Introduction to Database Design, F2015
IT University of Copenhagen
Carsten Schürmann

Hand-in 4 : Concurrency and OLAP

Theoretical Part

Problem 1: Serializability and 2PL [20 points]

Yes/No Questions [10 points, 2 points each]

1. All serial transactions are both conflict serializable and view serializable.
2. For any schedule, if it is view serializable, then it must be conflict serializable.
3. Under 2PL protocol, there can be schedules that are not serial.
4. Any transaction produced by 2PL must be conflict serializable.
5. Strict 2PL guarantees no deadlock.

Serializability [10 points]

Consider the following schedule. We write $R(\cdot)$ for read and $W(\dot{)}$ for write operations.

Time	T_1	T_2	T_3
1			$R(A)$
2			$W(A)$
3	$R(A)$		
4	$W(A)$		
5		$R(B)$	
6		$W(B)$	
7	$R(C)$		
8	$W(C)$		
9			$R(C)$
10			$W(C)$
11			$R(B)$
12			$W(B)$

1. [1 point] Is this schedule serial?
2. [3 points] Give the dependency graph of this schedule.
3. [1 point] Is this schedule conflict serializable?
4. [3 points] If you answer yes to the previous question, provide the equivalent serial schedule. If you answer no, briefly explain why.

5. [2 points] Could this schedule have been produced by 2PL?

Problem 2: Deadlock Detection and Prevention [30 points]

Deadlock Detection

Consider the following lock requests. We write $S(\cdot)$ and $X(\cdot)$ for requesting a shared, exclusive lock, respectively, and LM stands for the lock manager.

Time	T_1	T_2	T_3	LM
1	$S(D)$			g
2	$S(A)$			
3		$S(A)$		
4		$X(B)$		
5	$X(C)$			
6			$S(C)$	
7	$S(B)$			

1. [6 points] For theses lock requests determine which lock will be granted or blocked by the lock manager. Please write g in the LM row to indicate the lock is granted and b to indicate the lock is blocked. For example, in the table, the first lock ($S(D)$ at time 1) is marked as granted.

2. [4 points] Give the wait-for graph for the lock requests in the table above.

3. [3 points] Determine whether there exists a deadlock in the lock requests in the table above and briefly explain why.

Deadlock Prevention

Consider the following lock requests. In contrast to above, we represent three lock managers LM_1, LM_2, and LM_3 with different lock policies.

Time	T_1	T_2	T_3	LM_1	LM_2	LM_3
1	$S(D)$			g	g	g
2			$X(B)$			
3	$S(A)$					
4		$S(C)$				
5	$X(C)$					
6		$X(B)$				
7			$X(A)$			

1. [6 points] For the lock requests in table above, determine which lock request will be granted, blocked or aborted by the lock manager 1 (LM_1), which has no deadlock prevention policy. Please write g for grant, b for block and a for abort.

2. [5 points] Give the wait-for graph for the lock requests in the table above. Give a one-sentence reason why the lock requests in the table under LM_1 result in a dead-lock.

3. [3 points] To prevent deadlock, we use lock manager 2 (LM_2) that adopts the Wait-Die policy. We assume that in terms of priority: $T_1 > T_2 > T_3$. Determine which lock request will be granted (g), blocked (b) or aborted (a) by LM_2.

4. [3 points] Now we use lock manager 3 (LM_3) that adopts the Wound-Wait policy. Again, we assume that in terms of priority: $T_1 > T_2 > T_3$. Determine which lock request will be granted (g), blocked (b) or aborted (a) by LM_3.

Practical Part

In this hand-in you will be doing analytics on a data set of movie ratings from MovieLens (movielens.umn.edu). It contains about 1 million user ratings of movies (on the scale 1-5), each associated with a unique user ID and the date of the rating. For each user, gender, approximate age, occupation, and zip code is recorded. Finally, information about the genre(s) of each movie is available. The main data is available as three relations:

```
user(id,gender,age,occupation,zip)
rating(userId,movieId,rating,time)
movieGenre(genreId,movieId)
```

These relations use numerical codes for occupation, movie ID, and genres. Textual descriptions of these codes are available as:

```
occupation(id,description)
movie(id,title)
genre(id,name)
```

Finally, a separate relation with information about zip codes is available:

```
zipcode(zip,city,state,lattitude,longitude,timezone,dst)
```

From the course homepage you can download a (ziped) file movielens.sql containing this data, prepared for importing into MySQL. (There is also a bigger file, movielensXL.sql, if you feel like trying some of the tasks with a bigger, but not as detailed, data set.)

Tasks

Your first task is to perform *data cleaning*. In particular, we wish to remove all data that lacks proper foreign key references. For example, there are tuples in user that do not refer to a tuple in zipcode. Another issue is that age approximations can be made more accurate. In the data set, age 1 means Under

18, age 18 means 18- 24, age 25 means 25-34, etc., up to 56 which means 56+. Modify the ages so that they are a best guess for the range; clearly, this is not an exact science.

Second, you should choose a way of *enriching* the data set. For example, you could make a table with the median or average household income for each zip code. This data can be downloaded in XLS format from `http://www.psc.isr.umich.edu/dis/census/Features/tract2zip/index.html` To get the data into MySQL, export as a comma-separated (CSV) file, and use MySQL's

```
LOAD DATA LOCAL INFILE
```

syntax to load it into a table. Alternatively, integrate with information in the IMDB database you worked with previously. (NB! Movie IDs and titles are not identical.)

The third task is to create a relational OLAP model for the data, according to the principles discussed in the class. Since MySQL does not support materialized views, you will need to construct separate tables with pre-aggregated data, and put indexes on these. B-tree indexes will suffice, even though they may be slower than bitmap indexes in this context. Motivate your choice of pre-aggregation with a few usage scenarios, and give corresponding example SQL queries (probably working on the pre-aggregated tables rather than the whole data set).

To be handed in

- Name of all group members who contributed to the hand-in.
- A short description of the data cleaning and enrichment performed, including the MySQL commands used.
- A description of the relational OLAP model created (using the terms fact, measure, dimension), along with the transscript.
- A description of the pre-aggregation tables and indexes created, incl. transscript.
- For each group member, an interesting fact about the ratings data, found using OLAP exploration. E.g., those who liked Bowling for Columbine live in zip codes with average household income that is....

The second hand-in should be handed in by each group no later than

6. December, 2015, 23:55.

It suffices that one group member sends the solution as a single PDF file with a file name that includes the group number. Please upload your reply to learn-it. Late hand-ins will not be considered. This project will be graded and contribute with 15% to your final grade.