H2 database and MVCC - 1st week
Sungkeun Kim

1. Overview of weekly report

This is the first week of reporting H2 database with MVCC. This week, I mainly focused on getting familiar with using H2 database and conceptually understanding the MVCC. In order to do these, firstly I created a program that demonstrates how the H2 database works when MVCC mode is enabled. Secondly, I downloaded the H2 database source files and compiled it and then executed a test suit for MVCC.

2. MVCC - MultiVersion Concurrency Control

In order to understand the MVCC, I read a wikipedia page of MVCC. According to the wikipedia, MVCC is a commonly used solution to solve the concurrency problem. Compare to locking which is the most simplest method to resolve the concurrency problem, MVCC provides concurrent access to the database. For example, suppose, there are a writer who is trying to update a specific row and a reader who is trying to read the same row. If database uses locking method, the reader should wait until the writer commit the job. In contrast, if database uses MVCC method, the reader do not need to wait. Therefore, locking makes system slow but MVCC doesn’t. This is because when users connect to the database, users see its snapshot of the database. It means that different users see different snapshot of database so each user don’t need to wait for other users. Readers cannot see the latest updated data until the write commits its updated data.

Question: Can the database using MVCC be used for Banking system?

 I don’t think so, because when two person who are trying to withdraw $10 from the same account, its balance is $10, both person can withdraw $10. It could be problem.

3. Exemplary Program for MVCC

For better understanding, I wanted to see the differences between locking and MVCC in person. Therefore, I created a program that executes the same database queries but results are different depends on whether MVCC is enabled or not.

(github: https://github.com/danguria/H2Exercise)

Test Scenario

There are two users and both are trying to use the same database. First user is a writer who updates a specific row every one second and commits it every 10 seconds. The other user is a reader who tries to read the same row every one second.

How to enable/disable MVCC?

To use the MVCC feature in H2 database, append ";MVCC=TRUE" to the database URL.

```
static String DB_URL = "jdbc:h2:tcp://localhost/~/student;MVCC=FALSE";

static String DB_URL = "jdbc:h2:tcp://localhost/~/test;MVCC=TRUE";
```

Code Snippet

Writer

```
// update age every second and commit every 10 updates
for (int i = 1; i < Integer.MAX_VALUE; i++) {

    String sql = "UPDATE Registration " +
                 "SET age =" + i + "WHERE id in (1, 2)";

    H2Exercise.executeSql(conn, sql);

    if (i % 10 == 0) {
       H2Exercise.commit(conn);
    }

    Thread.sleep(1000);
}
```

Reader

```
public void run() {
    try {
       conn = H2Exercise.connectDB(H2Exercise.H2_DRIVER, H2Exercise.DB_URL, USER, PW);
       while (true) {
              H2Exercise.selectRecords(conn);
              Thread.sleep(1000);
       }
    } catch (Exception e) {
       e.printStackTrace();
    }
}


static void selectRecords(Connection conn) throws Exception {
    H2Exercise.log("Selecting records ...");

    Statement stmt = conn.createStatement();
    String sql = "SELECT id, first, last, age FROM Registration";
    ResultSet rs = stmt.executeQuery(sql);

    // Extract data from result set
    while(rs.next()){
        // Retrieve by column name
        int id  = rs.getInt("id");
        int age = rs.getInt("age");
        String first = rs.getString("first");
        String last = rs.getString("last");

        // Display values
        H2Exercise.log("ID: " + id + ", Age: " + age
                     + ", First: " + first + ", Last: " + last);
    }
    H2Exercise.log("Selected records ...");
    rs.close();
}
```

Test Result

“Log 1” and “Log 2” show different behavior when the program is running with different MVCC option. Log 1 shows the reader cannot read the specific row until the writer finish the job and commit it. However, log 2 shows the reader can read the row even if the writer still writing something to the same row.

