TÖL303G
Gagnasafnsfræði
Vikublað 11

Snorri Agnarsson

3. nóvember 2022

Efni vikunnar — Material of the Week

Byrjum að ræða um færslur (*transactions*) og tengt efni svo sem hreyfingaskrár (*log*) og læsingar. Lesið aftur kafla 1.2.4 og lesið 6.6. Kannski byrjum við að ræða þriggja laga högun upplýsingakerfa.

We will start to discuss transactions and related matters such as log files and locks. Read again chapter 1.2.4 and read 6.6. Perhaps we will start to discuss three-tier information systems.

Verkefni — Assignments

Íslenska

Finnið Gradiance verkefni á venjulegum stað, verkefni vikunnar heita „Verkefni 11“.

Verkefnið sem skila skal í Gradescope er eftirfarandi.

Klárið eftirfarandi Java forrit og keyrið það á nokkra mismunandi vegu. Þennan forritstexta má finna í Canvas í skránum `V11.java-beinagrind-utf8` og `V11.java-beinagrind-cp1252`.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.PreparedStatement;

// Notkun: java -cp .;sqlite-jdbc-....jar V11 <args>
//         þar sem <args> er: [autocommit|noautocommit] [index|noindex]
```

```
// Eftir:  Búið er að mæla tíma fyrir gagnagrunnsaðgerðir og
//         skrifa niðurstöður

// Use:  java -cp .;sqlite-jdbc-....jar V11 <args>
//       where <args> is: [autocommit|noautocommit] [index|noindex]
// Post: The duration of database operations has been measured and
//       the results written.
public class V11
{
    public static void main( String[] args )
        throws Exception
    {
        Class.forName("org.sqlite.JDBC");
        boolean USE_AUTOCOMMIT = args[0].equals("autocommit");
        boolean USE_INDEX = args[1].equals("index");

        Connection conn = null;
        try
        {
            conn = DriverManager.getConnection("jdbc:sqlite:v11.db");
            conn.setAutoCommit(USE_AUTOCOMMIT);

            /* Hér vantar Java forritstexta sem gerir eftirfarandi:
             * 1. Eyðir töflunni R ef hún er til, þ.e. (í SQLite)
             *         DROP TABLE IF EXISTS R
             * 2. Eyðir vísaskránni RINDEX ef hún er til, þ.e.
             *         DROP INDEX IF EXISTS RINDEX
             * 3. Býr til töflu R sem hefur dálk key af tagi INTEGER
             *    sem skal vera lykill, og dálk value af tagi DOUBLE, þ.e.
             *         CREATE TABLE R( key INTEGER PRIMARY KEY, value DOUBLE )
             * 4. Býr til vísaskrá RINDEX fyrir dálkinn value í R, þ.e.
             *         CREATE INDEX RINDEX ON R(value)
             *    en aðeins ef USE_INDEX er satt.
             * 5. Býr til PreparedStatement pstmt til að setja gildi í R, þ.e.
             *         INSERT INTO R VALUES(?,?)
             */

            /* Here Java source code is missing that should do the following:
             * 1. Deletes the table R if it exists, i.e. (in SQLite)
             *         DROP TABLE IF EXISTS R
             * 2. Deletes the index RINDEX if it exists, i.e.
             *         DROP INDEX IF EXISTS RINDEX
             * 3. Creates a table R with a column key of type INTEGER
             *    which should be the key, and a column value of type DOUBLE, i.e.
             *         CREATE TABLE R( key INTEGER PRIMARY KEY, value DOUBLE )
             * 4. Creates an index RINDEX for the value column in R, i.e.
             *         CREATE INDEX RINDEX ON R(value)
             *    but only if USE_INDEX is true.
             * 5. Creates a PreparedStatement pstmt to insert values in R, i.e.
             *         INSERT INTO R VALUES(?,?)
             */

            long start,end;

            start = System.nanoTime();
            int i;
            for( i=0 ; i!=1000000 ; i++ )
            {
                /*
```

```
             * Hér vantar Java forritstexta til að bæta við tvennd (i,y)
             * í R þar sem y skal vera slembitala jafndreifð á bilinu
             * [0,2[. Kallið á Math.random() til að fá tölu á bilinu
             * [0,1[ og varpið tölunni með viðeigandi margföldun yfir
             * í bilið [0,2[.
             * Notið pstmt til að framkvæma innsetninguna.
             * Einnig skal fylgjast með tmanum með kalli á
             * System.nanoTime(). Ef meira en ein mínúta er liðin síðan
             * lykkjan hófst skal hætta í lykkjunni.
             *
             * Takið eftir að ef USE_AUTOCOMMIT er satt þá gerist sjálfkrafa
             * COMMIT eftir hverja SQL aðgerð í þessari lykkju.
             */

