**CS 5200 Database Management Answers to Sample Final Exam Fall 2010**

1. **Transaction Isolation Level**. (8 points) You wish to determine the current total sales for the stores in a large chain of stores. What isolation level should be used?

The best choice is READ UNCOMMITTED.

Explain the reasons for your choice compared with the other isolation levels.

This is a read-only transaction that requires reading a large number of records to compute a continually changing total sales level. Here are the comparisons with the other isolation levels.

* + SERIALIZABLE. This isolation level would effectively shut down the entire system while it is running which could be a very long time. No chain of stores could afford to stop all of its transactions for such a long period of time. Furthermore, while this isolation level would ensure that the computed sales level is consistent and correct when it is computed, the sales level would immediately change, so consistency and correctness are of no benefit.
  + REPEATABLE READ. This has essentially the same disadvantage as SERIALIZABLE with respect to the effect on the system. It will still need to hold a large number of locks until commit time. Removing phantom protection improves performance slightly, but not enough to make a significant difference.
  + READ COMMITTED. This has much better performance than REPEATABLE READ. However, it still has more of an impact on the system than READ UNCOMMITTED. READ COMMITTED allows phantoms and need not compute a consistent result even when there are no phantoms. So it will only give an approximate result. This is the same as READ UNCOMMITTED, so there is no advantage to the use of READ COMMITTED.

Note that the problem requires that exactly one choice be made for the isolation level. As a general rule, if one gives multiple answers on an exam, then the score is determined by the worst answer given. Thus there is no advantage to giving multiple answers on an exam. Note that this rule applies to any exam, not just to this one.

The arguments were scored based on two criteria:

* + Whether the argument matches the situation being considered. Thus, a generic argument that would apply to any situation will have a much lower score than one that makes use of the situation.
  + Whether the argument makes a case for the choice made. So one could get a full score on the argument part of the problem even if the choice was not correct, and vice versa.

Note that these criteria apply to any exam, not just to this one.

1. **Serializability**. (8 points) Three transactions are running concurrently on a database system. They perform the following actions:

T3:W(Z), T2:R(X), T2:W(Y), T2:Commit, T1:R(Z), T3:W(Y), T3:Commit, T1:W(Y), T1:Commit

Determine whether this schedule belongs to each of these classes: serializable, conflict-serializable, view-serializable and strict. Explain why or why not in each case.

* + Conflict-serializable. There is a WR conflict on Z between T3 → T1. There are WW conflicts on Y for all 3 transactions: T2 → T3 → T1. The resulting graph is acyclic so the schedule is conflict-serializable with serial order {T2, T3, T1}.
  + View-Serializable. It is view-serializable because it is conflict-serializable. However, if one examines just the issue of view-serializability, one finds that there are two serial schedules that are view-equivalent to the original schedule: {T2, T3, T1} and {T3, T2, T1}.
  + Serializable. It is serializable because it is conflict-serializable.
  + Strict. It is not strict because T1 reads Z after T3 writes it and before T3 commits.

1. **Cost of Index Operations**. (12 points) The following table contains author names from 2,000,000 research papers:
2. create table Researcher(title varchar(250), author varchar(250),
3. primary key(title, author), index(author));

The average length of a title is 90 bytes, the average length of an author is 30 bytes. The average overhead for each record is 10 bytes. A disk block is 8,192 bytes. A random I/O takes 12ms, and the time to read one block sequentially is 0.1ms. On average, each research paper has 10 authors. How many disk blocks are required for this table? Show your work.

There are 2,000,000 papers and 10 authors/paper, so there are 20,000,000 records in all. A record requires 130 bytes so 63 will fit in one block. If the blocks are full (heap), then one will need about 300,000 blocks. If the blocks are 2/3 full (B-tree), then one will need about 500,000 blocks.

Consider both the B-tree file structure and the heap structure. The author "Albert Finney" occurs in 700 research papers. Is it faster to use the index to find the titles of these papers or to scan the whole table sequentially? Compute the time required for each of these and compare them.

* + Using the index will require 700 random I/Os which takes 700\*12ms = 8400ms = 8.4 seconds. Other I/Os are also required, but there are only a few of these so they do not affect the result.
  + Scanning the table will require 300,000 or 500,000 sequential I/Os. In the former case, the time is 30,000ms = 30 seconds. In the latter case, the time is 50,000ms = 50 seconds. Either answer is acceptable.

One common mistake for the index computation was to compute the time to get the record pointers from the secondary index and then ignore the time to get the records. However, it is the time to get the records that dominates, so much that one can ignore the time required to use the secondary index.

A common mistake for the scan computation was to assume that the primary index is ordered by author. In fact, the author is the second column in the primary index, so it is necessary to scan the entire table, not just the first part.

