CS 122B - Project 5 Report

Task 1. Prepared Statements

Prepared statements are utilized within servlets that require retrieving query results from the database, including:

MovieList.java

```
136
137             query = "SELECT M.id, M.title, M.year, M.director, GROUP_CONCAT(DISTINCT G.name SEPARA
138                     "FROM movies M\n" +
139                     "LEFT JOIN stars_in_movies SM ON M.id = SM.movieId\n" +
140                     "LEFT JOIN stars S ON SM.starId = S.id\n" +
141                     "LEFT JOIN genres_in_movies RM ON M.id = RM.movieId\n" +
142                     "LEFT JOIN genres G ON RM.genreId = G.id \n" +
143                     "LEFT JOIN ratings R ON M.id = R. movieId \n" +
144                     "WHERE MATCH (M.title) AGAINST( ? IN BOOLEAN MODE) \n" +
145                     "GROUP BY M.id, M.title, M.year, M.director, R.rating LIMIT 500;\n";
146
147             tj_startTime = System.currentTimeMillis();
148             preparedStatement = dbcon.prepareStatement(query);
149             preparedStatement.setString(1, where);
150         }
151
```

cs122b-spring18-team-47/project2/src/MovieList.java (Line: 148 - 149)

SingleStarServlet.java

```
53              String query = "SELECT T.name, T.birthYear, GROUP_CONCAT(DISTINCT M.title SEPARATOR', ') AS movies\n" +
54                             "FROM (SELECT * FROM stars S WHERE S.name = ? ) AS T\n" +
55                             "LEFT JOIN stars_in_movies SM ON SM.starId = T.id\n" +
56                             "LEFT JOIN movies M ON M.id = SM.movieId\n" +
57                             "GROUP BY T.name, T.birthYear;\n";
58
59
60              PreparedStatement preparedStatement = dbcon.prepareStatement(query);
61              preparedStatement.setString(1, name);
62              ResultSet rs = preparedStatement.executeQuery();
63
64
```

cs122b-spring18-team-47/project2/src/SingleStarServlet.java (Line: 60 - 62)

SingleMovieServlet.java

```
57
58      String query = "SELECT DISTINCT M.id, M.title, M.year, M.director, GROUP_CONCAT(DISTINCT G.name SEPARATOR ', ') AS genres
59              "FROM movies M \n" +
60              "LEFT JOIN stars_in_movies SM ON M.id = SM.movieId \n" +
61              "LEFT JOIN stars S ON SM.starId = S.id\n" +
62              "LEFT JOIN genres_in_movies RM ON M.id = RM.movieId \n" +
63              "LEFT JOIN genres G ON RM.genreId = G.id\n" +
64              "LEFT JOIN ratings R ON M.id = R. movieId \n" +
65              "WHERE  M.title = ? \n" +
66              "GROUP BY M.id, M.title, M.year, M.director, R.rating;";
67
68
69      PreparedStatement preparedStatement = dbcon.prepareStatement(query);
70      preparedStatement.setString(1, name);
71      ResultSet rs = preparedStatement.executeQuery();
72
```

cs122b-spring18-team-47/project2/src/SingleMovieServlet.java (Line: 69 - 71)

Below is the complete list of paths & line numbers to servlets that utilize prepared statements in addition to the files displayed above:

- *cs122b-spring18-team-47/project2/src/CheckoutServlet.java (Line: 60 - 66)*
- *cs122b-spring18-team-47/project2/src/EmployeeLoginServlet.java (Line: 53 - 55)*
- *cs122b-spring18-team-47/project2/src/InsertMovieServlet.java (Line: 69 - 73, 83 - 84, 99 - 101, 109 - 110, 120 - 123, 132 - 134, 143 - 144, 152 - 155, 165 - 176)*
- *cs122b-spring18-team-47/project2/src/InsertStarServlet.java (Line: 58 - 59, 77 - 81, 88 - 94)*
- *cs122b-spring18-team-47/project2/src/LoginServlet.java (Line: 75 - 77)*
- *cs122b-spring18-team-47/project2/src/SalesServlet.java (Line: 67 - 74, 83 - 85)*
- *cs122b-spring18-team-47/project2/src/MetadataServlet.java (Line: 50 - 51, 70 - 72)*
- *cs122b-spring18-team-47/project2/src/SuggestionServlet.java (Line: 73 - 75)*

Task 1. Connection Pooling

Connection pooling is configured in the following context.xml and web.xml files using LocalDB and MasterDB:

context.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<Context mapperContextRootRedirectEnabled="true">

    <Resource name="jdbc/moviedb"
              auth="Container"
              driverClassName="com.mysql.jdbc.Driver"
              type="javax.sql.DataSource"
              username="james"
              password="mypassword"
              url="jdbc:mysql://13.58.209.21:3306/moviedb"/>


