Project 5 Report

Task 1

- How did you use connection pooling?

By adding the resource “jdbc/TestDB” to the connection pooling, we were allowed to reference this resource in the web.xml. Since we did that, in our java files, we were able to use this connection that we referenced in the web.xml and instead of connecting to the database every time in each java file that required a JDBC connection.

- File name, line numbers as in Github

File path: cs122b-winter18-team-10/project2/WebContent/META-INF/context.xml
Line numbers: 1-9

File path: cs122b-winter18-team-10/project2/WebContent/WEB-INF/web.xml
Line numbers: 12-26

File path: cs122b-winter18-team10/project2/src/Login.java
Line numbers: 82-103

For all the java files that required JDBC pooling, we used the same code
File path: cs122b-winter18-team10/project2/src/AndroidCount.java
Line numbers: 54-75
File path: cs122b-winter18-team10/project2/src/AndroidLogin.java
Line numbers: 63-84
File path: cs122b-winter18-team10/project2/src/AndroidSearch.java
Line numbers: 57-78
File path: cs122b-winter18-team10/project2/src/Checkout.java
Line numbers: 73-94
File path: cs122b-winter18-team10/project2/src/Confirm.java
Line numbers: 67-88
File path: cs122b-winter18-team10/project2/src/EmployeeLogin.java
Line numbers: 81-102
File path: cs122b-winter18-team10/project2/src/Insert.java
Line numbers: 56-77
File path: cs122b-winter18-team10/project2/src/MainSuggestion.java
Line numbers: 58-79
File path: cs122b-winter18-team10/project2/src/Metadata.java
Line numbers: 56-77
File path: cs122b-winter18-team10/project2/src/MovieListBrowse.java
Line numbers: 55-76
File path: cs122b-winter18-team10/project2/src/MovieListSearch.java
Line numbers: 58-79

File path: cs122b-winter18-team10/project2/src/SingleMovie.java
Line numbers: 59-80
File path: cs122b-winter18-team10/project2/src/SingleStars.java
Line numbers: 56-77
File path: cs122b-winter18-team10/project2/src/addMovie.java
Line numbers: 57-78
File path: cs122b-winter18-team10/project2/src/importMovies.java
Line numbers: 111-132
File path: cs122b-winter18-team10/project2/src/movieStarSuggestion.java
Line numbers: 54-75

- Snapshots

context.xml

```
<Context path="/project2">

    <Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
        maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
        password="mypassword" driverClassName="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true" />


</Context>
```

web.xml

```
  <resource-ref>
    <description>
            Resource reference to a factory for java.sql.Connection
            instances that may be used for talking to a particular
            database that
            is configured in the server.xml file.
        </description>
    <res-ref-name>
            jdbc/TestDB
        </res-ref-name>
    <res-type>
            javax.sql.DataSource
        </res-type>
    <res-auth>Container</res-auth>
  </resource-ref>
```

Login.java

```
82              Context initCtx = new InitialContext();
83              if (initCtx == null)
84                  out.println("initCtx is NULL");
85
86              Context envCtx = (Context) initCtx.lookup("java:comp/env");
87              if (envCtx == null)
88                  out.println("envCtx is NULL");
89
90              // Look up our data source
91              DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
92
93              // the following commented lines are direct connections without pooling
94              //Class.forName("org.gjt.mm.mysql.Driver");
95              //Class.forName("com.mysql.jdbc.Driver").newInstance();
96              //Connection dbcon = DriverManager.getConnection(loginUrl, loginUser, loginPasswd);
97
98              if (ds == null)
99                  out.println("ds is null.");
100
101             Connection dbcon = ds.getConnection();
102             if (dbcon == null)
103                 out.println("dbcon is null.");
```

For all the java files that required JDBC pooling, we used the same code, so for the sake of saving space we will not attach screenshots.

- How did you use Prepared Statements?

For each of our java files that were involved in the search function, instead of just adding the string into the query and executing the query, we used Prepared Statements and set the strings with the search parameters and then executed the prepared statements and saved it to a Result Set so we could get the results back. One problem that we ran into was that the prepared statements could not find the “?” to place the values in since the question mark was hidden in the percent and star symbols. A solution that we found was to add the percent and star symbols in ps.setString() statement.