TID of writer : 9   TID of reader : 11 (blue colored)

```
[9] connecting to h2 ...
[9] Setting auto commit to OFF ...
[9] Setted auto commit to OFF...
[9] Updating dummy records into the REGISTRATION table...
[9] Updated dummy records into the REGISTRATION table...
[9] Inserted dummy records into the REGISTRATION table...
[9] Updated dummy records into the REGISTRATION table...
[11] connecting to h2 ...
[11] successfully connected to h2 ...
[11] Selecting records ...
[9] Updating dummy records into the REGISTRATION table...
[9] Updated dummy records into the REGISTRATION table...
[9] Updating dummy records into the REGISTRATION table...
[9] Updated dummy records into the REGISTRATION table...

org.h2.jdbc.JdbcSQLException: Timeout trying to lock table "REGISTRATION"; SQL statement:
SELECT id, first, last, age FROM Registration [50200-194]
```

<Log 1. Logs when the program is running with disabled MVCC>

TID of writer : 9   TID of reader : 11 (blue colored)

```
[9] connecting to h2 ...
[9] Setting auto commit ...
[9] Setted auto commit ...
[9] Updating dummy records into the REGISTRATION table...
[9] Updated dummy records into the REGISTRATION table...
[11] connecting to h2 ...
[11] successfully connected to h2 ...
[11] Selecting records ...
[11] ID: 1, Age: 18, First: Zara, Last: Ali
[11] ID: 2, Age: 25, First: Mahnaz, Last: Fatma
[11] Selected records ...
[9] Updating dummy records into the REGISTRATION table...
[9] Updated dummy records into the REGISTRATION table...
[11] Selecting records ...
[11] ID: 1, Age: 18, First: Zara, Last: Ali
[11] ID: 2, Age: 25, First: Mahnaz, Last: Fatma
[11] Selected records ...
[9] Updating dummy records into the REGISTRATION table...
[9] Updated dummy records into the REGISTRATION table...

..... (log skipped)
```

```
[9] committing ...
[9] committed ...

[11] Selecting records ...
[11] ID: 1, Age: 10, First: Zara, Last: Ali
[11] ID: 2, Age: 10, First: Mahnaz, Last: Fatma
[11] Selected records ...

[9] Updating dummy records into the REGISTRATION table...
[9] Updated dummy records into the REGISTRATION table...
[11] Selecting records ...
[11] ID: 1, Age: 10, First: Zara, Last: Ali
[11] ID: 2, Age: 10, First: Mahnaz, Last: Fatma
[11] Selected records ...
```

<Log 2. Logs when the program is running with enabled MVCC>

4. H2 database source hierarchy and build system.

So far, I have learned what the MVCC is and how it works on the H2 database. After that, I started to figure out how source files look like and tried to build these files.

Source hierarchy

Currently, the master branch of the github for H2 database is failing to build. Instead of downloading the latest master branch, I downloaded the latest released source files which is 1.4.194. Source files are coming into five categories which have special goal. Table 1 show the source hierarchy and its purpose.

Folder	Purpose
h2/src/docsrc	Create the documentation from sources (incl. API Javadocs)
h2/src/installer	Create the h2.zip file and the Windows installer.
h2/src/main	Core H2 source files

h2/src/test	Everything about testing
h2/src/tools	Create utility tools such as building, testing tools

