Peter Kaufman

English 111

Professor Hixon

4 October 2017

Database Difference Checker (DBC)

By Peter Kaufman

Abstract

Many organizations collect data for later use. This data can be stored in many different places, one of which is a MySQL database. There is often one that is used for a development (dev) environment and one for a live environment. As time goes on, the dev environment and database change. As a result, the live database needs to be updated for the environment to run the same as the dev environment. In order to make the live database the same as the dev database, DBC was created to compare the two databases and determine the SQL statements to make them the same. The database comparison works with column, table, view, and index differences. The DBC can also work with updating a database when the dev database cannot be connected to. This is done by taking a database (DB) snapshot, which stores the structure of a database schema in a JSON file (Aaron).

Introduction

The DBC consists of twelve Java classes and JFrames and determines which columns, indices, and tables need to be added and dropped. The DBC can compare two databases using the MySQL username, password, host, port, and database name (Figure 1); take a DB snapshot (Figure 2); and compare a database to a DB snapshot (Aaron)(Figure 3).

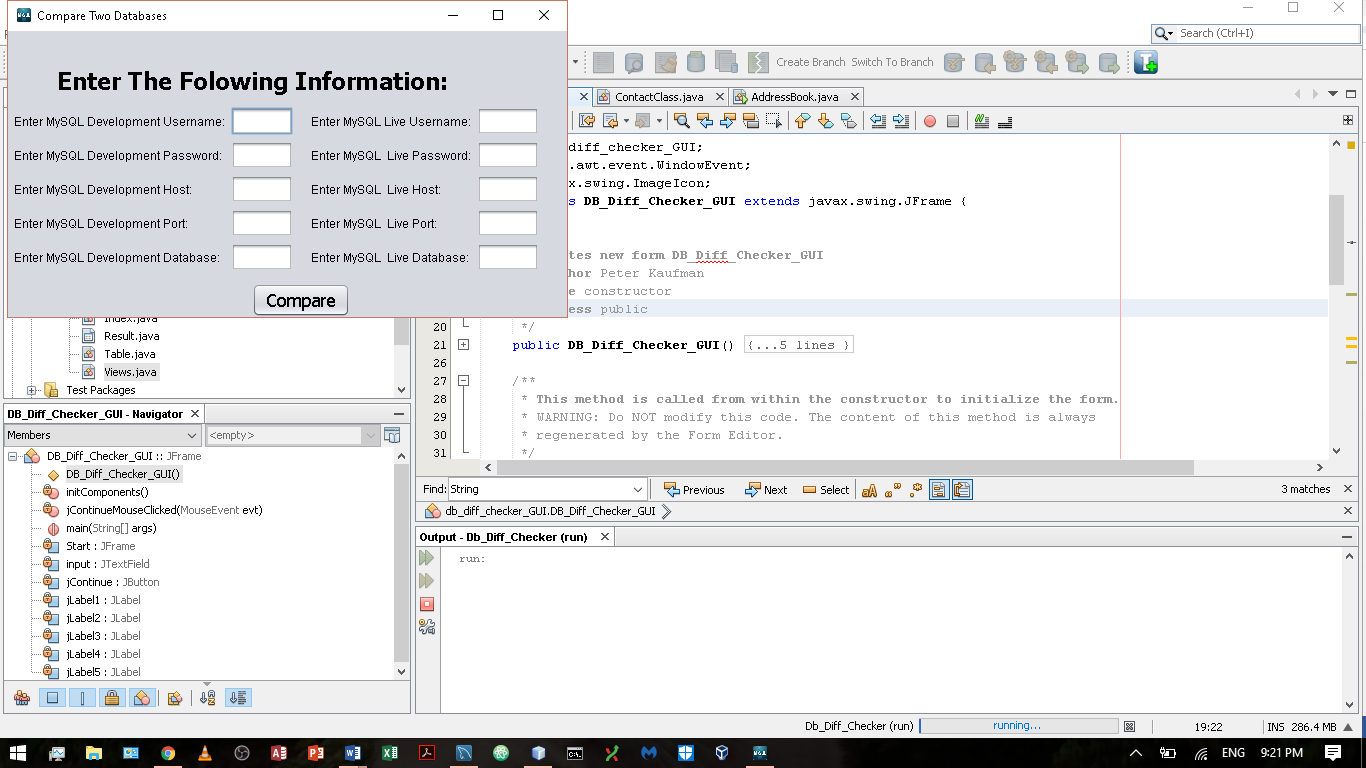


Figure 1

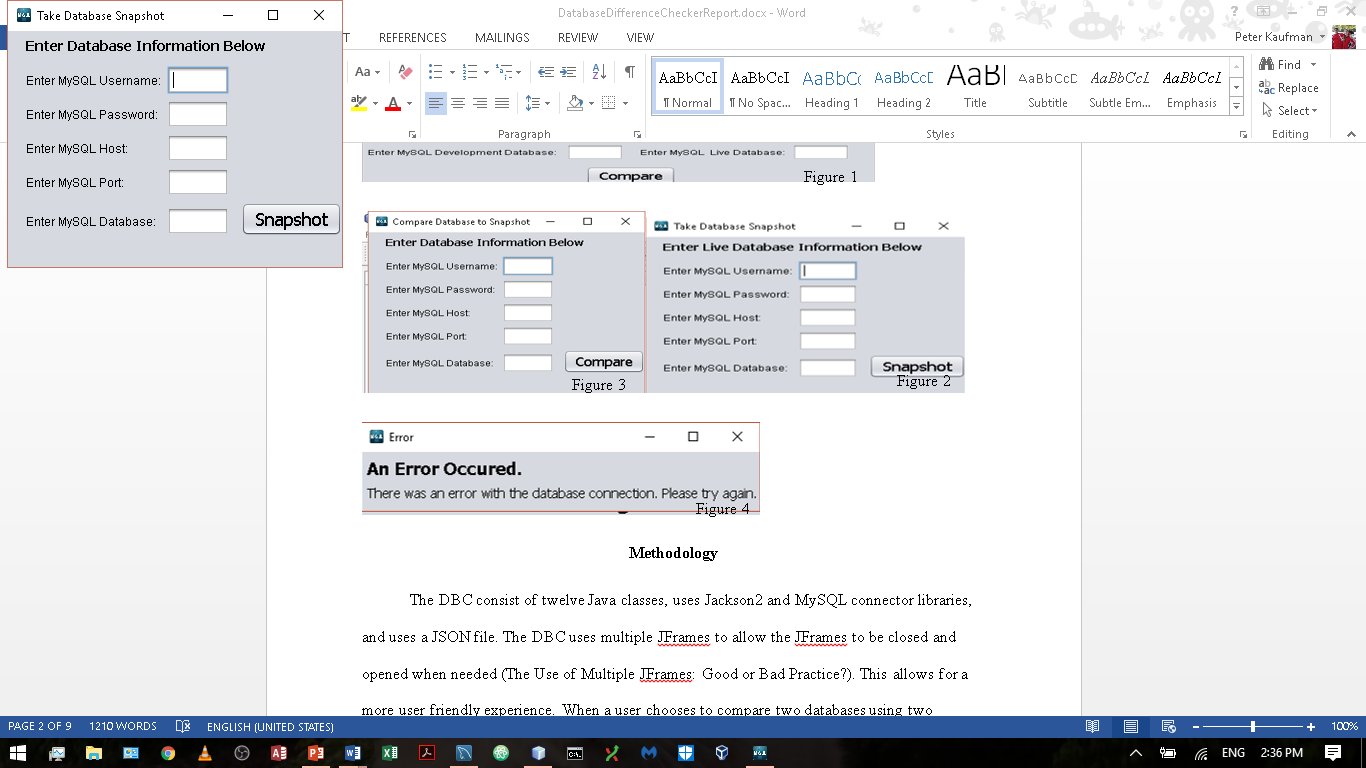
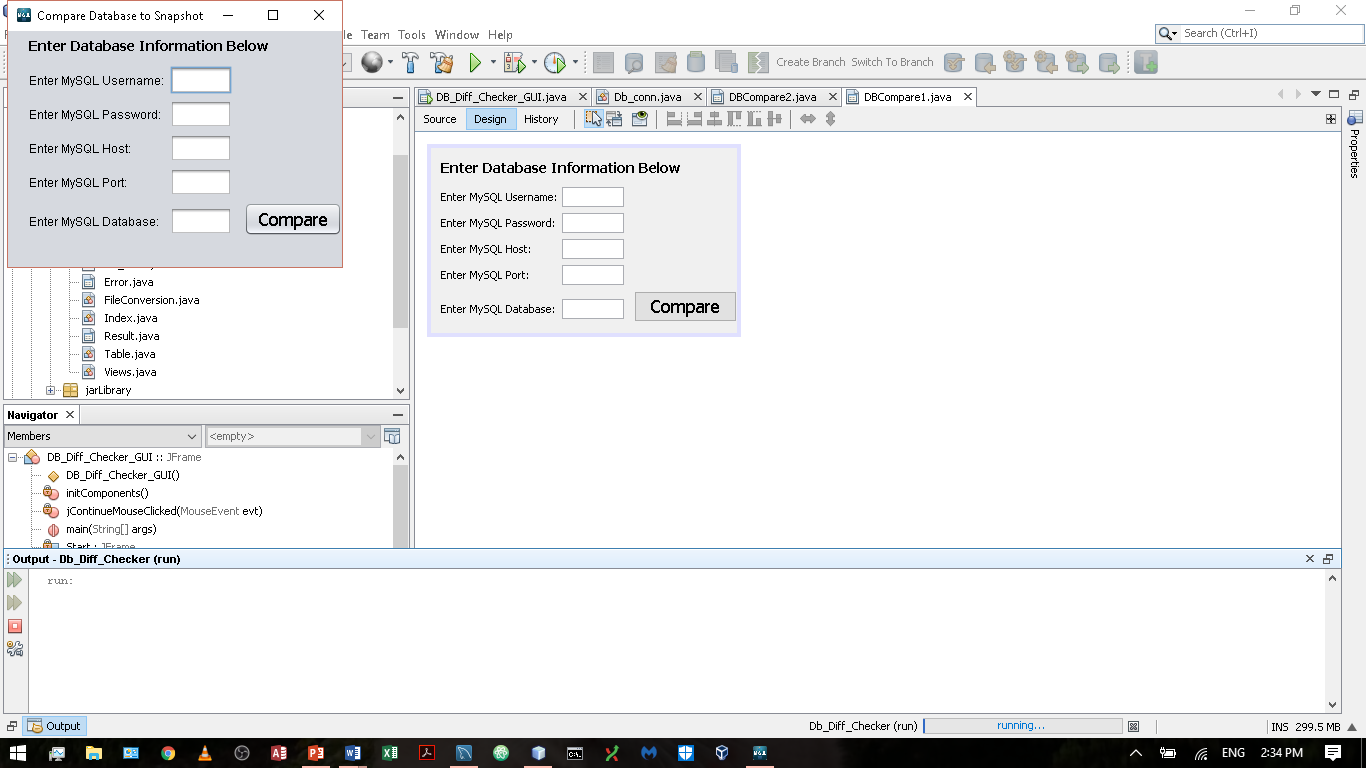