- File name, line numbers as in Github

File path: cs122b-winter18-team10/project2/src/AndroidSearch.java
Line numbers: 84-113

File path: cs122b-winter18-team10/project2/src/MainSuggestion.java
Line numbers: 86-114

File path: cs122b-winter18-team10/project2/src/MovieListSearch.java
Line numbers: 88-135

File path: cs122b-winter18-team10/project2/src/movieStarSuggestion.java
Line numbers: 94-134

- Snapshots

AndroidSearch.java

```
        int count = 0;
        PreparedStatement ps = null;

        String queryInner = "";
        if (prefix.length>0) {
            queryInner += "(select movies.id from movies where match (title) against (? in boolean mode)";
            count++;
            for (int i = 1; i < prefix.length; ++i) {
                queryInner += "\nand match(title) against (? in boolean mode)";
                count++;
            }
        }

        if (queryInner.equals("(select movies.id from movies where match (title) against ('*' in boolean mode)"))
            queryInner = "(select movies.id from movies where match (title) against ('' in boolean mode)";
        queryInner += ")";
        String queryOuter = "Select movies.id, movies.title,movies.year,movies.director,group_concat(distinct stars.name),
                "from movies, stars_in_movies, stars, genres, genres_in_movies where \n" +
                "stars_in_movies.movieId = movies.id and stars_in_movies.starId = stars.id\n" +
                "and genres.id = genres_in_movies.genreId and genres_in_movies.movieId = movies.id\n" +
                "and movies.id in\n";
        queryOuter += queryInner;
        queryOuter += "\ngroup by movies.id";

        ps = dbcon.prepareStatement(queryOuter);

        for (int i = 0; i<count; ++i) {
            ps.setString(i+1,prefix[i]+'*');
        }
        ResultSet rs = ps.executeQuery();
```

MainSuggestion.java

```
    PreparedStatement ps = null;
    int count = 0;
    String queryInner = "";
    if (prefix.length>0) {
        queryInner += "(select movies.id from movies where match (title) against (? in boolean mode)";
        count++;
        for (int i = 1; i < prefix.length; ++i) {
            queryInner += "\nand match(title) against (? in boolean mode)";
            count++;
        }
    }

    if (queryInner.equals("(select movies.id from movies where match (title) against ('*' in boolean mode)"))
        queryInner = "(select movies.id from movies where match (title) against ('' in boolean mode)";
    queryInner += ")";
    String queryOuter = "Select movies.id, movies.title,movies.year,movies.director,group_concat(distinct stars.name), g
            "from movies, stars_in_movies, stars, genres, genres_in_movies where \n" +
            "stars_in_movies.movieId = movies.id and stars_in_movies.starId = stars.id\n" +
            "and genres.id = genres_in_movies.genreId and genres_in_movies.movieId = movies.id\n" +
            "and movies.id in\n";
    queryOuter += queryInner;
    queryOuter += "\ngroup by movies.id";
    System.out.println(queryOuter);
    System.out.println(count);
    ps = dbcon.prepareStatement(queryOuter);
    for (int i = 0; i < count; i++) {
        ps.setString(i+1, prefix[i]+"*");
    }
    ResultSet rs = ps.executeQuery();
```

MovieListSearch.java

```
88              PreparedStatement ps = null;
89              String query = "Select movies.id, movies.title,movies.year,movies.director,group_concat(distinct stars.nam
90                      + "from movies, stars_in_movies, stars, genres, genres_in_movies where \n"
91                      + "stars_in_movies.movieId = movies.id and stars_in_movies.starId = stars.id\n"
92                      + "and genres.id = genres_in_movies.genreId and genres_in_movies.movieId = movies.id\n"
93                      +  "and movies.id in\n"
94                          + "(Select movies.id\n"
95                          + "From movies, stars_in_movies, stars, genres, genres_in_movies\n"
96                          + "Where stars_in_movies.movieId = movies.id and stars_in_movies.starId = stars.id\n"
97                          + "and genres.id = genres_in_movies.genreId and genres_in_movies.movieId = movies.id\n"
98                          + "and upper(movies.title) like upper(?) and upper(movies.director) like upper(?)\n"
99                          + "and upper(stars.name) like upper(?)\n";
100             if (!year.isEmpty()) {
101                 try {
102                     int i = Integer.parseInt(year);
103                     System.out.println(i);
104                     query = query + "and movies.year = ?\n";
105                     query = query + "group by movies.id)\n"
106                             + "group by movies.id";
107                     ps = dbcon.prepareStatement(query);
108                     ps.setString(1,'%'+title+'%');
109                     ps.setString(2,'%'+dir+'%');
110                     ps.setString(3,'%'+star+'%');
111                     ps.setString(4,year);
112
113                 }
114                 catch(NumberFormatException e) {
115                     query = query + "and movies.year =-1\n";
116                     query = query + "group by movies.id)\n"
117                             + "group by movies.id";

118                      ps = dbcon.prepareStatement(query);
119                      ps.setString(1,'%'+title+'%');
120                      ps.setString(2,'%'+dir+'%');
121                      ps.setString(3,'%'+star+'%');
122                  }
123
124             }else {
125                 query = query + "group by movies.id)\n"
126                         + "group by movies.id";
127                 ps = dbcon.prepareStatement(query);
128                 ps.setString(1,'%'+title+'%');
129                 ps.setString(2,'%'+dir+'%');
130                 ps.setString(3,'%'+star+'%');
131
132             }
133
134             JsonArray jsonArray = new JsonArray();
135             ResultSet rs = ps.executeQuery();
```

