Problem 1 (10)

State if the following statements are **TRUE** or **FALSE**. If a statement is false, please explain why. 2 points for correct answer, -1 point for wrong answer.

1. Any schedule produced by a lock scheduler using shared and exclusive locks is conflict serializable.
2. The LRU algorithm should not be used for the buffer management of B+-tree indexes.
3. A B+-tree is typically more efficient than extensible hashing for all types of queries.
4. Sequence pointers are used in leaf nodes of B+ trees to speed up range queries.
5. A hard disk with 5 double-sided platters, 500 tracks per surface, 1000 sectors per track, where each sector is 2048 bytes can store an 1 TBytes file.

**Answer:**

Problem 2 (25)

Consider the following database that stores information about the current NBA season:

**Player** (playerID: integer, name : varchar(50), position : varchar(10), height : integer, weight : integer, teamName: varchar(30))

Each Player is assigned a unique playerID. The position of a player can either be Guard, Center or Forward. The height of a player is in inches while the weight is in pounds. Each player plays for only one team. The teamName field is a foreign key to Team.

**Team** (name: varchar(30), city : varchar(20))

Each Team has a unique name associated with it. There can be multiple teams from the same city.

**Game** (gameID: integer, homeTeam: varchar(30), awayTeam : varchar(30), homeScore : integer, awayScore : integer)

Each Game has a unique gameID. The fields homeTeam and awayTeam are foreign keys to Team. Two teams may play each other multiple times each season. There is an integrity check to ensure homeTeam and awayTeam are different.

**GameStats** (playerID : integer, gameID: integer, points : integer, assists : integer, rebounds : integer)

GameStats records the performance statistics of a player within a game. A player may not play in every game, in which case it will not have its statistics recorded for that game. gameID is a foreign key to Game. playerID is a foreign key to Player.

Write the SQL statements for the following queries:

1. List the name and the position of the players that play in a team in Los Angeles. [5 points]

2. For each city that has at least one NBA team, list the city name and the height of the tallest player playing the "Center" position in that city. [10 points]

3. For each win of the "Celtics", list the game ID and the name(s) of player(s) who scored the maximum number of points for the Celtics in that win, along with that maximum number of points. [10 points]

You can use sub-queries and/or intermediate relations.

Answer:
1) SELECT P.name, P.position
   FROM Team T, Player P
   WHERE T.name = P.teamName and T.city = “Los Angeles”

2) SELECT T.city, MAX(P.height)
   FROM Team T, Player P
   WHERE T.name = P.teamName AND P.position = “Center”
   GROUP BY T.city

3) SELECT G.gameID
   FROM Game G
   WHERE (G.homeTeam="Celtics" AND G.homeScore > G.awayScore) OR
   (G.awayTeam="Celtics" AND G.homeScore < G.awayScore)
   INTO T1

   SELECT GS.gameID, MAX(GS.points) as MaxPoints
   FROM T1, GameStats GS, Player P
   WHERE T1.gameID = GS.gameID AND GS.playerID = P.playerID AND
   P.teamName="Celtics”
   GROUP BY GS.gameID
   INTO T2

   SELECT GS.gameID, P.name, GS.points
   FROM T2, GameStats GS, Player P
   WHERE T2.gameID = GS.gameID AND T2.MaxPoints = GS.points AND
   GS.playerID = P.playerID AND P.teamName="Celtics”

Problem 3 (20)

In the following schedules, Ri(A) stands for a Read(A) operation by transaction i; Wi(A) stands for a Write(A) operation by transaction i.

1. For each of the following schedules show if it is conflict-serializable and give a conflict-equivalent serial schedule. Show all the conflict operations.