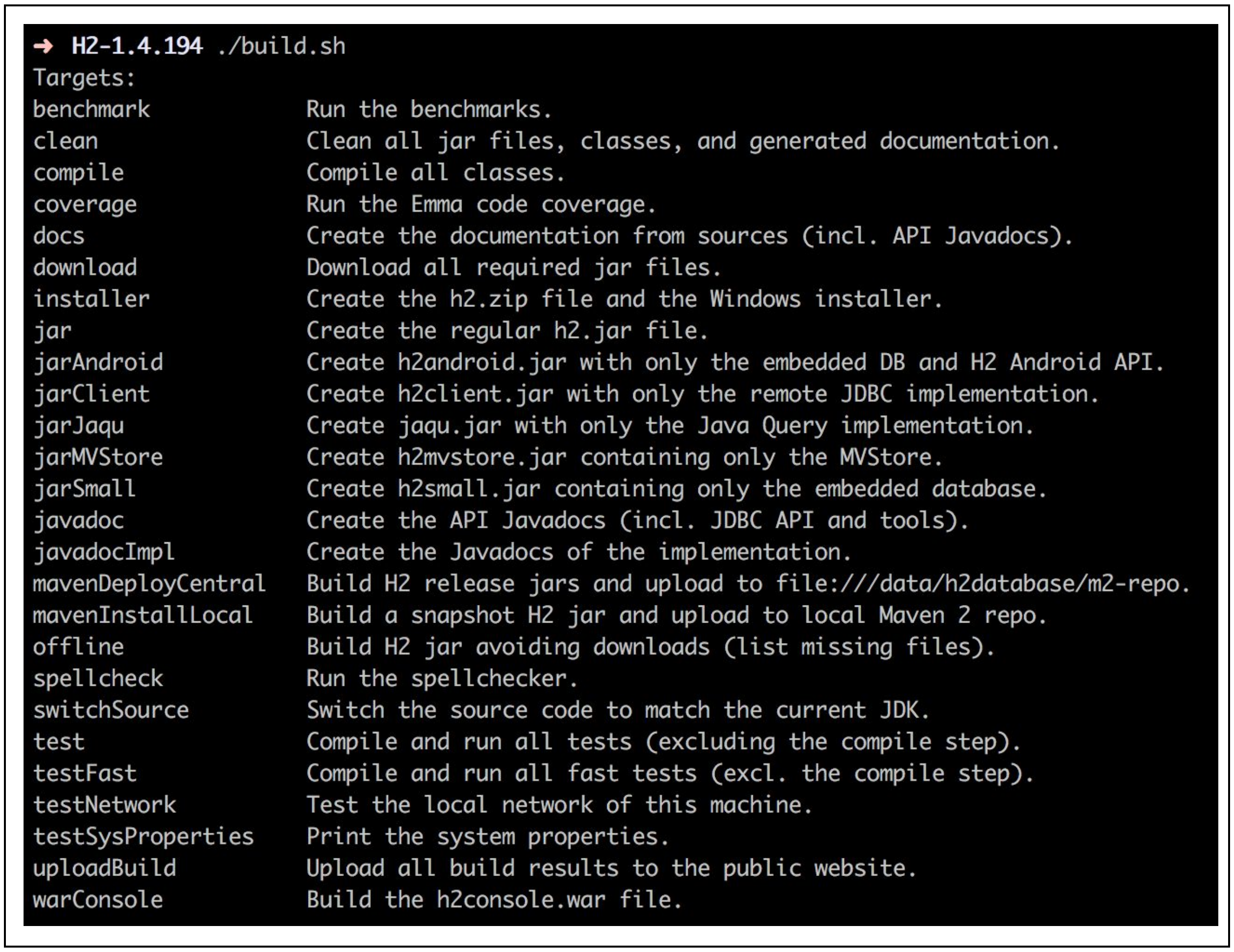
<Table 1. Source hierarchy>

Build system

H2 is build and run through Maven which project management tool. The maven requires using JDK 1.7 or higher but H2 is required to be build with JDK 1.6. In order to build successfully, I should create toolchain which is explained at the H2 documentation. However, I couldn't succeed in building it using Maven because when maven compile H2 source files with java compiler, JDK was not 1.6. More strange thing is that toolchain seems to work well in the logs. I am trying to fix this issue by understanding toolchain mechanism in detail.

```
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ h2 ---
[INFO] Toolchain in compiler-plugin:
JDK[/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/]
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 504 source files to
/Users/sugnkeunkim/Documents/workspace/H2-1.4.194/target/classes
[INFO] -------------------------------------------------------------
[ERROR] COMPILATION ERROR :
[INFO] -------------------------------------------------------------
[ERROR] javac: invalid target release: 1.7
Usage: javac <options> <source files>
use -help for a list of possible options
```

Fortunately, I found that there is another way to compile and build the H2 artifacts. H2 database provides shell script for building it. Figure 3. Shows building options.

