EECS 339: Introduction to Database Systems

Final Exam
March 18, 2015

Name:

I Short Answers

1. **(6 points)**: Consider the distributed data management systems below:

   (a) Main memory (NewSQL)
   (b) Dynamo (NoSQL)
   (c) Conventional, disk-based database.

   For the following use cases, select the system that is best for the job.

   1.1. An online game where people earn points by tending to a virtual farm with cows and goats.

   A **B** **C**

   1.2. A bank's database with many transfers between accounts.

   A **B** **C**

   1.3. An order entry system for a massive widget store. The seller needs to maintain precise figures for his revenue streams and stock levels in many warehouses.

   A **B** **C**

   1.4. An online advertising firm specializing in personalized matching between a user's browsing history and their marketing.

   A **B** **C**

   1.5. An online auction site where users rapidly bid on one or more items. For fairness, the auctioneer requires serializability on all bidding.

   A **B** **C**

   1.6. A social media site for microblogging and sharing links with friends.

   A **B** **C**

I.1 Database Design

2. **(6 points):** Consider the following schema for a delivery company:

```
package (p_id, start_address, end_address, shipping_priority,
                 sent_time, delivered_time);
driver (d_id, name, address, gender, dob, hire_date);
delivered_by (d_id, p_id);
```

It has the workload:

(a)
```
SELECT name, hire_date
FROM driver
WHERE dob > DATETIME('April 12, 1975');
AND datediff(NOW() - hire_date, year) > 10;
```

(b)
```
SELECT d_id, name, p_id
FROM driver, package, delivered_by
WHERE datediff(delivered_time - sent_time, 'day') > 6
AND p_id.priority = 'high'
AND delivered_by.d_id = driver.d_id AND package.p_id = delivered_by.p_id;
```

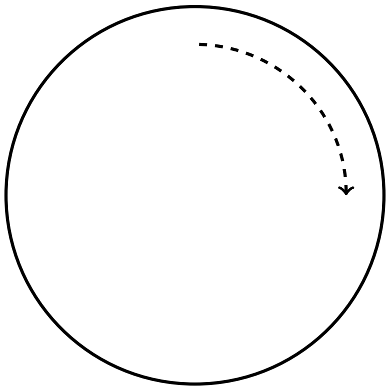
2.1. What are the indexable columns for the first query?

2.2. Given unlimited space, how would you index the data for the second query?

2.3. How would you create a materialized view for the second query?

I.2 Consistent Hashing

3. **(4 points)**: Construct a consistent hash table for your DynamoDB deployment. You use the hash function $i \, mod \, 1024$.



Place the following nodes in the hash table:

- Node A: key 37506
- Node B: key 80879
- Node C: key 33191
- Node D: key 23067

Be sure to label each of them with their hashed position on the circle.

4. **(4 points)**: Name the node to which each of the following keys is assigned on your table:

- Key 238:
- Key 269:
- Key 946:
- Key 33:
- Key 332:
- Key 74:
- Key 108:

5. **(4 points:)** You may notice a marked imbalance in how much data is assigned to each node. How does DynamoDB address this issue?

I.3 CAP Theorem

6. **(8 Points:)** Recall that the CAP theorem states that a distributed system cannot simultaneously provide the following guarantees:

   - **C**onsistency–all nodes see the same data at the same time
   - **A**vailability–a guarantee that every request receives a response about whether it succeeded or failed
   - **P**artition tolerance–the system continues to operate despite arbitrary message loss or failure of part of the system

   For the following scenarios, state which parts of CAP each implements.

   6.1. A column store using two-phase locking wherein if a query waits for a lock for too long it times out.

   6.2. Dynamo's eventual consistency model.

   6.3. A storage engine with weak consistency but many replicas.

   6.4. A shared-nothing database that maintains ACID properties using optimistic concurrency control.

II Concurrency Control

7. **(8 points:)** For the following transaction schedules, determine whether they are conflict serializable by swapping non-conflicting operations. If they are serializable, specify their order and the swaps needed to get that schedule. If not, identify two pairs of operations that are not swappable making it unserializable.

   7.1. Schedule:

Transaction 1	Transaction 2
1. RA	
2. WA	
	3. RB
	4. RA
	5. **commit**
6. WB	
7. WA	
8. **commit**	

   Serializable? **Yes** **No**

   Explanation:

7.2. Schedule:

Transaction 1	Transaction 2
1. RA	
	2. RB
3. WA	
	4. WB
5. RB	
	6. RA
7. WB	
8. **commit**	
	9. WA
	10. **commit**

Serializable? **Yes** **No**

Explanation:

7.3. Schedule:

Transaction 1	Transaction 2
1. RA	
2. WA	
	3. RB
	4. RC
	5. WA
6. RB	
7. RC	
8. **commit**	
	9. WC
	10. **commit**

Serializable? **Yes** **No**

Explanation:

7.4. Schedule:

Transaction 1	Transaction 2
	1. RA
	2. WA
3. RB	
4. WB	
	5. RC
	6. WC
	7. **commit**
8. RD	
9. WD	
10. **commit**	

Serializable? **Yes** **No**

Explanation:

8. **(9 points):** Consider the following queries executing under optimistic concurrency control using two-pass validation. Determine whether one or both transactions succeed. If a transaction aborts, state whether it stopped when the failed query was holding the lock or not. Presume that no additional queries overlap with the ones described here.