Figure 2

Figure 3

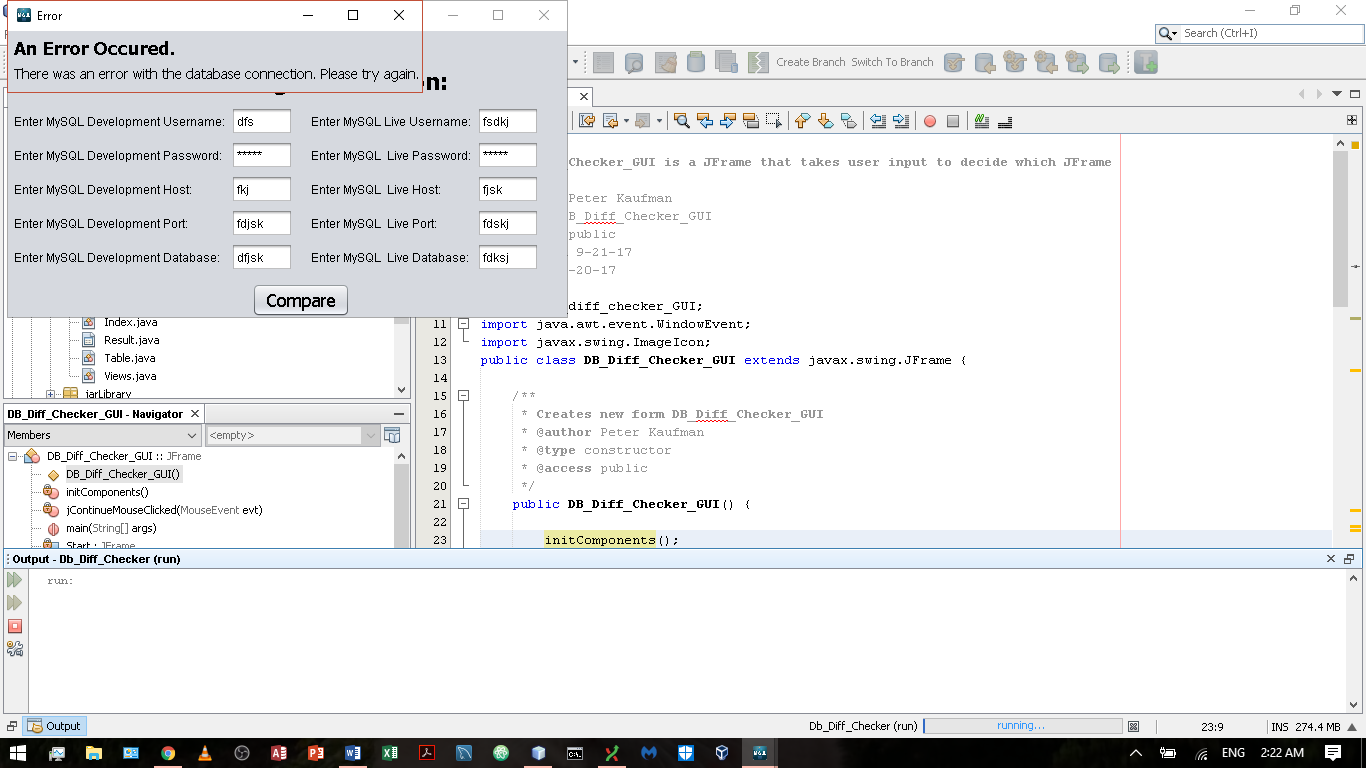


Figure 4

Methodology

The DBC consist of twelve Java classes, uses Jackson2 and MySQL connector libraries, and uses a JSON file. The DBC uses multiple JFrames to allow the JFrames to be closed and opened when needed (The Use of Multiple JFrames: Good or Bad Practice?). This allows for a more user friendly experience. When a user chooses to compare two databases using two database connections, a new window pops for the user to input information to connect to the dev and live databases (Figure 1). Once the user inputs the necessary information, a database connection is made for each of the databases. For each database, the connection is tested. If the connection is unable to be established, then an error window will pop up and inform the user (Figure 4). If the connection is established, the views of the database are found using an SQL statement. In the process, each view’s create statement is collected and added to the view ArrayList. Next, all of the tables, columns, and indices of the database are found. This information is then added by sorting the results by table name and column order. This allows Table objects to be made and its columns and indices to be added to the Table object. Collecting the index, table, and column information in one shot makes the program faster because there is no need to run several smaller queries to do what one larger query can do in one shot. After the Database objects have been initialized, the two Database objects are compared. First off, an ArrayList of Table objects from each Database object is compared to look for missing and extra tables. If any extra or missing tables are found, the appropriate SQL statement(s) is/are generated to make the table list the same. These tables are added to a list of tables to exclude from the rest of the comparisons. Second, a list of tables, which are to be updated, are compiled based on whether or not the create statements of a table with the same name in each database are the same. If any difference in the create statements is found, then the table name is added to the list. Third, this set of tables is then used to update these tables. The table information is compared column by column and index by index. If any are found to be extra or missing, the appropriate SQL statements are generated. Last, all of the views in the live database have their drop statements generated, and all the views in the dev database have their add statements generated. After each of these four comparisons, the SQL statements are added to a “master list.” This list is displayed in a new JFrame where the user can copy the code in order to run it elsewhere. If the user chooses to do a database comparison with a DB snapshot, the process is the same except that a JSON file must be converted back into a Database object before the comparisons occur. Furthermore, if the user chooses to take a DB snapshot, a Database object is initialized, and then it is converted to a JSON file, which can be converted back to a Database object as desired later. If an error occurs at any time, a new JFrame will appear with an error message related to why the error occurred. Listed below are the most important methods of the DBC.