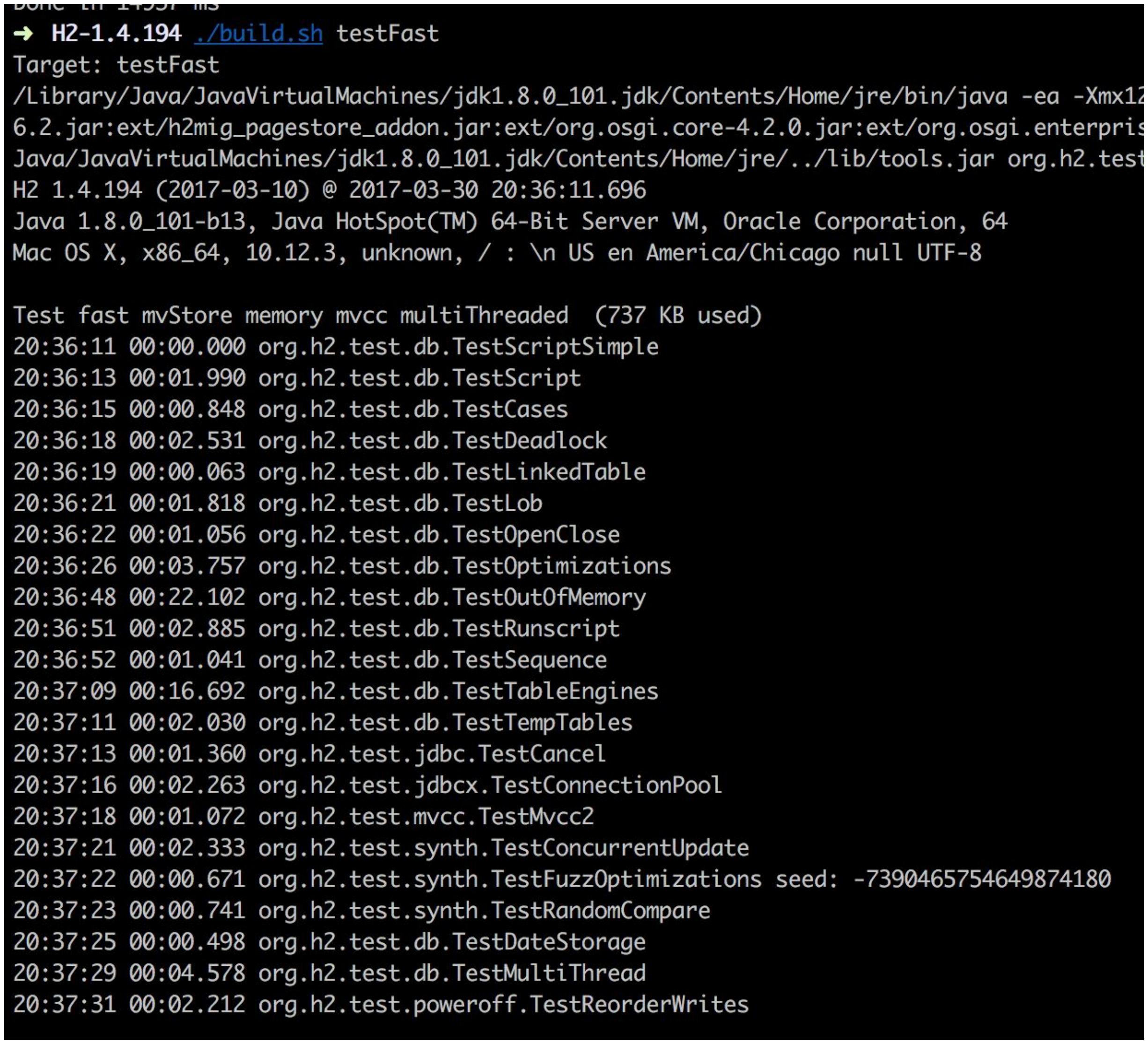


```
➜ H2-1.4.194 ./build.sh
Targets:
benchmark             Run the benchmarks.
clean                 Clean all jar files, classes, and generated documentation.
compile               Compile all classes.
coverage              Run the Emma code coverage.
docs                  Create the documentation from sources (incl. API Javadocs).
download              Download all required jar files.
installer             Create the h2.zip file and the Windows installer.
jar                   Create the regular h2.jar file.
jarAndroid            Create h2android.jar with only the embedded DB and H2 Android API.
jarClient             Create h2client.jar with only the remote JDBC implementation.
jarJaqu               Create jaqu.jar with only the Java Query implementation.
jarMVStore            Create h2mvstore.jar containing only the MVStore.
jarSmall              Create h2small.jar containing only the embedded database.
javadoc               Create the API Javadocs (incl. JDBC API and tools).
javadocImpl           Create the Javadocs of the implementation.
mavenDeployCentral    Build H2 release jars and upload to file:///data/h2database/m2-repo.
mavenInstallLocal     Build a snapshot H2 jar and upload to local Maven 2 repo.
offline               Build H2 jar avoiding downloads (list missing files).
spellcheck            Run the spellchecker.
switchSource          Switch the source code to match the current JDK.
test                  Compile and run all tests (excluding the compile step).
testFast              Compile and run all fast tests (excl. the compile step).
testNetwork           Test the local network of this machine.
testSysProperties     Print the system properties.
uploadBuild           Upload all build results to the public website.
warConsole            Build the h2console.war file.
```

