CSE 530 – Midterm Exam

Name:

Question	Points Possible	Points Earned
1	22	
2	10	
3	18	
4	16	
5	17	
Total	83	

Question 1 – Transactions and Locking

(7 points) What is a serializable schedule? Describe why database management systems care about whether a given schedule is serializable.

A serializable schedule is any schedule that is equivalent to a serial schedule.
It is because that we want to interleave transactions to realize concurrency for improving the database performace, but guarantee the same outcome

(5 points) Describe how log files are used to keep transactions durable.

If a server fails, the databases may be left in a state where some modifications were never written from the buffer cache to the data files, and there may be some modifications from incomplete transactions in the data files. When an instance of SQL Server is started, it runs a recovery of each database. Every modification recorded in the log that may not have been written to the data files is rolled forward. Every incomplete transaction found in the transaction log is then rolled back to make sure the integrity of the database is preserved.

(5 points) Explain why binary locks are considered too restrictive to be used.

Binary locks enforce mutual exclusion, which means one cannot access item while it's locked. Thus, it is too restrictive for read especially because read is not changing data and can be performed simultaneously.

(5 points) Under a conservative two-phase locking scheme, transactions will attempt to acquire all of the locks that they need before the transaction begins. What is the advantage of acquiring locks in this way vs. strict two-phase locking?

This can avoid deadlocks

Question 2 – Data Warehousing

(5 points) Explain why OLAP databases are typically not normalized in the same way that OLTP databases are.

OLAP is applied for larger amount of data while OLTP handles small dataset.
Denormalized database fair well under heavy read-load and when the application is read-intensive, for the reason that:
1. the data is present in the same table so there is no need for any joins, hence the selects are very fast
2. a single table with all the required data allows much more efficient index usage.

(5 points) Give an example of when normalization may be used within a data warehouse.

In dimension models, sometimes we do care about space.
In this case, we make a variation based on star schema with a little bit normalization to get rid of the data redundancy for snowflake schema

Question 3 – Distributed Databases

(7 points) Describe the difference between two phase and three phase commits. Why is it important for distributed databases to use three phase commits?

3PC Break the commit (2nd in 2PC)into two phases:
1. prepare to commit
2. commit
Because of biggest drawback of 2PC is that it is a blocking protocol.
In other words, some problems exist for 2PC like
global transaction manager were to crash for some reason.
This causes performance degradation, especially if participants are
holding locks to shared resources. Another problematic scenario is
when both the coordinator and a participant that has committed crash
together.

(5 points) Assuming that we are not using any replication, which partitioning scheme will take up more spacefor a given table: vertical or horizontal? Why?

Vertical.
It is because that vertical partitioning must contain primary key for each partition,
which take extra space, while primary key is optional for horizontal partitioning

(6 points) When replication is used within a distributed database, the database now has a choice of which data source to use when completing a query. Name two things that a database will use to help determine which replica it will attempt to access.

1. Distinguished Copy: make a distinguished copy for all copies.
Thus, all locking requests are sent to this copy
2. Voting Method: simple majority

1. proximity how far it's from the copy
2. It will be useful to look at larger picture, which means containing more satisfied data

Question 4 – MongoDB

The following is an example of a document that can be found in the "VideoGames" collection of a MongoDB database called "exam2":

{

Title: The Legend of Zelda

Year: 1986

Character: {

Name: Link

Age: 12

}

Ratings: [10, 9, 8, 9]

}

You should assume that there are many other documents in this collection, and that each document has the same format as this one, but with different data.

Complete each of the queries below. You may use MongoDB syntax or the Java syntax from lab 5, whichever you prefer. If you use Java syntax, you may assume that you already have the following objects (no need to recreate them):

```
vgDb = new Database("exam2");

vgCollection = vgDb.getCollection("VideoGames");
```

(4 points): Write a query that will retrieve the document for the game titled "Bubble Bobble":

```
db.VideoGames.find( { "Title" : "Bubble Bobble" } )
```

(6 points): Write a query that will retrieve the titles (and only the titles) for all documents that have a character with the name Link.

```
db.VideoGames.find( { "Character.Name" : ”Link" }, { "Title" : 1 }) )
```

(6 points) Write a query that will retrieve all documents that have at least one rating of 9 or greater.

```
db.VideoGames.find( { Ratings : { $gte: 9} })
```

Question 5 – NoSQL

(6 points) Give a brief description of each of the three parts of the CAP theorem:

Consistency: Every read receives the most recent write or an error
Availability: if you can talk to a node in the cluster, it can read and write data (Every request receives a (non-error) response, without the guarantee that it contains the most recent write)
Partition-tolerance: the cluster can survive communication breakages in the cluster that separate the cluster into multiple partitions unable to communicate with each other

(5 points) What is the practical implication of the CAP theorem as it applies to a distributed database?

The trade off between Consistency and Availability.
For distributed database, we may sacrifice some Consistency for Availability

(6 points) Give two reasons why someone might prefer to use a graph database instead of a relational database.

1. the problem is relationship centric rather than data centric
2. the command line is simple and straightforward, no need to be parsed to a tree, optimized and then executed just as SQL