H2 database and MVCC - 4th week

Sungkeun Kim

Report History

Week 1	Understanding MVCC and test program
Week 2	Issue analysis - Isolation level with MVCC(#174, #216) - Part 1(Pending)
Week 3	Issue analysis - executing UPDATE and SELECT FOR UPDATE simultaneously(Part 1)
Week 4	Issue analysis - executing UPDATE and SELECT FOR UPDATE simultaneously (Part 2) Diagrams

1. Overview of weekly report

Previous week, I chose another issue which is obvious bug. I found open issue related to the MVCC which is the #180. The issue claims that when “UPDATE” and “SELECT FOR UPDATE” queries are executed simultaneously, some rows are lost from the database. I reproduced this issue with a test program and narrowed down the scope of the debugging areas.

In this week, I found out the reason why the expected row is lost from the database. In order to debug it, I reviewed the relative codes and put some logs to the system to trace the bug. Now, I made a diagram with important data structure which is very related with this bug.

Next week, I will find out a patch for this bug. If you have any comments, please let me know.

2. Debugging

2.1 Major cause

Here is an important data structure MVMap which is a shared to other transactions for storing the result of each query. Different transactors will share this MVMap when they access the same table. When the updater updates its rows to MVMap and other transaction like a selector is already under operation with same row, the updater stop adding it and release its lock to let the other transaction finish its job. Sometime later updater tries to add its rows to MVMap again. However, updater does not process all its rows, which means some value of row remains with “null” value. The problem is these nulled rows are removed when updater commits its

operation because this is null. If one of removed rows is used by other transaction such as selector in the production scenario, it cannot be seen.

2.2 Visualization

Because it is difficult to explain the concurrency issue, I created an animated sequence diagram based on the complex debugging logs. First Part explains the normal situation and second part explains the how the bug occurs in detailed.

Before starting bug analysis, let's assume couple of things. Firstly, there is database which has two columns and initialized as follow.

Id(primary key)	last_updated
0	100
1	101
2	102
3	103
4	104
5	105

Secondly, there are two threads. First thread repeatedly updates every time stamp in the table(let's say it Updater) and second thread repeatedly selects a random row for update and check if the row is retrieved in the database correctly(let's say it Selector).