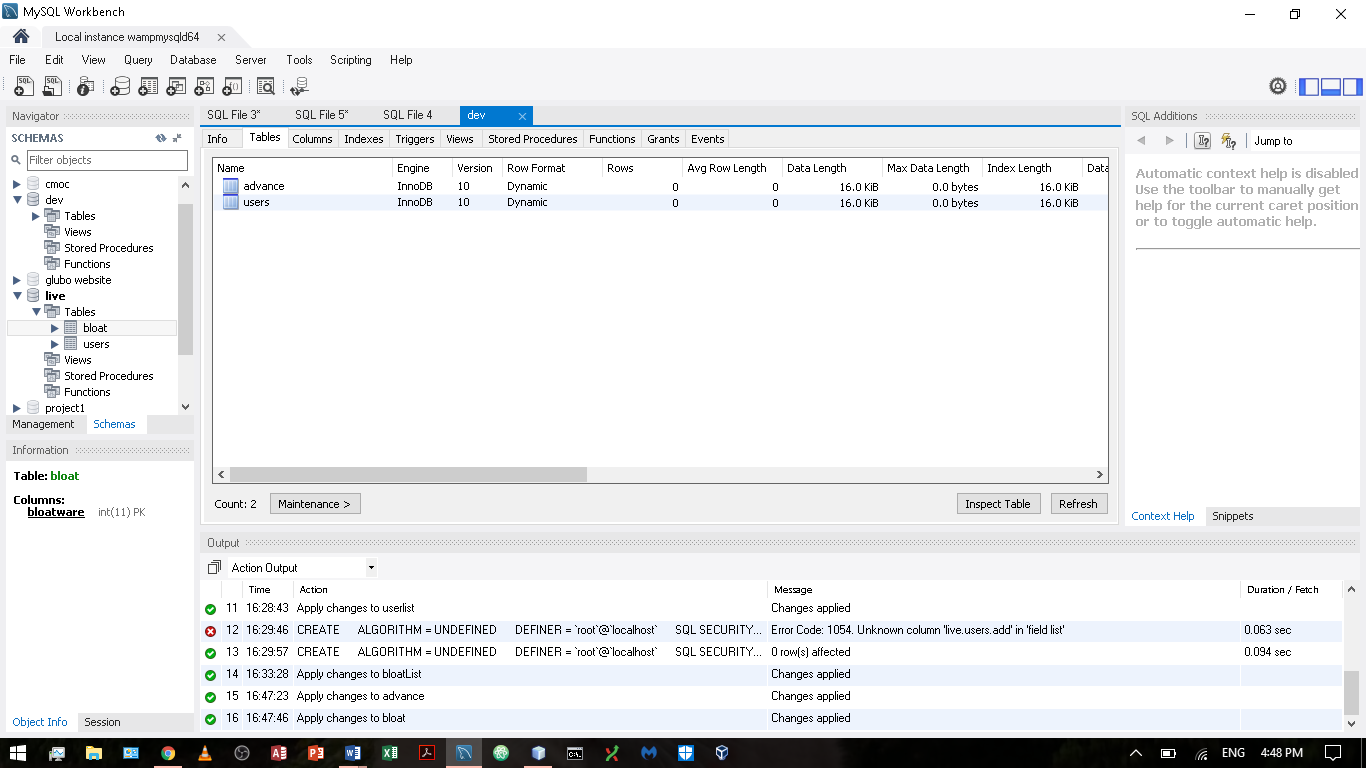
|  |  |  |
| --- | --- | --- |
| Class | Method | Description |
| DB\_Diff\_Checker\_GUI | jContinueMouseClicked | Determines which method the user has selected and opens the appropriate JFrame |
| DBCompare1 | compare2 | Compares a database to a database snapshot |
| DBCompare1 | DB1btnActionPerformed | Determines whether to take a database snapshot or compare a database to a database snapshot based on the JFrame’s title |
| DBCompare1 | takeSnapshot | Takes a snapshot of a database by converting a Database object to a JSON file |
| DBCompare2 | jButton1ActionPerformed | Determines whether the information supplied by the user is adequate. If so 2 databases are compared otherwise a message is displayed |
| DBCompare1/DBCompare2 | displayResult | Opens a new JFrame which displays the SQL statements to be run to make the dev and live database the same |
| DBCompare1/DBCompare2/Db\_conn | Error | Opens a new JFrame which displays the error that occurred |
| Database | tablesDiffs | Updates the list of tables which are not the same in dev |
| Table | Equals | Takes in a Table and compares it to the current one, the result is the SQL statements to be run to make the two tables the same |
| FileConversion | writeTo | Turns a Database object into a JSON file |
| FileConversion | readFrom | Turns a JSON file into a Database object |
| Db\_conn | getTableList | Gets the tables, columns, and indices of the db |
| Db\_conn | getViews | Gets the views of the db |

Result & Discussion

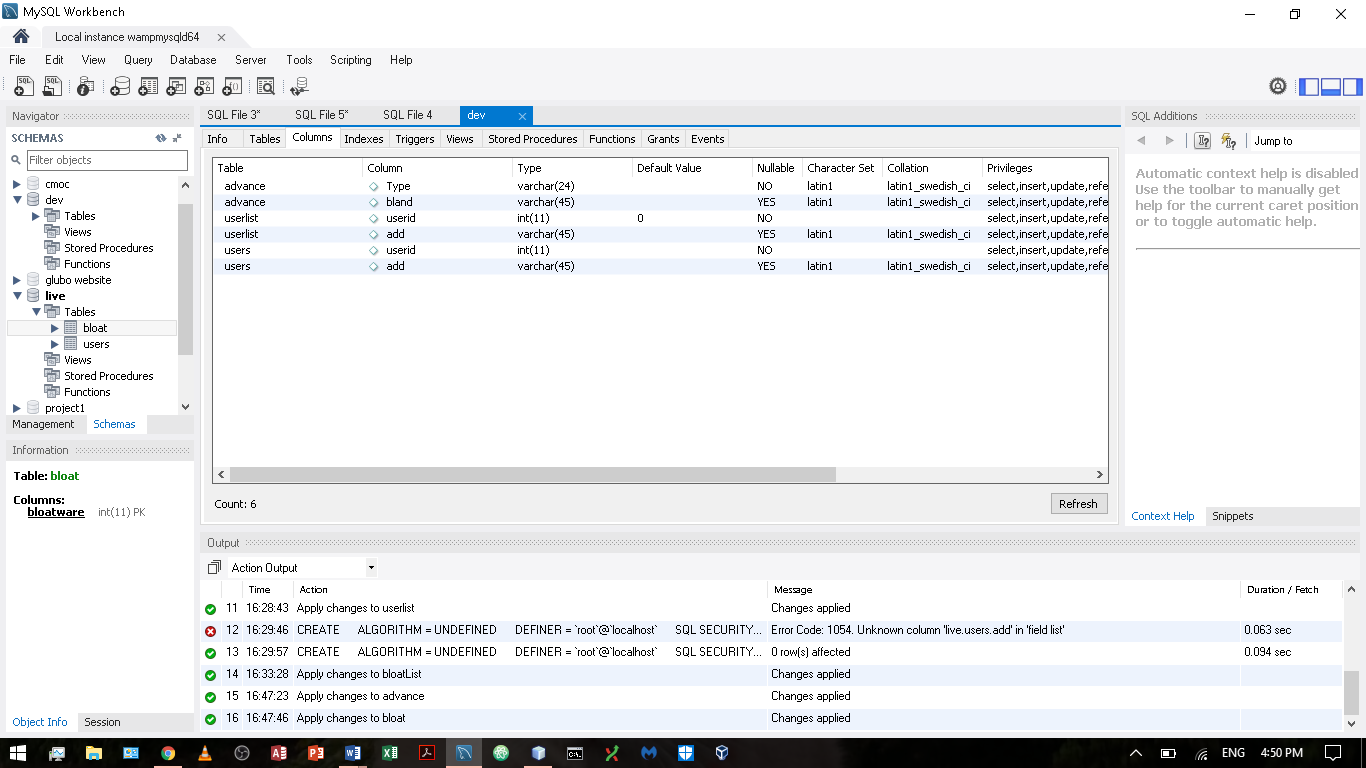
Say there are two databases one called live and one called dev as shown below (Dev, Live). If the DBC is run to make the live database the same as the dev database, the result is shown below (DBC Result). When run, these SQL statements make the live database the same as the dev database. After running the above code, the result of running the DBC again is shown below (DBC Result2). The views are the only ones that show up because any views from the live database are automatically dropped and added regardless of whether or not they are different.

