bytes.). Explain.

(e) (1 point) Suppose we stored the maximum number of records on a page, and then deleted one record. Now we want to insert another record. Are we guaranteed to be able to do this? Explain why or why not.

(f) (1 point) Now suppose we deleted 3 records. Without reorganizing any of the records on the page, we would like to insert another record. Are we guaranteed to be able to do this? Explain why or why not.

Q2.2 (7 points) For the following questions, consider the following table schema:

```
Teams (
teamId INTEGER PRIMARY KEY,
name CHAR(20) NOT NULL,
```

region CHAR(20) NOT NULL,

num_players INTEGER NOT NULL);

Given that the file contains 16 data pages and counting each read or write of a page as 1 I/O, compute the **worst-case** number of I/Os it would take to complete the following 3 queries.

- Assume the file layout consists of a heap file with a page directory of 2 pages, where each page in the page directory can hold pointers to ten different data pages.
- Assume the buffer is big enough to hold the page directory and all data pages.
- Assume operations are independent; i.e query 2 is run on a copy of the file that was never queried
- Assume that to access each data page, you need to access its corresponding entry in the page directory (e.g., you cannot scan all 16 data pages by following sibling pointers between data pages)

(a) (1 point) SELECT * From Teams WHERE num_players > 20 AND num_players < 40;

(b) (1 point) INSERT INTO Teams Values (404, "Not Found", "Unknown", 0); -- assume we know that key value 404 is not already in the table

(c) (1.5 points) INSERT INTO Teams Values (405, "Not Found", "Unknown", 0); -- assume we do not know beforehand if key value 405 is already in the table or not

(d) (1.5 points) SELECT * From Teams WHERE team_id = 3;

(e) (2 points) Now, disregard the page directory. For the SQL statements (a)(b)(d) above, select the scheme for storing the Teams table that would maximize speed in the average case: A heap file, with each page 2/3 full, or a packed sorted file, where the sorted file is sorted on the primary key. If the two would take the same time, indicate that.

Q2.3 (5 points) For the questions below that asks which pages are in the buffer pool at some point in time, please list the pages in **alphabetical order**. Assume we have 4 empty buffer frames and the following access pattern, in which pages are immediately unpinned: A C D F D C B C A E C D

a) (1 point) Which pages are in the buffer pool at the end if we use MRU cache policy?

b) (1 point) How many cache hits will there be after MRU cache policy?

c) (1 point) Which pages are in the buffer pool at the end if we use Clock cache policy?

d) (1 point) How many cache hits will there be after Clock cache policy?

e) (1 point) How many reference bits are set after Clock cache policy?

Q2.4 (5 points) In what follows, suppose the size of a page is 4KB, and the size of the memory buffer is 320KB. We want to perform external sorting for a dataset. Answer the following questions given this configuration.

a) (1 point) What is the maximum size of the dataset (in KB) that can be sorted in one pass?

b) (2 points) What is the maximum size of the dataset (in KB) that can be sorted in n passes, for n > = 1? (Use n in your solution)

c) (2 points) If the size of the dataset is 800KB, how many disk I/O do we need to sort this dataset?

Q2.5 (6 points) (Hashing) For this question, we consider the following relation:

```
Students (
sid INTEGER PRIMARY KEY,
name VARCHAR(50) NOT NULL,
graduation_year INTEGER NOT NULL,
favorite_number INTEGER NOT NULL);
```

- sid is an integer ranging from 20000000-29999999
- graduation year is one of 2019, 2020, 2021, 2022
- favorite number can be any 32-bit integer

Unless otherwise stated, assume that all hash functions used partition data evenly (perfect hashing).

(a) (2 points) If we have 10 pages of memory, and the students relation is 200 pages large, how many I/Os are required to hash the relation on sid? Include the final write in your answer.

(b) (2 points) If we have B pages of memory, and the students relation is B + 1 pages large, how many hash functions do we need to hash students on sid using the external hashing algorithm?

(c) (2 points) If we have B pages of memory, and the students relation is B^2 pages large, how many hash functions do we need to hash students on sid?

Q2.6 (5 points) Consider the following SQL query that finds all applicants who want to major in CS at HKU, live in Hong Kong, and go to a school ranked better than 10 (i.e., rank <10).