     <Resource name="jdbc/LocalDB"
               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/MasterDB"
               auth="Container"
               type="javax.sql.DataSource"
               maxTotal="100" maxIdle="30" maxWaitMillis="10000"
               username="mytestuser"
               password="mypassword"
               driverClassName="com.mysql.jdbc.Driver"
               url="jdbc:mysql://18.191.129.160:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
</Context>
```

cs122b-spring18-team-47/project2/WebContent/META-INF/context.xml (Line: 14 - 21)

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xs
  <display-name>project2</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

  <resource-ref>
    <description>MySQL DataSource example</description>
    <res-ref-name>jdbc/LocalDB</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
  </resource-ref>


  <resource-ref>
    <description>MySQL DataSource MasterDB</description>
    <res-ref-name>jdbc/MasterDB</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
  </resource-ref>


</web-app>
```

cs122b-spring18-team-47/project2/WebContent/WEB-INF/web.xml (Line: 13 - 18)

Connection pooling is utilized within all servlets like such (the jdbc resource used differs depending on the read/write functionalities of each servlet):

MovieList.java

```
        BufferedWriter writer = new BufferedWriter(new FileWriter(myfile, true));


        try {
            Context initCtx = new InitialContext();
            Context envCtx = (Context) initCtx.lookup("java:comp/env");
            if (envCtx == null)
                out.println("envCtx is NULL");
            dataSource = (DataSource) envCtx.lookup("jdbc/LocalDB");
            Connection dbcon = dataSource.getConnection();
            if (dbcon == null)
                out.println("dbcon is null.");

            JsonArray jsonArray = new JsonArray();

```

cs122b-spring18-team-47/project2/src/MovieList.java (Line: 58 - 65)

SingleMovieServlet.java

```
44          try {
45                  Context initCtx = new InitialContext();
46
47                  Context envCtx = (Context) initCtx.lookup("java:comp/env");
48                  if (envCtx == null)
49                      out.println("envCtx is NULL");
50
51                  dataSource = (DataSource) envCtx.lookup("jdbc/LocalDB");
52
53                  Connection dbcon = dataSource.getConnection();
54
55                  if (dbcon == null)
56                      out.println("dbcon is null.");
57
```

cs122b-spring18-team-47/project2/src/SingleMovieServlet.java (Line: 45 - 56)

SingleStarServlet.java

```
37          try {
38              Context initCtx = new InitialContext();
39
40              Context envCtx = (Context) initCtx.lookup("java:comp/env");
41              if (envCtx == null)
42                  out.println("envCtx is NULL");
43
44              dataSource = (DataSource) envCtx.lookup("jdbc/LocalDB");
45
46              Connection dbcon = dataSource.getConnection();
47
48              if (dbcon == null)
49                  out.println("dbcon is null.");
50
```

cs122b-spring18-team-47/project2/src/SingleStarServlet.java (Line: 38 - 49)

Below is the complete list of paths & line numbers to servlets that utilize connection pooling in addition to the files displayed above:

- *cs122b-spring18-team-47/project2/src/CheckoutServlet.java (Line: 43 - 54)*
- *cs122b-spring18-team-47/project2/src/EmployeeLoginServlet.java (Line: 39 - 50)*
- *cs122b-spring18-team-47/project2/src/InsertMovieServlet.java (Line: 52 - 63)*
- *cs122b-spring18-team-47/project2/src/InsertStarServlet.java (Line: 44 - 55)*
- *cs122b-spring18-team-47/project2/src/LoginServlet.java (Line: 61 - 72)*
- *cs122b-spring18-team-47/project2/src/SalesServlet.java (Line: 48 - 59)*
- *cs122b-spring18-team-47/project2/src/MetadataServlet.java (Line: 36 - 47)*
- *cs122b-spring18-team-47/project2/src/SuggestionServlet.java (Line: 56 - 67)*

Task 2. URLs

The following IP addresses are all accessible and can be opened via Google's 80 port and AWS's 8080 port:

- Google Instance: 35.237.67.190
- AWS Instance 1 (Load Balancing): 18.220.221.195
- AWS Instance 2 (Master): 18.191.129.160
- AWS Instance 3 (Slave): 18.191.118.83

Task 2. Load Balancing

With two backend SQLs, connection pooling works by recognizing a user's request and requesting a connection from the pool in order to gain access to mySQL server. The connection is returned to the connection pool once the session is over so that it can later on be re-used by other requesting threads.

```
57          try {
58              Context initCtx = new InitialContext();
59              Context envCtx = (Context) initCtx.lookup("java:comp/env");
60              if (envCtx == null)
61                  out.println("envCtx is NULL");
62              dataSource = (DataSource) envCtx.lookup("jdbc/LocalDB");
63              Connection dbcon = dataSource.getConnection();
64              if (dbcon == null)
65                  out.println("dbcon is null.");
66
67              JsonArray jsonArray = new JsonArray();
68
69              String query = "";
70              PreparedStatement preparedStatement = dbcon.prepareStatement("");
```