<Figure 3. H2 build options>

Testing for MVCC

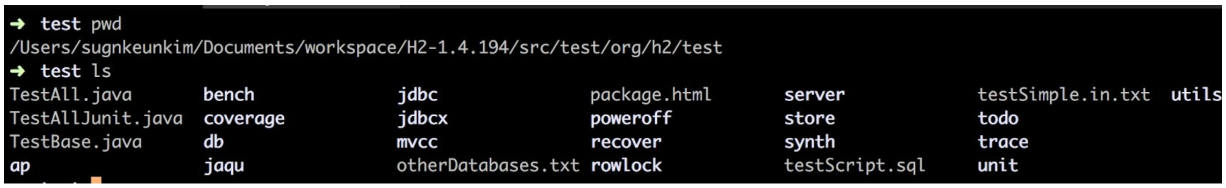
There are many test suits for the H2 database. Mvcc test is one of them. In order to test mvcc, run build script with test option. As I mentioned in Table 1., source files for test are located in “h2/src/test”. There are many other test packages.



```
➜ H2-1.4.194 ./build.sh testFast
Target: testFast
/Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/jre/bin/java -ea -Xmx12
6.2.jar:ext/h2mig_pagestore_addon.jar:ext/org.osgi.core-4.2.0.jar:ext/org.osgi.enterpris
Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/jre/../lib/tools.jar org.h2.test
H2 1.4.194 (2017-03-10) @ 2017-03-30 20:36:11.696
Java 1.8.0_101-b13, Java HotSpot(TM) 64-Bit Server VM, Oracle Corporation, 64
Mac OS X, x86_64, 10.12.3, unknown, / : \n US en America/Chicago null UTF-8

Test fast mvStore memory mvcc multiThreaded  (737 KB used)
20:36:11 00:00.000 org.h2.test.db.TestScriptSimple
20:36:13 00:01.990 org.h2.test.db.TestScript
20:36:15 00:00.848 org.h2.test.db.TestCases
20:36:18 00:02.531 org.h2.test.db.TestDeadlock
20:36:19 00:00.063 org.h2.test.db.TestLinkedTable
20:36:21 00:01.818 org.h2.test.db.TestLob
20:36:22 00:01.056 org.h2.test.db.TestOpenClose
20:36:26 00:03.757 org.h2.test.db.TestOptimizations
20:36:48 00:22.102 org.h2.test.db.TestOutOfMemory
20:36:51 00:02.885 org.h2.test.db.TestRunscript
20:36:52 00:01.041 org.h2.test.db.TestSequence
20:37:09 00:16.692 org.h2.test.db.TestTableEngines
20:37:11 00:02.030 org.h2.test.db.TestTempTables
20:37:13 00:01.360 org.h2.test.jdbc.TestCancel
20:37:16 00:02.263 org.h2.test.jdbcx.TestConnectionPool
20:37:18 00:01.072 org.h2.test.mvcc.TestMvcc2
20:37:21 00:02.333 org.h2.test.synth.TestConcurrentUpdate
20:37:22 00:00.671 org.h2.test.synth.TestFuzzOptimizations seed: -7390465754649874180
20:37:23 00:00.741 org.h2.test.synth.TestRandomCompare
20:37:25 00:00.498 org.h2.test.db.TestDateStorage
20:37:29 00:04.578 org.h2.test.db.TestMultiThread
20:37:31 00:02.212 org.h2.test.poweroff.TestReorderWrites
```