Relation	Cardinality	Number of pages	Primary key
Applicants (id, name, city, sid)	2,000	100	id
Schools (sid, sname, srank)	100	10	sid
Major (id, major)	3,000	200	(id,major)

SELECT A.name

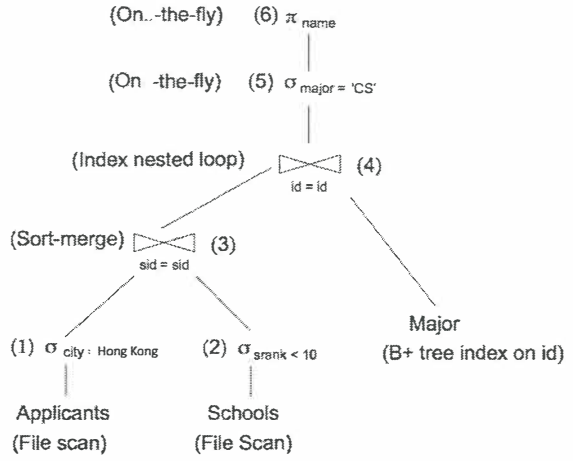
FROM Applicants A, Schools S, Major M

WHERE A.sid = S.sid AND A.id = M.id AND A.city = 'Hong Kong' AND S.srank < 10 AND M.major = 'CS'

Assume that:

- Each school has a unique rank number (srank value) between 1 and 100 (both inclusive)
- There are 20 different cities
- Applicants.sid is a foreign key that references Schools.sid
- Major.id is a foreign key that references Applicants.id
- There is an unclustered, Alternative-2 B+ tree index on Major.id and all index pages (a total of 5) are already in memory
- For Major table, all tuples belonging to same student are stored in one physical page
- The buffer size is 150 pages

The query optimizer is currently considering the below query plan.



(a) (1 point) What is the cardinality of the output of operator 1 in unit of tuples?

Suppose in what follows that the output cardinality of operator 3 is known to be 9 tuples and they have different Applicant ids.

(b) (1 point) What is cost of scanning Applicants?

(c) (1 point) What is cost of scanning Schools?

(d) (1 point) The cost of the index-nested loop join operator?

(e) (1 point) What is the total I/O cost of this whole plan? Do not include the cost of writing the final result.

Q2.7 (6 points) Consider the following relations

Customers (cid INTEGER PRIMARY KEY, name CHAR(30), address CHAR(30));
Orders (oid INTEGER PRIMARY KEY, cid INTEGER REFERENCES Customers(cid), item_name CHAR(20), item_count INTEGER);

In this question, we will consider executing the following query using various join algorithms.

```
SELECT *
FROM Customers C, Orders O
WHERE C.cid = O.cid;
```

Assume that

- Orders has [O] = 100 pages, with $p_O = 50$ tuples per page
- Customers has [C] = 40 pages, with $p_C = 25$ tuples per page
- We have an Alternative-2 B+ tree index on Orders.cid with height h = 2 (our assumption here is that that a tree with only the root has height 0)
- Each customer has at least one order, and Orders.cid is uniformly distributed
- Our buffer has size B = 10, unless noted otherwise
- For index nested loops join, assume the cost model from the lecture (we do not cache index pages)
- Unless otherwise noted, each of the following parts are to be answered independently, i.e. assume the query is executed from scratch for each part
- The files are not already sorted

(a) (2 points) In the best case, what is the I/O cost of a block nested loops join?

(b) (2 points) Assume the index on Orders.cid is clustered. In the best case, what is the I/O cost of an index nested loops join?

(c) (2 points) Assume the index on Orders.cid is unclustered. In the worst case, what is the I/O cost of an index nested loops join?

Q.2.8 (6 points) A relational database has the table:
Employee(PNR text, SALARY integer, GENDER text, DEPT text, PRIMARY KEY(PNR))
The cost-based optimizer uses the following statistics stored as meta-data in the catalogue:

- There are 10000 rows in the table
- There are 20 departments in the company
- GENDER is either 'Male' or 'Female'
- The lowest salary is 20000 and the highest is 100000

The query optimizer makes the following assumptions on statistics:

- The query optimizer assumes even distributions of data values in columns
- The query optimizer assumes independence between values in different columns

For questions of selectivity, you may answer either with a fraction or a decimal

(a) (2 points) What is the selectivity of the condition: SALARY < 25000?

(b) (2 points) What is the selectivity of the condition: (SALARY>100000 OR GENDER !='Male') And DEPT='Toys'?

(c) (2 points) What is the selectivity of the condition: (GENDER = 'Male' OR SALARY > 80000)?

Q2.9 (5 points) Consider the schedule below

T1	R(B)						R(C)			R(A)	
T2		R(B)			R(D)						W(A)
T3			W(B)			R(C)					
T4				W(B)				W(C)	W(D)		

(a) (2 points) Draw the dependency graph for it.