   1. R1(A) W1(A) R2(A) R2(B) W3(B) W2(C) R4(A) R4(B) R4(C) R2(D) R3(E)
   2. R1(A) R4(A) W1(A) W3(B) R2(A) R2(B) W2(C) R4(B) R4(C) R2(D) R3(E)
2. Consider the following schedule example where an item is missing:

\[ W3(B) \ R2(A) \ W1(A) \ R3(?) \ R2(B) \ W1(B) \]

and let:

I. **Producible using 2 Phase Locking**

II. **Conflict Serializable**

Choose one answer for the following cases:

\begin{itemize}
  \item[a)] If \( ? = A \), this schedule is which of the following:
    \begin{itemize}
      \item[a.1] I & II
      \item[b.1] I only
      \item[c.1] II only
      \item[d.1] neither I nor II
    \end{itemize}
  \item[b)] If \( ? = B \), this schedule is which of the following:
    \begin{itemize}
      \item[a.2] I & II
      \item[b.2] I only
      \item[c.2] II only
      \item[d.2] neither I nor II
    \end{itemize}
  \item[c)] If \( ? = C \), this schedule is which of the following:
    \begin{itemize}
      \item[a.3] I & II
      \item[b.3] I only
      \item[c.3] II only
      \item[d.3] neither I nor II
    \end{itemize}
\end{itemize}

**Answer:**

1.

1.1 You create the precedence graph and since there is no cycle this is a conflict serializable schedule. The equivalent serial schedule is: T1, T2, T3, T4

1.2 You create the precedence graph and there is a cycle between T1, T2, T4. Not conflict serializable.

2.

\begin{itemize}
  \item[a)] d (it is not conflict serializable so not produced by 2PL)
  \item[b)] c (it is conflict serializable but no produced by 2PL)
  \item[c)] a (it is conflict serializable and also there a sequence of lock and unlock operations that can produce this schedule under 2PL. Notice that a transaction may request a lock anytime before it is needed.
\end{itemize}

**Problem 4 (20)**

Consider the following schema:

Product (id(4), name(16), manufacturer(20), category(10), color(10), webpage(40))
Sales (pid(4), quantity(4), shippingaddress(20), date(12), shippingmethod(10))

Each attribute has a fixed length, with the size (in bytes) indicated by the number in
parentheses. The number of tuples in each relation is: \( T(\text{Product}) = 1,000,000 \) and \( T(\text{Sales}) = 2,000,000 \).

The size of one disk block is 1000 bytes, and there are 101 buffer blocks available in main memory.

1. Compute the number of blocks taken by each table \( B(\text{Product}) \) and \( B(\text{Sales}) \).

2. Consider the logical plan below:

\[
\begin{align*}
&\Pi_{\text{name, date}} \\
&\sigma_{\text{category=’Toy’ AND quantity>100}} \\
&\bowtie_{\text{id=pid}} \\
&\text{Product} \quad \text{Sales}
\end{align*}
\]

1. Assume the following execution plan: the join is implemented as a hash join, its result is pipelined into the selection operator, and, from there, pipelined into the projection operators. Compute the total cost of this physical plan.

2. Derive a new logical plan by pushing selections and projections down as far as possible (you have to draw a plan).

3. Consider a physical plan for your new logical plan in which all selections and projects are pipelined and the join is a block-nested loop join. Further assume that 1% of all Products are in category “Toy” and that 20% of all Sales have a quantity over 100. Compute the cost of your plan.
1. Every record for Product is 100 bytes and for Sales 50 bytes. So, we can store 10 records from Products and 20 records from Sales in each page. Thus, total number of pages from product is 100K and for Sales 100K.

2. First, we need to check if the hash join can be done in one step. Since the size of the buffer is B=101 pages and the size of each relation is 100K pages, we can partition each relation into 100 partitions (B-1) of size 1000 pages. However, this partition size is larger than the size of the buffer. So, we need to repartition every partition again. Now, we will create partitions of size 10 and this is smaller than the buffer. Therefore, now we can do the join on the smaller partitions.