            /*
             * Here Java source code us missing to insert a tuple (i,y)
             * into R where y should be a random number uniformly distributed
             * in the interval [0,2[. Call Math.random() to return a random
             * in [0,1[ and scale it appropriately.
             * Use pstmt to perform the insertion.
             * Also the time should be monitored by calling
             * System.nanoTime(). If more than one minute has passes since the
             * loop started, the loop should be terminated.
             *
             * Note that if USE_AUTOCOMMIT is true then an automatic
             * COMMIT is performed after each SQL operation in this loop.
             */
        }
        if( !USE_AUTOCOMMIT ) conn.commit();

        end = System.nanoTime();
        System.out.println("Tími fyrir/Time for "+
                           i+" innsetningar/inserts: "+
                           (double)(end-start)/1e9
                          );

        System.out.println("Tími per innsetningu/Time per insert: "+
                           (double)(end-start)/1e9/i
                          );

        start = System.nanoTime();
        ResultSet r =
            stmt.executeQuery
                ("SELECT COUNT(*) FROM R WHERE "+
                 "value BETWEEN 0.05 AND 0.15"
                );
        r.next();
        System.out.println("Niðurstaða leitar/Result of search: "+r.getInt(1));
        System.out.println("Tími fyrir leit/Time for search: "+
                           (double)(System.nanoTime()-start)/1e9
                          );
    }
    catch(SQLException e)
    {
        System.err.println(e.getMessage());
    }
    finally
    {
        try
```