<Figure 4. Mvcc Testing>



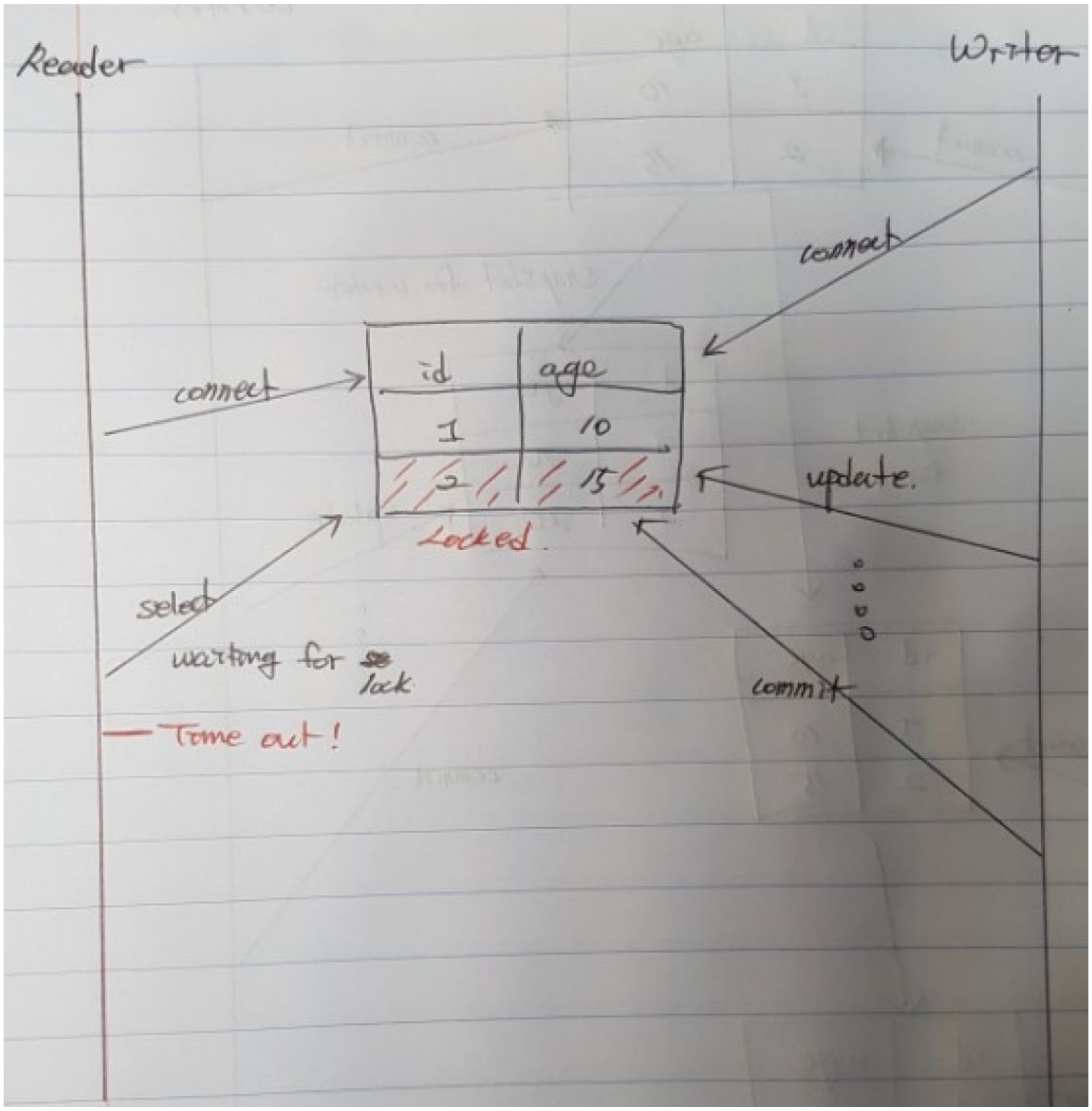
```
➜ test pwd
/Users/sugnkeunkim/Documents/workspace/H2-1.4.194/src/test/org/h2/test
➜ test ls
TestAll.java       bench        jdbc              package.html  server          testSimple.in.txt  utils
TestAllJunit.java  coverage     jdbcx             poweroff      store           todo
TestBase.java      db           mvcc              recover       synth           trace
ap                 jaqu         otherDatabases.txt rowlock      testScript.sql  unit
```