Dev:

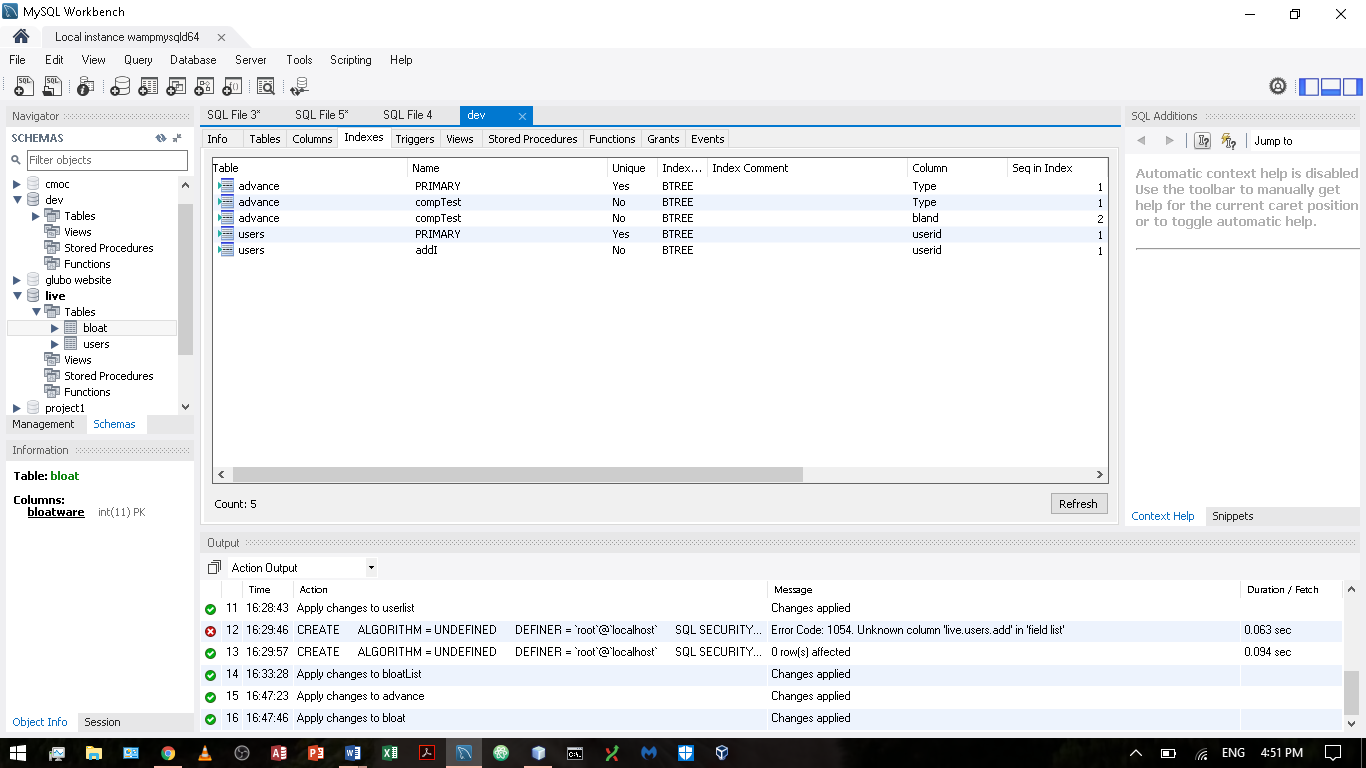
Tables



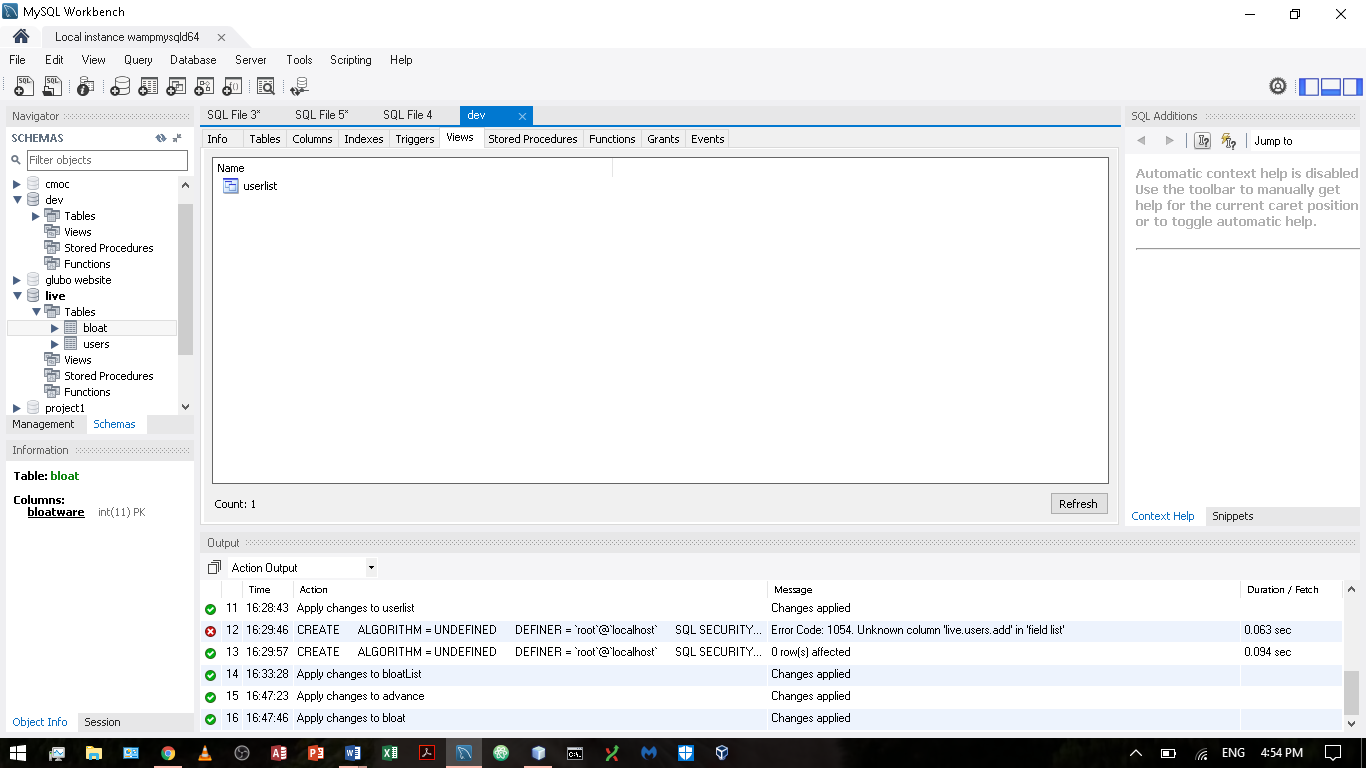
Columns