```
            {
                if(conn != null)
                  conn.close();
            }
            catch(SQLException e)
            {
                System.err.println(e);
            }
        }
    }
}
```

Skilið eftirfarandi í Gradescope:

- Öllum forritstexta fyrir ykkar endanlega Java forrit.
- Eftirfarandi töflu, útfylltri með viðeigandi tölum til að sýna tímann sem það tekur fyrir hverja innsetningu í töfluna R. Helst skal sýna meðaltíma fyrir hverja innsetningu af 1.000.000 innsettum röðum. Það er hins vegar ekki alltaf raunhæft. Ef í ljós kemur að það er ekki raunhæft þá skuluð þið láta texta fylgja með til að útskýra hvernig tölurnar eru reiknaðar. Notið viðeigandi einingar í töflunni, til dæmis sekúndur (s), nanósekúndur (ns), millisekúndur (ms), míkrósekúndur (μs), mínútur (mín), klukkustundir (klst). **Munið að taflan á að sýna meðaltíma per aðgerð, ekki heildartíma.**

Tími fyrir innsetningu			
Án vísis		**Með vísi**	
Án AutoCommit	**Með AutoCommit**	**Án AutoCommit**	**Með AutoCommit**

- Einnig eftirfarandi töflu, útfylltri með viðeigandi tölum. Ef ykkur finnst þörf á getið þið breytt forritinu þannig að margar leitir séu framkvæmdar til að hægt sé að reikna meðaltíma. Þá þarf hins vegar að sjá til þess að ekki sé aftur og aftur verið að framkvæma sömu leitina. Fyrir þessar mælingar skuluð þið ekki nota autocommit.

Tími fyrir leit	
Án vísis	**Með vísi**

English

Find the Gradiance assignment in the usual place, the weeks assignment is named "Verkefni 11".

The assignment you should turn in to Gradescope is the following.

Finish programming the following Java program and run it in various ways. This source code can be found in Canvas in the files `V11.java-beinagrind-utf8` and `V11.java-beinagrind-cp1252`.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.PreparedStatement;

// Notkun: java -cp .;sqlite-jdbc-....jar V11 <args>
//         þar sem <args> er: [autocommit|noautocommit] [index|noindex]
// Eftir:  Búið er að mæla tíma fyrir gagnagrunnsaðgerðir og
//         skrifa niðurstöður

// Use:  java -cp .;sqlite-jdbc-....jar V11 <args>
//       where <args> is: [autocommit|noautocommit] [index|noindex]
// Post: The duration of database operations has been measured and
//       the results written.
public class V11
{
    public static void main( String[] args )
        throws Exception
    {
        Class.forName("org.sqlite.JDBC");
        boolean USE_AUTOCOMMIT = args[0].equals("autocommit");
        boolean USE_INDEX = args[1].equals("index");

        Connection conn = null;
        try
        {
            conn = DriverManager.getConnection("jdbc:sqlite:v11.db");
            conn.setAutoCommit(USE_AUTOCOMMIT);

            /* Hér vantar Java forritstexta sem gerir eftirfarandi:
             * 1. Eyðir töflunni R ef hún er til, þ.e. (í SQLite)
             *         DROP TABLE IF EXISTS R
             * 2. Eyðir vísaskránni RINDEX ef hún er til, þ.e.
             *         DROP INDEX IF EXISTS RINDEX
             * 3. Býr til töflu R sem hefur dálk key af tagi INTEGER
             *    sem skal vera lykill, og dálk value af tagi DOUBLE, þ.e.
             *         CREATE TABLE R( key INTEGER PRIMARY KEY, value DOUBLE )
             * 4. Býr til vísaskrá RINDEX fyrir dálkinn value í R, þ.e.
             *         CREATE INDEX RINDEX ON R(value)
             *    en aðeins ef USE_INDEX er satt.
             * 5. Býr til PreparedStatement pstmt til að setja gildi í R, þ.e.
             *         INSERT INTO R VALUES(?,?)
             */

            /* Here Java source code is missing that should do the following:
             * 1. Deletes the table R if it exists, i.e. (in SQLite)
             *         DROP TABLE IF EXISTS R
             * 2. Deletes the index RINDEX if it exists, i.e.
             *         DROP INDEX IF EXISTS RINDEX
             * 3. Creates a table R with a column key of type INTEGER
             *    which should be the key, and a column value of type DOUBLE, i.e.
             *         CREATE TABLE R( key INTEGER PRIMARY KEY, value DOUBLE )
             * 4. Creates an index RINDEX for the value column in R, i.e.
             *         CREATE INDEX RINDEX ON R(value)
             *    but only if USE_INDEX is true.
             * 5. Creates a PreparedStatement pstmt to insert values in R, i.e.
             *         INSERT INTO R VALUES(?,?)
```

```
 */

long start,end;

start = System.nanoTime();
int i;
for( i=0 ; i!=1000000 ; i++ )
{
    /*
     * Hér vantar Java forritstexta til að bæta við tvennd (i,y)
     * í R þar sem y skal vera slembitala jafndreifð á bilinu
     * [0,2[. Kallið á Math.random() til að fá tölu á bilinu
     * [0,1[ og varpið tölunni með viðeigandi margföldun yfir
     * í bilið [0,2[.
     * Notið pstmt til að framkvæma innsetninguna.
     * Einnig skal fylgjast með tmanum með kalli á
     * System.nanoTime(). Ef meira en ein mínúta er liðin síðan
     * lykkjan hófst skal hætta í lykkjunni.
     *
     * Takið eftir að ef USE_AUTOCOMMIT er satt þá gerist sjálfkrafa
     * COMMIT eftir hverja SQL aðgerð í þessari lykkju.
     */

    /*
     * Here Java source code us missing to insert a tuple (i,y)
     * into R where y should be a random number uniformly distributed
     * in the interval [0,2[. Call Math.random() to return a random
     * in [0,1[ and scale it appropriately.
     * Use pstmt to perform the insertion.
     * Also the time should be monitored by calling
     * System.nanoTime(). If more than one minute has passes since the
     * loop started, the loop should be terminated.
     *
     * Note that if USE_AUTOCOMMIT is true then an automatic
     * COMMIT is performed after each SQL operation in this loop.
     */
}
if( !USE_AUTOCOMMIT ) conn.commit();

end = System.nanoTime();
System.out.println("Tími fyrir/Time for "+
                   i+" innsetningar/inserts: "+
                   (double)(end-start)/1e9
                  );

System.out.println("Tími per innsetningu/Time per insert: "+
                   (double)(end-start)/1e9/i
                  );

start = System.nanoTime();
ResultSet r =
    stmt.executeQuery
        ("SELECT COUNT(*) FROM R WHERE "+
         "value BETWEEN 0.05 AND 0.15"
        );
r.next();
System.out.println("Niðurstaða leitar/Result of search: "+r.getInt(1));
System.out.println("Tími fyrir leit/Time for search: "+
                   (double)(System.nanoTime()-start)/1e9
```

```
                            );
        }
        catch(SQLException e)
        {
            System.err.println(e.getMessage());
        }
        finally
        {
            try
            {
                if(conn != null)
                  conn.close();
            }
            catch(SQLException e)
            {
                System.err.println(e);
            }
        }
    }
}
```

Turn in the following in Gradescope:

- The whole source code for your final Java program.
- The following table, filled in with the appropriate numbers to show the time taken for each insert into the table R. Preferably show the average time for each of 1,000,000 inserted row. However, this may not always be practical. If it turns out not to be practical then describe how the numbers are derived. Use the appropriate units in the table, for example seconds (s), nanoseconds (ns), milliseconds (ms), microseconds (μs), minutes (min), hours (hours). **Remember that the table should show the average time per operation, not the total time.**

Time for insertion			
Without index		**With index**	
Without AutoCommit	**With AutoCommit**	**Without AutoCommit**	**With AutoCommit**

- Also turn in the following table, filled in with the appropriate numbers. If you feel the need you can change the program to perform many searches so that the average time can be computed. In that case you must ensure that the same search is not performed again and again. For these measurements you should not use autocommit.

Time for search	
Without index	**With index**