1. **Query Optimization**. (12 points) A database has stores and merchandise. Here is the schema:
2. create table Store(
3. name varchar(100) primary key,
4. location varchar(20) not null,
5. index(location));
6. create table Merchandise(
7. id int primary key,
8. description varchar(600) not null,
9. index(description),
10. availableAt varchar(100) references Store(name));

There are 500,000 items of merchandise and 10,000 stores in the database. The average size of a store record is 50 bytes and the average size of a merchandise record is 100 bytes. Both of these include the overhead. All indexes are B-trees. There are 500 locations. You want to find all merchandise items sold in the location "Boston". Consider the following query evaluation plans:

* + Select Boston stores, and then join with Merchandise using an index nested loop join.

This is not possible because there is no index on the availableAt column of Merchandise. Here is the analysis if there were an index:

With 10,000 stores and 500 locations, there are an average of 20 stores/location. Selecting the Boston stores requires 2 random I/Os for obtaining the leaf node of the index, and 20 random I/Os to retrieve the store records. This is a total of 22\*12ms = 264ms or about 300ms. Each of the store records has an average of 50 merchandise items for a total of 1000 merchandise items, each of which requires 2 random I/Os (one for the index and one for the merchandise record). The time for retrieving the merchandise records is 1000\*24ms = 24 seconds. The total time is about 24 seconds.

* + Select Boston stores, and then join with Merchandise using a sort-merge join.

With 10,000 stores and 500 locations, there are an average of 20 stores/location. Selecting the Boston stores requires 2 random I/Os for obtaining the leaf node of the index, and 20 random I/Os to retrieve the store records. This is a total of 22\*12ms = 264ms or about 300ms. The sort-merge join requires scanning the entire Merchandise table. There are 100 bytes in a Merchandise record or about 80 per block. There are 500,000 records or about 6000 blocks. Scanning this table sequentially requires 12ms + 6000\*0.1ms = 612ms or about 600ms. So the total time is about 1 second.

* + Join Merchandise with Store using an index nested loop and then select the merchandise in Boston stores.

There are 500,000 records in the Merchandise table and for each of them one must look up a store record. Each look up operation requires 2 random I/Os so the total time is 500,000 \* 24ms = 12,000,000ms = 12,000 seconds or over 3 hours. Obviously, this is not a reasonable alternative.

Use the random and sequential I/O times specified in the previous problem. Discuss the performance of the 3 plans using approximate estimates. Which one is the best?

The first plan is not possible, but even it was possible it would be slower than the second plan. The second plan is much faster than the third.

1. **Physical Database Design**. (8 points) The following is the database design for a film production company. An actor is a person, but a person need not be an actor:
2. create table Person(id int primary key,
3. name varchar(200) not null, position varchar(200));
4. create table Actor(id int primary key references Person(id),
5. biography varchar(65000));

In the physical database design we consider whether to denormalize the schema with a single table as follows:

create table Person(id int primary key,

name varchar(200) not null, position varchar(200),

isActor boolean not null, biography varchar(65000));

Give at least two advantages of each design relative to the other, i.e., four advantages in all: two in each direction. Be sure to clearly label which one is which.

Advantages of the normalized (original) design.

* + The main advantage is the prevention of update anomalies.
  + There is no overhead required to prevent update anomalies. This could be regarded as being the same as the main advantage, but it is somewhat different so credit was given.
  + Better query performance for queries that use only one of the tables.
  + Better update performance for updates that use only one of the tables.

Advantages of the denormalized design.

* + Better query performance for actor information because no join is necessary.
  + Better update performance for actor information because only one table is updated.
  + Simpler design because there is only a single table.

Characteristics that are not advantages of either design.

* + Space utilization. In fact, the two designs use almost exactly the same amount of space. The normalized design has additional overhead for storing 2 records for every actor instead of just 1 record. However, the denormalized design has the additional boolean attribute for every person.
  + Query performance in general. Neither design is better with respect to general queries. It depends on what kind of query.
  + Update performance in general. Neither design is better with respect to general updates. It depends on what kind of update.
  + Intuitive design. This is a subjective judgement. Some people would regard the denormalized design as more intuitive, while others would find the normalized design more intuitive. Only an objective criterion can clearly distinguish the designs.

1. **Normalization**. (12 points) Normalize the relation ABCDEFG when it satisfies the following functional dependencies:
   * E→G
   * AC→D
   * FG→E
   * AFG→B

Show the list of all candidate keys of the relation.

The core is ACF. This determines D, so the exterior attributes are EG.

* + ACFE determines DGB so it is a key.
  + ACFG determines DEB so it is a key.

Note that the relation ABCDEFG is not 3NF because AC→D and AFG→B determine attributes that are not in either of the candidate keys.

Decompose into BCNF if possible.

Every decomposition must include at least one relation that has the attributes EFG since FG→E is one of the dependencies and it cannot be inferred from the other dependencies. However, the dependency E→G is contained in EFG and it will not be a key since it does not determine F. Therefore, no decomposition can be BCNF.

If not, then decompose into 3NF. Show your work. In your normalization show the attributes in each relation and the primary key of each relation in the decomposition.

Applying the 3NF algorithm gives the following decomposition:

* + EG with primary key E.
  + ACD, with primary key AC.
  + EFG, with primary key FG.
  + ABFG, with primary key AFG.
  + ACEF, with primary key ACEF.

The last relation could also be ACFG with primary key ACFG.