cs122b-spring18-team-47/project2/src/MovieList.java

Read requests will be routed to localhost MySQL server, which can be either master instance or slave instance. However, write requests will be routed to master instance. The following screenshot of context.xml shows that it is connected to masterDB by using master instance IP address.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <Context mapperContextRootRedirectEnabled="true">
4
5     <Resource name="jdbc/moviedb"
6               auth="Container"
7               driverClassName="com.mysql.jdbc.Driver"
8               type="javax.sql.DataSource"
9               username="james"
10              password="mypassword"
11              url="jdbc:mysql://13.58.209.21:3306/moviedb"/>
12
13
14      <Resource name="jdbc/LocalDB"
15                auth="Container"
16                type="javax.sql.DataSource"
17                maxTotal="100" maxIdle="30" maxWaitMillis="10000"
18                username="mytestuser"
19                password="mypassword"
20                driverClassName="com.mysql.jdbc.Driver"
21                url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
22
23      <Resource name="jdbc/MasterDB"
24                auth="Container"
25                type="javax.sql.DataSource"
26                maxTotal="100" maxIdle="30" maxWaitMillis="10000"
27                username="mytestuser"
28                password="mypassword"
29                driverClassName="com.mysql.jdbc.Driver"
30                url="jdbc:mysql://18.191.129.160:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
31 </Context>
```

cs122b-spring18-team-47/project2/WebContent/META-INF/context.xml

Servlets such as MovieList.java below do READ requests as they only search the database for the queried movies:

```
57          try {
58              Context initCtx = new InitialContext();
59              Context envCtx = (Context) initCtx.lookup("java:comp/env");
60              if (envCtx == null)
61                  out.println("envCtx is NULL");
62              dataSource = (DataSource) envCtx.lookup("jdbc/LocalDB");
63              Connection dbcon = dataSource.getConnection();
64              if (dbcon == null)
65                  out.println("dbcon is null.");
66
67              JsonArray jsonArray = new JsonArray();
68
69              String query = "";
70              PreparedStatement preparedStatement = dbcon.prepareStatement("");
```

cs122b-spring18-team-47/project2/src/MovieList.java (Line: 57 - 70)

Other servlets that utilize LocalDB:

- *cs122b-spring18-team-47/project2/src/CheckoutServlet (Line: 42 - 66)*
- *cs122b-spring18-team-47/project2/src/EmployeeLoginServlet (Line: 38 - 55)*
- *cs122b-spring18-team-47/project2/src/LoginServlet (Line: 60-77)*
- *cs122b-spring18-team-47/project2/src/MetadaServlet (Line: 35-44)*
- *cs122b-spring18-team-47/project2/src/SingleMovieServlet (Line: 44-71)*
- *cs122b-spring18-team-47/project2/src/SingleStarServlet (Line: 37- 62)*
- *cs122b-spring18-team-47/project2/src/SuggestionServlet (Line: 55-75)*

Servlets such as SalesServlet.java below do WRITE requests as they modify the database with new entries regarding user/movie information:

```
47          try {
48              Context initCtx = new InitialContext();
49
50              Context envCtx = (Context) initCtx.lookup("java:comp/env");
51              if (envCtx == null)
52                  out.println("envCtx is NULL");
53
54              dataSource = (DataSource) envCtx.lookup("jdbc/MasterDB");
55
56              Connection dbcon = dataSource.getConnection();
57
58              if (dbcon == null)
59                  out.println("dbcon is null.");
60
61              if(todo == null) {
62                  String query = "INSERT INTO sales(customerID, movieId, saleDate) VALUES(\n" +
63                                 "(SELECT  C.id\n" +
64                                 "FROM customers C \n" +
65                                 "WHERE C.firstName = ? AND C.lastName = ? AND C.ccId = ?), ? ,(SELECT CURDATE() AS date));";
66
67                  PreparedStatement preparedStatement = dbcon.prepareStatement(query);
68
69                  preparedStatement.setString(1, first_name);
70                  preparedStatement.setString(2, last_name);
71                  preparedStatement.setString(3, credit_id);
72                  preparedStatement.setString(4, movieID);
73
74                  preparedStatement.executeUpdate(query);
75              }
```

cs122b-spring18-team-47/project2/src/SalesServlet.java (Line: 48 - 59)

Other servlets that utilize MasterDB:

- *cs122b-spring18-team-47/project2/src/InsertMovieServlet.java (Line: 52 - 63)*
- *cs122b-spring18-team-47/project2/src/InsertStarServlet.java (Line: 44 - 55)*

Task 3.
Script: The Python script used to parse the log file is located at cs122b-spring18-team-47/project2/WebContent/TimeParser.py on GitHub.

Log Files: The log files are located at cs122b-spring18-team-47/project2/WebContent/logs/ folder on GitHub.

HTML File: The jmeter_report.html along with all the relevant screenshots are located under cs122b-spring18-team-47/project2/WebContent/performance folder on GitHub.

WAR & README: The WAR and README files are located at **cs122b-spring18-team-47** folder on GitHub.