(b) (1 point) Is it conflict-serializable? Explain why.

(c) (2 points) Give a serial schedule to which our schedule above is conflict-equivalent (hint: you can use topological-sort in the dependency graph)

Q2.10 (4 points) Consider the following schedule, with A = 90 and B = 90 at the start.

T1	T2
	read(A) read(B) **if** (A>100) **then** B=A+B
read(B) read(A) **if** (B>100) **then** A=A+B **else** A=A+10 write(A)	
	else B=B+10 write(B)

(a) (2 points) Is this schedule serializable? Explain why.

(b) (2 points) It this schedule conflict-serializable? Explain why.

Q2.11 (7 points) (PDBMS) For this question, assume the following. We have three tables:
Teachers(name, salary, course),
Students(name, SID, course, grade),
Employees(name, eid, company, salary)

[T] = 10 pages, [S] = 10000 pages, [E] = 100000 pages, 1 page = 4 KB. We have 4 machines with 52 buffer pages each. Assume each I/O takes 1 ms. There are 1000ms in a second. All questions are to be considered independently of each other.

(a) (1 point) Assume Students is round-robin partitioned across the 4 machines, and that Teachers lies entirely on Machine 1. What is the minimum network cost (in KB) incurred from performing a join between Teachers and Students?

(b) (1 point) If Students is round-robin partitioned across the 4 machines and Employees is range-partitioned on the name key across 4 machines, what would be the worst-case network cost be (in KB) to perform a sort-merge join between Students and Employees on the name key?

(c) (1.5 points) Let us assume Students and Employees are both hash-partitioned on the name key using the same hash function across the 4 machines and that, due to key skew, machine 1 has 50% of Students on it, with the remaining 50% of the Students table distributed evenly among machines 2, 3, and 4. Assume Employees is distributed uniformly across all 4 machines. If we were to execute a block nested loop join between Students and Employees on each machine, how much time would be taken, in seconds, to complete this? Ignore the time it takes to write the output.

(d) (1.5 points) Assuming Students is range-partitioned across the 4 machines with 40% of Students going to machine 1 and the remaining 60% split evenly across machines 2, 3, and 4. How long, in seconds, would it take to parallel sort the Students table?

(e) (2 points) Now, assume Students is range-partitioned on the name key evenly across the 4 machines. How long, in seconds, would it take to parallel sort the Students table on this key?

Q2.12 (2 points) In a PDBMS, we wish to calculate the square root of the sum of the squared salary of each row in a table Employees(name, eid, company, salary), which is hash partitioned by name, i.e.

```
SELECT SQRT(SUM(salary * salary))
FROM Employees
```

Give a global and local aggregate function pair that would allow us to efficiently compute the aggregate SQRT(SUM(salary * salary)) via hierarchical aggregation. Explain.

Q2.13 (2 points) Suppose you are you are a machine that has just recovered from a crash. You discover only an abort record in your log for transaction T. Under proper 2PC and logging protocols, what can you conclude?

Q2.14 (2 points) Suppose in 2PC with logging, we have one coordinator and three participants. It takes 30ms for a coordinator to send messages to all participants; 5, 10, and 15ms for participant 1, 2, and 3 to send a message to the coordinator respectively; and 10ms for each machine to generate and flush a record. Assume for the same message, each participant receives it from the coordinator at the same time. Under proper 2PC and logging protocols, how long does the whole 2PC process (from the beginning to the coordinator's final log flush) take for a successful commit in the best case?

Q2.15 (1 point) What is the difference between horizontal scaling and horizontal partitioning?

Q2.16 (1 point) (PDBMS) We want to hash partition by name the Dogs table, across many machines. 50% of the Dogs in our database all share the same name. Suppose we run a series of experiments, each time increasing the number of machines over which we hash-partition the Dogs table, measuring the time to complete a parallel scan of Dogs each time. Across experiments, how will the completion time for a parallel scan vary with the number of machines used?

Q2.17 (1 point) What is the difference between a transformation and an action in Apache Spark ?

Q2.18 (1 point) Explain the notion of lazy evaluation in Apache Spark.

Q2.19 (1 point) (Cassandra) We consider a Cassandra NoSQL system with a replication factor F= 3 (hence, 3 copies of the same data). Let us denote by W the number of ACKs required for writing, by R the number of ACKs required for reading. Describe briefly the characteristics of the following two configurations: (a) R=1, W=3 (b) R=1,W=1

Q2.20 (1 point) Over R, W, and F, what is the formula that ensures strong consistency (as opposed to eventual consistency) in a NoSQL system with replication? Explain briefly.

END OF PAPER