Indices

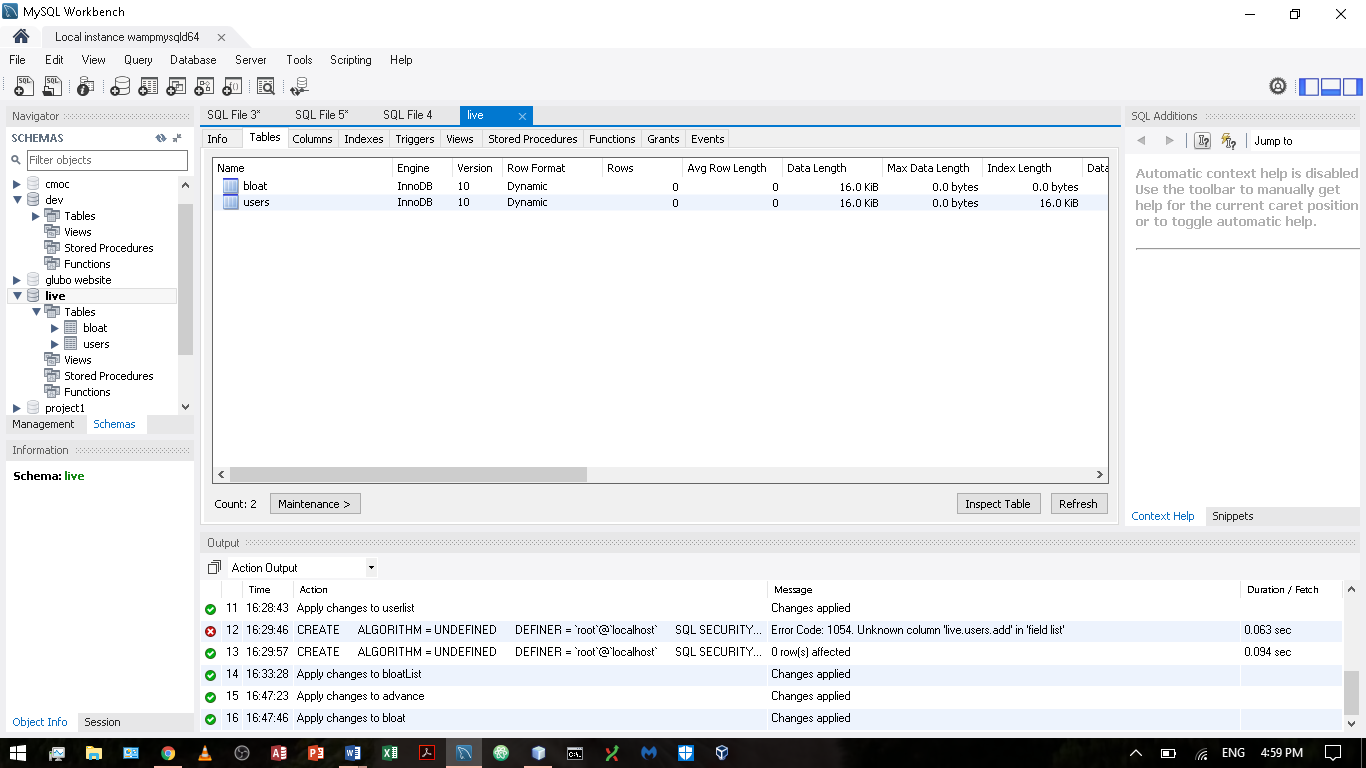


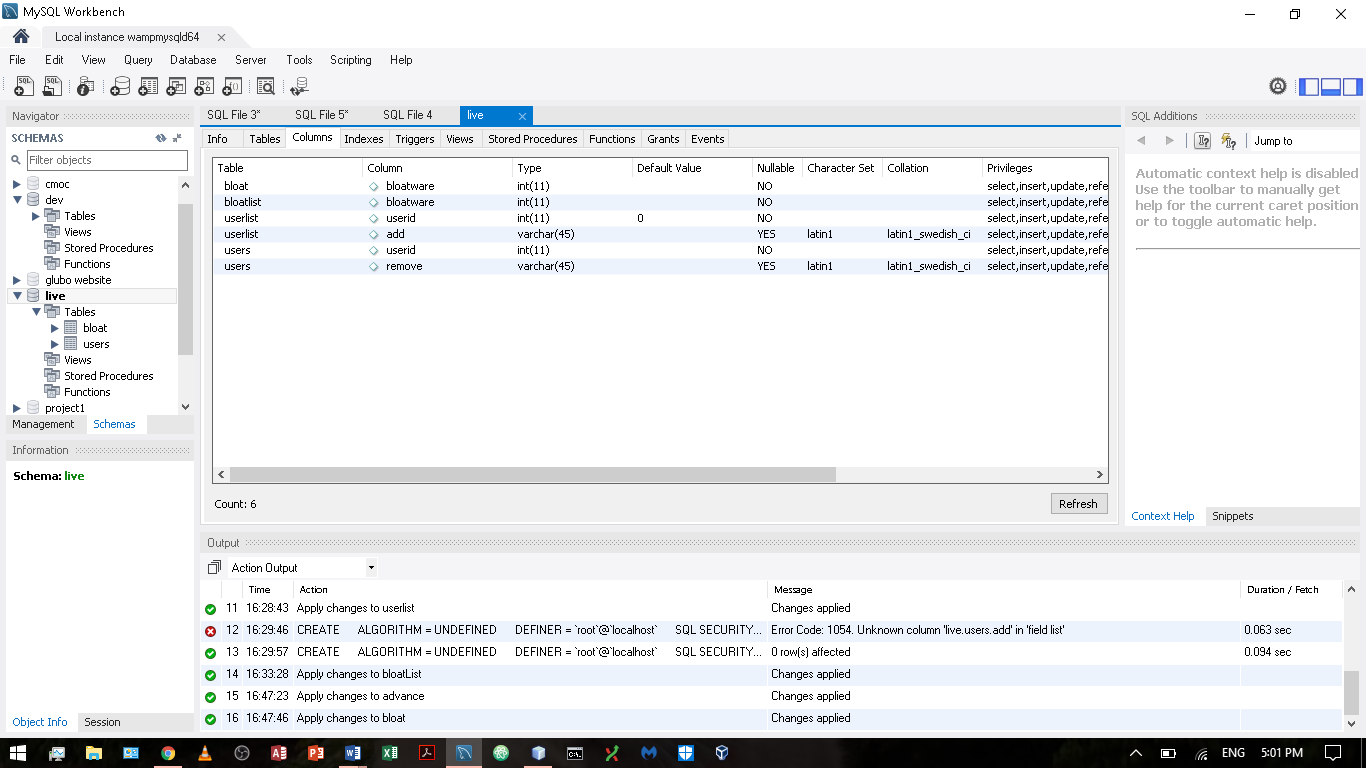
Views



Live:

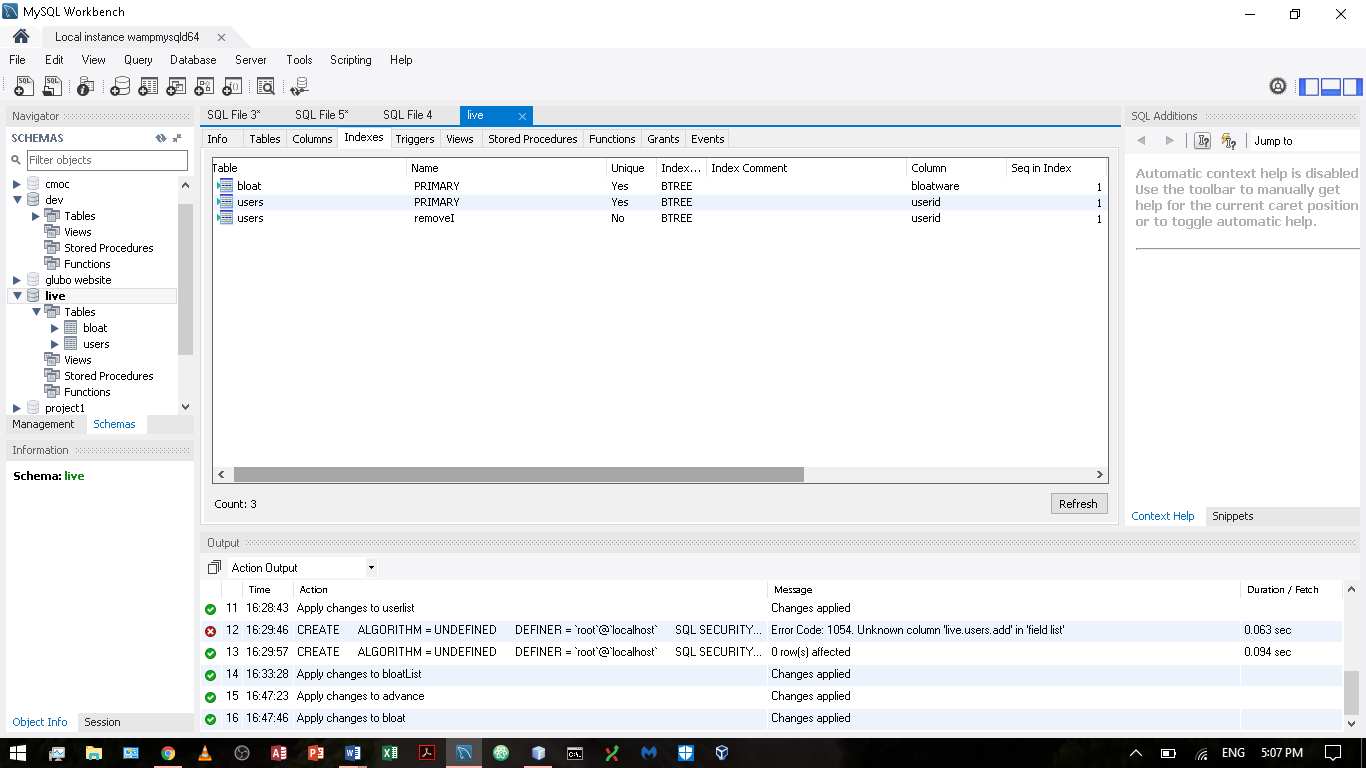
Tables



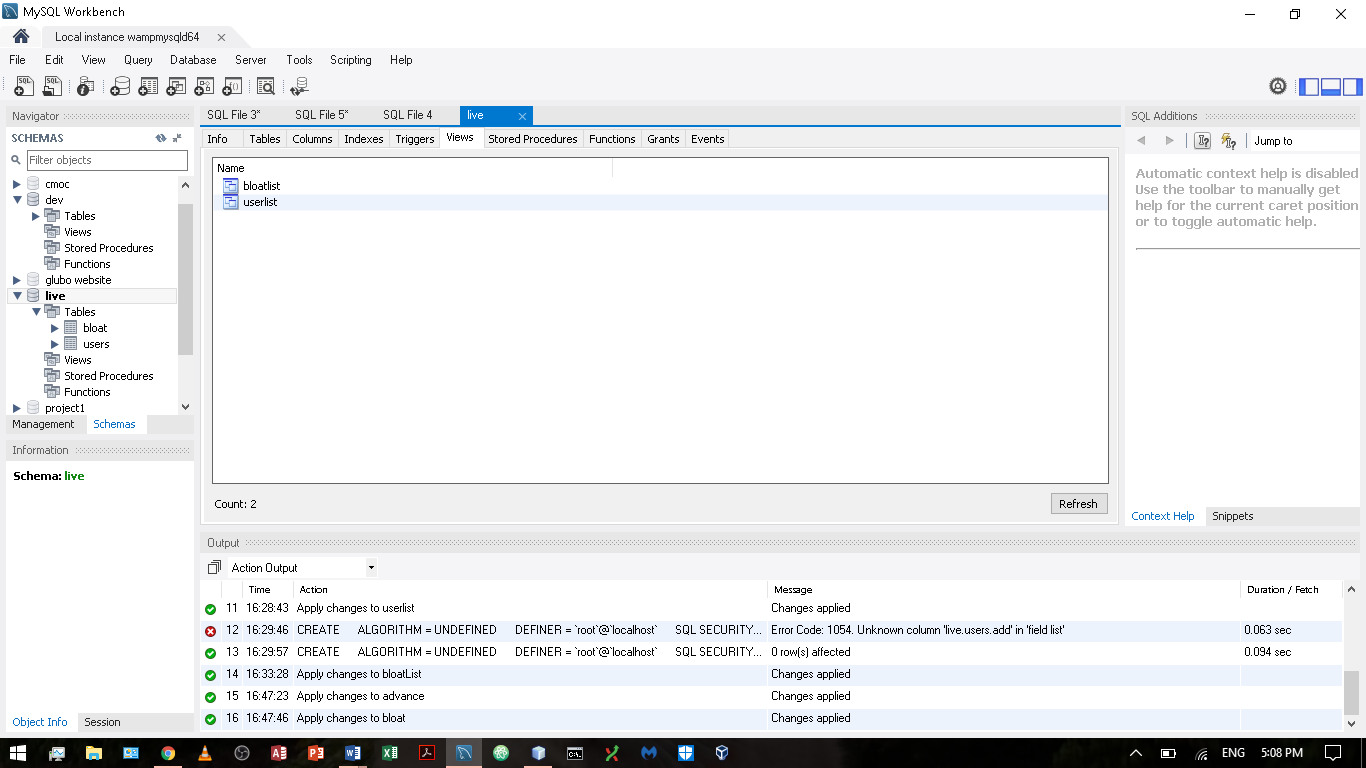