movieStarSuggestion.java

```
94            PreparedStatement psmovie = null;
95            PreparedStatement psstar = null;
96            int count = 0;
97
98            String queryMovie = "";
99            if (prefix.length>0) {
100               queryMovie += "select movies.id, movies.title from movies where match (title) against (? in boolean mode)
101               count++;
102               for (int i = 1; i < prefix.length; ++i) {
103                   queryMovie += "\nand match(title) against (? in boolean mode)";
104                   count++;
105
106               }
107           }
108
109           if (queryMovie.equals("select movies.id, movies.title from movies where match (title) against ('*' in boolean
110               queryMovie = "select movies.id, movies.title from movies where match (title) against ('' in boolean mode)
111
112
113           // STAR
114           String queryStar = "";
115           if (prefix.length>0) {
116               queryStar += "select stars.id, stars.name from stars where match (name) against (? in boolean mode)";
117               for (int i = 1; i < prefix.length; ++i) {
118                   queryStar += "\nand match(name) against (? in boolean mode)";
119               }
120           }
121
122            if (queryStar.equals("select stars.id, stars.name from stars where match (name) against ('*' in boolean mo
123                queryStar = "select stars.id, stars.name from stars where match (name) against ('' in boolean mode)";
124
125
126            psmovie = dbcon.prepareStatement(queryMovie);
127            psstar = dbcon.prepareStatement(queryStar);
128
129            for (int i = 0; i<count; ++i) {
130                psmovie.setString(i+1,prefix[i]+'*');
131                psstar.setString(i+1,prefix[i]+'*');
132            }
133            ResultSet rs = psmovie.executeQuery();
134            ResultSet rs2 = psstar.executeQuery();
```

Task 2

- Address of AWS and Google instances

Load Balancer/Instance 1:

- Public: 18.218.178.161
- Private: 172.31.41.121

Master/Instance 2:

- Public: 18.220.202.224
- Private:172.31.46.146

Slave/Instance 3:

- Public: 52.14.136.195
- Private: 172.31.34.136

Google Instance:

- Public: 35.229.51.22
- Private: 10.142.0.2

- Have you verified that they are accessible? Does Fablix site get opened both on Google's 80 port and AWS' 8080 port?

Yes and yes

- How connection pooling works with two backend SQL?

We added the resource "jdbc/master" to the connection pool in the context.xml file.This "jdbc/master" uses the master IP instead of localhost because we want to write only into the master SQL instance. By doing this, we were able to create a resource-ref inside of our web.xml file. By adding this "jdbc/master" resource to the connection pool, the two backend SQL could either use this new connection or the old connection that we made earlier that just uses localhost in the url.

- File name, line numbers as in Github

File path: cs122b-winter18-team-10/project2/WebContent/META-INF/context.xml
Line numbers: 8-11
File path: cs122b-winter18-team-10/project2/WebContent/WEB-INF/web.xml
Line numbers: 27-38

- Snapshots

context.xml

```
<Context path="/project2">

    <Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
        maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
        password="mypassword" driverClassName="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true" />

    <Resource name="jdbc/master" auth="Container" type="javax.sql.DataSource"
        maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
        password="mypassword" driverClassName="com.mysql.jdbc.Driver"
        url="jdbc:mysql://172.31.46.146:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true" />

</Context>
```

web.xml

```
<resource-ref>
  <description>
          Master DB
      </description>
  <res-ref-name>
          jdbc/master
      </res-ref-name>
  <res-type>
          javax.sql.DataSource
      </res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

- How read/write requests were routed?

So we examined each of our java files and in the files that required a change such as an insert into the SQL server, we would make sure the connection we used was the "jdbc/master". This

ensured that we would not be able to write into the slave SQL instance. And for the read requests, we just left it as “jdbc/TestDB” because this connection used localhost and it didn't matter which instance we were on when all we had to do was read from the SQL server. So the only java files that we changed were the ones that updated the SQL server.

- File name, line numbers as in Github

File path: cs122b-winter18-team-10/project2/src/Confirm.java
Line numbers: 76
File path: cs122b-winter18-team-10/project2/src/Insert.java
Line numbers: 65
File path: cs122b-winter18-team-10/project2/src/addMovie.java
Line numbers: 66
File path: cs122b-winter18-team-10/project2/src/importMovies.java
Line numbers: 120

- Snapshots

Confirm.java

We provided a screenshot from Confirm.java. All the other files (Insert, addMovie, and importMovies) used the same line of code.

Task 3

- Have you uploaded the log file to Github? Where is it located?

Yes
File path: cs122b-winter18-team-10/project2/test.txt

- Have you uploaded the HTML file to Github? Where is it located?

- Have you uploaded the script to Github? Where is it located?

Yes
File path: cs122b-winter18-team-10/project2/src/logparser.java

- Have you uploaded the WAR file and README to Github? Where is it located?

Yes
File path: cs122b-winter18-team-10/project2/project2.war
Yes
File path: cs122b-winter18-team-10/READ.md