<Figure 4. Various test packages>

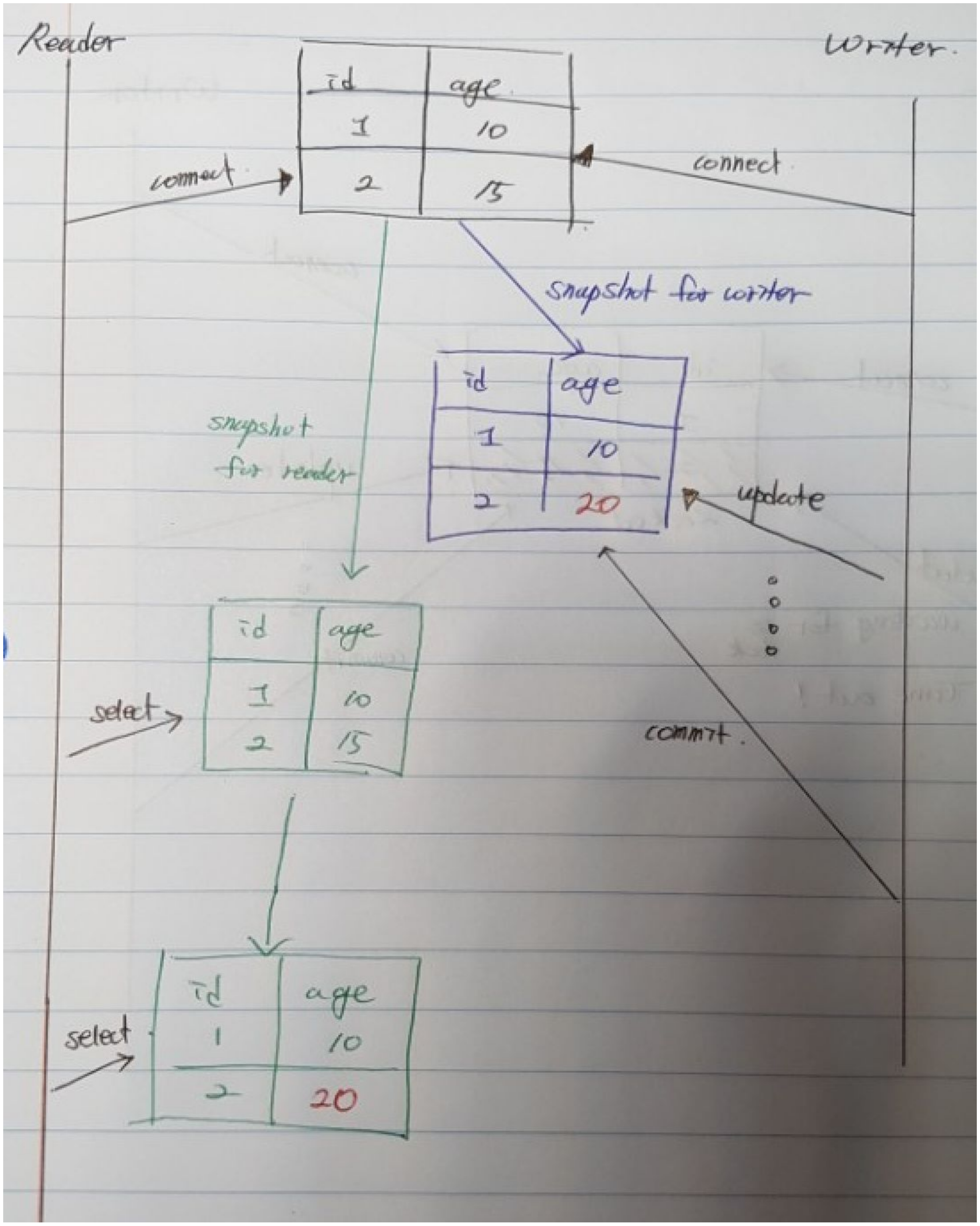
5. Next Step

This week, as a starting point of the H2 database and MVCC, I learned what the MVCC is and what it does under H2 database. Also, I made the exemplary program and downloaded H2 database source files and built them.

Next step, I am planning to learn the MVCC in more detail, and review the codes related to MVCC in the H2database.


Reader
Writer
connect
connect
id
age
1
10
2
15
update.
Locked
select
waiting for lock
commit
Time out!

<Figure 1. Scenario with disabled MVCC>


Reader
Writer.
connect
connect
snapshot for writer
snapshot for reader
update
select
commit
select
id | age
1 | 10
2 | 15
2 | 20

<Figure 2. Scenario with enabled MVCC>