

CIS 560 Database System Concepts

Spring 2007

Homework 9 of 10: Problem Set (PS9) / Project Part 3 of 3 ACID Properties and Concurrency

Assigned: Sun 29 Apr 2007
Due: Fri 04 May 2007 (before midnight)

The purpose of this assignment is to exercise your basic understanding of concurrent transactions. In this problem set, you will design a view serializable (and conflict serializable) update transaction and simulate it in MySQL by implementing a server-side session manager.

This homework assignment is worth a total of 20 points..

Use your KSQL drop box to turn in a.zip file PS9-XYZ.zip, with your initials in place of XYZ.

References for this assignment:

Section 13.4.5, MySQL manual: LOCK TABLES and UNLOCK TABLES syntax
<http://dev.mysql.com/doc/refman/5.0/en/lock-tables.html>

Write the following five queries and tie each to a radio button or drop-down menu entry.

1. **(40%) Implementing a concurrent update manager.** Write a JSP servlet that is embedded in the record browser page for your main relationship set (Encountered or Took). This servlet should:
 - a) Initiate a new update transaction each time the “update” button is pressed.
 - b) Issue a transaction ID to the client (browser session) instance, printing this out at the bottom of the HTML page.
 - c) Attempt to acquire all locks needed on the relations for which mutually exclusive access is required (as determined in MP9-2).
 - d) Read the computed attribute **or** update the tuple and computed attribute (as specified in MP9-3).
 - e) Release locks.
 - f) Print a result message: “SUCCESS: [attribute name] updated in [table name]” or “FAILURE: could not get lock on [table name]”

You may propagate this servlet over the other entities as well, but it is not required.
Turn in this code as PS9-1.

2. **(20%) Mutual exclusion and concurrent query.** Here are the queries and computed attributes. Do **one** of these, for your chosen project.

(*AMMDB*) Calculate, in the Encountered table, a Boolean attribute *IsMin* that is true if and only if the dungeon level of the encounter is the shallowest depth at which *that player* has ever encountered *that monster*. You should add this table to the schema so that it will serialize properly.

(*GradMiner*) Calculate, in the Took table, a Boolean attribute *Earliest* that is true if and only if the semester of the course is the earliest one in which *that student* has ever

encountered *that KSU course*. (Note that this transaction is analogous to IsMin above, but is a bit contrived, because it is unlikely that there would be any interferent readers and writers for this particular column *unless* there is concurrent I/O going on to import historical records. i.e., you wouldn't have concurrent transactions for "123-45-6789 took CIS 560 in Spring 2007" and "123-45-6789 took CIS 560 in Fall 2007", unless the database was *very slow*...)

As you can verify for yourself, write access to the computed attribute has to be mutually exclusive. **Explain why.**

Write down the query and in your PS9.pdf file, specify exactly what tables need to be locked, how, and why. (Hint: think about the read, or S, locks and read-write, or X locks that we talked about in class.)

For your own convenience, you might want to change your reader to put NULL values for trailing attributes that are not supplied in sample data (from you or from the instructor).

- 3. (20%) Testing concurrency.** Simulate a concurrent update by making the query sleep for 10 seconds between reads and writes **or** halfway through reads. Make two buttons: "Refresh" (requiring a read lock) and "Update" (requiring a read-write lock). Show that the lock matrix we went over in class is respected by printing the lock requests and a timestamp next to each one.

Note that because MySQL manages all lock queues transparently, you won't get information on blocking and lock release from the DBMS. Instead, let your timestamps illustrate the (approximate) acquire/release order.

- 4. (20%) ACID properties, liveness, and starvation-freedom.** Write a short paragraph documenting how the above lock mechanism does and does not observe the properties of Atomicity, Consistency, Isolation, Durability, Liveness, and Starvation-Freedom. If liveness and starvation freedom are not observed, what would it take to guarantee them?

Project wrap-up. As specified in lab on Tue 24 Apr 2007, your project should exist online at a fixed URL from the time you turn it in through finals week. Send the URL to CIS560TAL@listserv.ksu.edu; if you have any questions about whether the project specification, or any homework specification, has been met, please e-mail or IM the instructor, or come in during office hours.

Please note: You are required to complete this part with PS9 or MP10, no matter which of these you do.

Extra Credit. Change PS9-3 to a servlet that generates multiple read and read-write updates when you press a "Test" button. Do either of the following variants for 20% extra credit.

Dungeon-crawling philosophers. For read/write, take a **particular** Person, a **particular** Monster, and generate a stream of encounters at random depths within ± 10 levels of the monster's regular depth, to a min of 0 (Town) and a max of 100.

Student philosophers. For read/write, take a **random** Student, a **particular** KSU-Course, and generate a stream of course registrations in random semesters in the range Fall, 2003 – Spring, 2007.

You may keep the same data for all the other attributes, as we aren't interested in that. Test your solution by printing out all the update results in a scroll box.

Class participation. Post any unclear points about ACID properties or concurrency to the class web group before the due date of the assignment.

The Road Ahead

- Homework 10 (this assignment) will wrap up your project with aggregation queries.