   8.1. T_1 receives its transaction id before T_2 begins validation. They have the following read and write sets:
   T_1: read set: B C, write set: A
   T_2: read set: A, write set: B, C

   8.2. T_1 enters its validation phase before T_2, but does not have a transaction id before T_2 begins validation. They have the following read and write sets:
   T_1: read set A, B, write set: C
   T_2: read set B, C, write set: A

   8.3. T_1 receives its transaction id before T_2 begins validation. They have the following read and write sets:
   T_1: read set: A, B C, write set: D
   T_2: read set: A, write set: B, C

III Phantom Tuples

For the following transactions, state whether they are subject to the phantom problem and justify your answer.

9. **(2 points):**

Transaction 1	Transaction 2
`SELECT * FROM employees;`	
	`INSERT into employees VALUES ('JR', 'Art Dept');`
`SELECT count(*) FROM employees;`	

Phantom possible: **YES** **NO**

Why?

10. **(3 points):**

Transaction 1	Transaction 2

`SELECT room_number`
`FROM classrooms, buildings`
`WHERE building.b_name = 'Tech'`
`AND building.b_id = classrooms.b_id;`	

	`INSERT into classrooms VALUES (1, 'Tech', 414);`

`SELECT max(room_number)`
`FROM classrooms, buildings`
`WHERE building.b_name = 'Tech'`
`AND building.b_id = classrooms.b_id;`	

Phantom possible: **YES** **NO**

Why?

11. **(3 points):**

Transaction 1	Transaction 2

`SELECT borrower, sum(amt_owed)` as debt
`FROM car_loans`
`GROUP BY borrower`
`HAVING debt > 10,000;`	

	`DELETE FROM car_loans` `WHERE amt_owed = 0;`
`SELECT distinct borrower` `FROM car_loans;`	

Phantom possible: **YES** **NO**

Why?

12. **(3 points):**

Transaction 1	Transaction 2

`SELECT min(sign_out_date)`
`FROM library_books`
`WHERE status = 'borrowed';`	

	`UPDATE library_books`
	`SET status = 'missing'`
	`WHERE sign_out_date < DATETIME(NOW() - 3 months);`

`SELECT distinct borrower`
`FROM library_books`
`WHERE status='missing';`	

Phantom possible: **YES** **NO**

Why?

III.1 Hierarchical Locking

For the following transactions, specify the locks needed to complete them. Consider both intention and conventional locks.

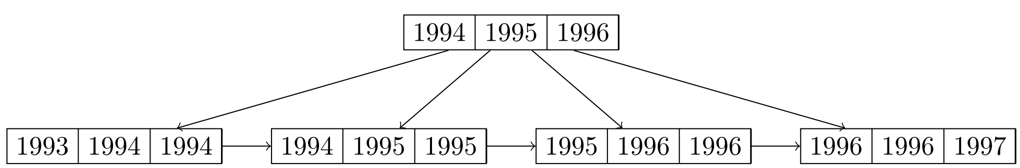
13. **(4 points):**

```
SELECT count(*)
FROM students;
```

14. **(4 points):**

```
SELECT avg(age)
FROM students
WHERE birth_year >= 1995;
```

If the students table has the following B+ Tree index on birth_year, label the tree pages that would be locked to prevent phantoms. Consider only page-level locking, not record-level. Be sure to note each lock type.



15. **(4 points):**

```
UPDATE students
SET status='enrolled'
WHERE class_year=2019;
```

Presume that students has a B+ Tree index on class_year. Describe the locks needed on the index and database, including record-level locking.

IV Recovery

Consider the following log for a database implementing ARIES recovery:

LSN	TID	Type	Object
Checkpoint 1			
1	T1	SOT	
2	T2	SOT	
3	T2	UP	A
4	T1	UP	B
5	T3	SOT	
6	T3	UP	A
Checkpoint 2			
7	T1	UP	C
8	T3	UP	B
9	T2	EOT	
10	T1	UP	A
11	T1	EOT	
12	T3	UP	C

Two checkpoints are taken at the indicated times. At the time of Checkpoint 1, the dirty page table and the transaction table are both empty. The system crashes immediately following when LSN 12 was written to disk.

16. **(4 points):** Draw a timeline for this log.

17. **(4 points):** Presuming there are no flushes, draw a transaction table and a dirty page table for this log after the ARIES analysis phase.

TID	LastLSN

PageId	RecLSN

18. **(4 points):** If Checkpoint 2 has the dirty page table below, was there a flush or not? If so, what is the earliest point the flush could have happened?

PageId	RecLSN
A	6

19. **(4 points):** Fill in the dirty page table just before the crash reflecting the dirty page table above:

PageId	RecLSN

V Distributed Query Processing

For the following queries, draw a distributed query execution plan over three nodes. Be sure to insert the split and merge operators to properly align the data for comparison and for streaming the output to the user.

20. **(4 points):**

```
SELECT *
FROM A, B
WHERE A.v = B.v;
```

Presume that A and B are both partitioned on v.

21. **(4 points):**

```
SELECT count(*)
FROM students;
```

22. **(4 points):**

```
SELECT avg(salary)
FROM employees
WHERE dept='Human Resources';
```