Midterm 2 Review Problems

Note: These problems are *not* comprehensive. Make sure to do a thorough review of all of the relevant material.

1. Imagine you are given the task of creating a database for a hospital. Among other things, you need to keep track of information about patients, doctors, and the relationships between them. If you decide to use an XML database, describe two different ways that you could capture the relationships between patients and doctors in the database.

Questions 2-5

Recall the XML version of our movie database. The root element of the database is called `imdb`, and that root element has three child elements:

- one called `movies`, which has a nested `movie` element for each movie in the database
- one called `people`, which has a nested `person` element for each person in the database
- one called `oscars`, which has a nested `oscar` element for each Oscar award in the database.

Here is an example of one of the `movie` elements:

```
<movie id="M0120338" directors="P0000116"
       actors="P0000138 P0000701 P0000708 P0000870 P0000200"
       oscars="O19980000000 O19980000116">
  <name>Titanic</name>
  <year>1997</year>
  <rating>PG-13</rating>
  <runtime>194</runtime>
  <genre>DR</genre>
  <earnings_rank>9</earnings_rank>
</movie>
```

2. The `actors` attribute above is an example of what type of attribute? How is this type of attribute similar to a foreign key? How is it different?

3. Write an XPath expression that obtains the names of all movies with an R rating from the 1990s. (There are many possible answers here. See if you can come up with at least two!)

4. Write a FLWOR expression that solves the same problem as Question 3. Order the results alphabetically by name.

5. Write a FLWOR expression that finds the average runtime of all movies with an R rating from the 1990s. Your query should return a single element of type `avg_rating`.

Questions 6-9
Here is an example of one of the `person` elements from our XML movie database:

```
<person id="P0000243" directed="M2671706"
    actedIn="M0097441 M0107818 M0139654 M2671706"
    oscars="O20020000243 O19900000243">
  <name>Denzel Washington</name>
  <dob>1954-12-28</dob>
  <pob>Mount Vernon, New York, USA</pob>
</person>
```

6. Recall that a person element only includes the `directed` attribute if that person is a director – i.e., if they have directed one or more of the movies in the database. Write a query that returns the `name` and `pob` elements of all directors who were born in New York state (i.e., whose `pob` value ends with "New York, USA".

7. Write a query that produces, for every movie directed by a person born in New York state, a pair of elements for the name of that movie and the name of the director. Since both of the relevant elements are called `name`, you should reformat the results so that the element for the movie's name is called `movie` and the element for the director's name is called `director`.

8. Now write a query that produces, for each director born in New York state, a new element of type `ny_director` that has the following nested child elements: the existing `name` element of the director, one called `birthplace` for their place of birth, and one or more elements of type `directed`, each of which has as its value the name of one of the movies that the person directed.

9. Finally, revise your query for Problem 8 so that you only include a director in the results if they have directed more than one of the movies in the database.

Questions 10 and 11
Consider the following relation:

Staff(id CHAR(5) PRIMARY KEY, name VARCHAR(30),
status VARCHAR(10), salary REAL, specialty_type VARCHAR(20))

Assume that we're taking the approach to marshalling from PS 3. The primary-key value (i.e., the value of *id*) is stored in the key portion of the key/value pair, and the rest of the column values are stored in the value portion, which is a record that begins with field offsets. In addition, you should assume that we're using 1-byte characters, 2-byte offsets, and an 8-byte double for the salary.

10. What would the marshalled key and value look like for the following tuple?
`('12345', 'Doogie Howser', 'MD', NULL, 'GP')`

11. What steps would the DBMS have to take to unmarshall (i.e., extract) the value of the status field from an arbitrary marshalled tuple of this relation?

Questions 12-14
Consider the following schedule involving two transactions, T1 and T2.

T1	**T2**
r(M)	
	r(N)
	w(N)
r(N)	
w(M)	
commit	
	commit

12. Explain briefly why this schedule is *not* recoverable.

13. Describe what change or changes you would need to make to turn this schedule into a recoverable schedule.

14. Is the original schedule serializable? Explain briefly why or why not.

15. What does it mean for a schedule to be cascadeless? How can a DBMS guarantee this property if it uses locks for concurrency control? How can a DBMS guarantee this property if it uses a timestamp-based approach?

Questions 16-21
Consider the following potential schedule involving two transactions:

s1; r1(X); s2; r2(X); w1(Y); r2(Y); w2(Y); w2(Z); c2; w1(Z); c1

16. Would this schedule execute to completion if it were attempted on a system that uses rigorous two-phase locking and no update locks? If so, show the full schedule in table form including lock requests, unlock actions, and any times when transactions are made to wait. If not, show the partial schedule and explain why it cannot be completed. Assume that transactions acquire a lock immediately before they first need it and upgrade shared locks as needed.

17. Repeat the previous problem, but change T1's write of Z to be a write of X.

18. Would the original schedule above execute to completion if it were attempted on a system that uses timestamp-based concurrency control *without* commit bits? If so, show the full schedule in table form including columns for the state of the data items. If not, show the partial schedule and explain why it cannot be completed. Assume timestamps of 10 and 20 are assigned to the transactions.

19. Repeat Question 17 for a system that uses timestamp-based concurrency control *with* commit bits.

20. Repeat Question 17 for a system a system that uses *multiversion* timestamp-based concurrency control *without* commit bits.

21. Repeat Question 17, but change T1's write of Z to be a write of X.

Questions 22 and 23
Consider the following schedule involving two transactions, which is being executed using timestamp-based concurrency control.

T1	T2
TS = 10	
r(M)	
	TS = 20
	r(M)
	w(N)
r(N)	
commit	
	commit

22. What would be the response of the system to the request by **T1** to **read N** shown above, assuming all prior actions in the schedule have been handled as usual and that the system is using:
    a. **regular** timestamp-based concurrency control **<u>without</u> commit bits**
    b. **regular** timestamp-based concurrency control **<u>with</u> commit bits**
    c. **multiversion** timestamp-based concurrency control **<u>without</u> commit bits**.

    Note: If the T1 would be made to wait, you can just say that. You do <u>not</u> need to also consider what would happen after the wait comes to an end.

23. Now imagine that we **change** T1's read of N to a **write of N** in the schedule above and leave the rest of the schedule as is.
    What would the response of the system be to T1's write request, assuming all prior actions in the schedule have been handled as usual and that the system is using:
    a. **regular** timestamp-based concurrency control **<u>without</u> commit bits**
    b. **regular** timestamp-based concurrency control **<u>with</u> commit bits**
    c. **multiversion** timestamp-based concurrency control **<u>without</u> commit bits**.

Questions 24-27
Imagine that you are charged with implementing a bank's database, and that the database will be distributed across the bank's 6 branches.

24. You decide to replicate the table that contains account balances across the 6 branches using synchronous replication. How would you explain the decision to use replication to the bank's managers? Next, how would you explain to them why you have decided to use *synchronous* replication rather than asynchronous replication?

25. You are planning to use voting-based replication, and your intern has proposed three possible configurations for the voting:
    a. update 4 copies, read 3 copies
    b. update 2 copies, read 4 copies
    c. update 3 copies, read 5 copies

    Which of these would work if the DBMS is using primary-copy locking?

26. How would your answer to the previous question change if the DBMS used fully distributed locking instead?

27. Which of the six configurations from the previous two questions would you recommend if you know that the bank’s workload includes a large number of updates? There may be more than one possible good choice, and you should discuss the advantages and drawbacks of your chosen configuration.