The total cost is: 2*2* (100K+100K) + 100K +100K = 1M I/Os

X We have to push the selections down and we can do the projections on id and name from the Products and on the pid and date for Sales. Then, we store the intermediate results in temporary files (T1 and T2) and after that we apply the block nested loop join.

Here is the new plan:

Assuming that we keep only the id and the name of each tuple in Product that have category =”Toy”, we use 20 bytes for each record and we can store 50 in each page. Also, if we keep pid and data for Sales with quantity>100, we need to use 16 bytes and we can put 62 records in each page. Since, we know the selectivity of
each selection operation, we can estimate the size of the intermediate results. For T1 we have, 1M * 1% = 10000 records => 200 pages and for T2 we have 2M * 20% = 400K records => 6452 pages.
Then, we use block nested loop join and we get: ceiling(200/99) * 6452 + 200 = 3*6452+200 = 19552 I/Os to do the join. So, total I/Os for the plan is:

100K + 100K + 200+6452 + 19552 = 226204 I/Os

Problem 5 (10 points)

Consider the following log (next page) corresponding to a particular schedule at the time of a system crash for four transactions T1, T2, T3, and T4. Suppose that we use the immediate modification protocol with checkpointing. Describe the recovery process from the system crash. Show all the Undo and Redo operations and the values of the data items A, B, C, and D after the recovery.

10 <T1 start>
20 <T1, D, 20, 25>
30 < T1 commit>
40 <checkpoint {}>
50 <T2 start>
60 <T2, B, 12, 18>
70 <T4 start>
80 <T4, D, 25, 15>
90 <T3 start>
100 <T3, C, 30, 40>
110 <T4, A, 30, 20>
120 <T4 commit>
130 <T2, D, 15, 25>
Crash!

Answer:

This question was based on a recovery protocol that I discussed last year but not this year.

However, I assume that we use the same log above for ARIES. Assume that when we do the checkpoint, the DPT table contains D, 20 and the TT table is empty. I assume that End comes after commit.

First, we apply the Analysis phase. In the analysis phase we put the new transactions in TT and we remove the committed ones. Also, for the DPT, we add all the new affected pages. At the end of the analysis phase, TT will contain (T2, 130) and (T3, 100) and the DPT will
contain (D, 20), (B,60), (C, 100) and (A, 110).

In the Redo phase, we start from record 20 and we redo everything. At the end of this phase the values for A,B,C, and D are 20, 18, 40, 25 (note that the first value in the log record is the old value and the next is the new value).

We need to Undo T2 and T3 and ToUndo={130, 100}. We start fro 130, we undo this and we write a CLR record. Now D=15. The new ToUndo={100, 60}. Next, we undo 100, we write a CLR records, C=30, and we write a T3 End. Now, ToUndo={60}. Finally, we undo 60, we write a CLR record, B=12 and we write a T2 End.

The final values for A,B,C,D are: 20, 12, 30, 15

**Problem 6 (15 points)**

Answer the following questions about Extendible Hash indexes. Assume that each bucket can store up to 4 entries (bucket capacity is 4.)

(1) Insert the following keys into an initially empty Extendible Hashing index:

3; 5; 7; 9; 10; 15; 25; 31; 44; 64

(2) Show the index after inserting a single key whose insertion causes a bucket split into the index that you created in (1).

(3) What is the maximum number of data entries that can be inserted into the index that you created in (1) before you have to split a bucket? Explain briefly.

(4) What is the minimum number of record insertions into the index that you created in (1) that will cause a split of all buckets? Explain briefly.

**Answer:**

Next page:
We can insert a value that goes to the bucket 11. Then, this bucket will have to split and the directory will be doubled since local and global depth are the same. We can insert 11.

(2) We can insert a value that goes to the bucket 11. Then, this bucket will have to split and the directory will be doubled since local and global depth are the same. We can insert 11.
(3) The number is 3. With 2 new insertions that two first buckets get full and one more will cause a split.

(4) 5