Indices

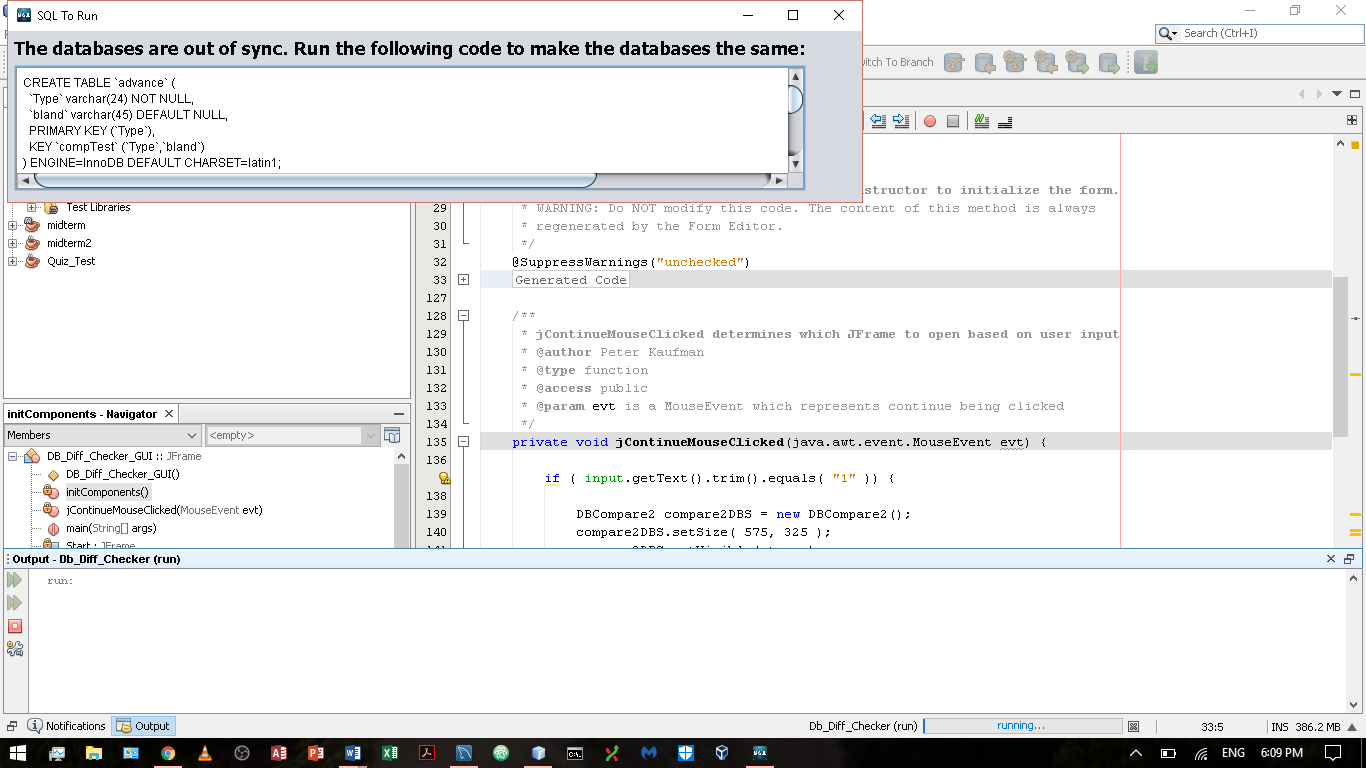
Columns

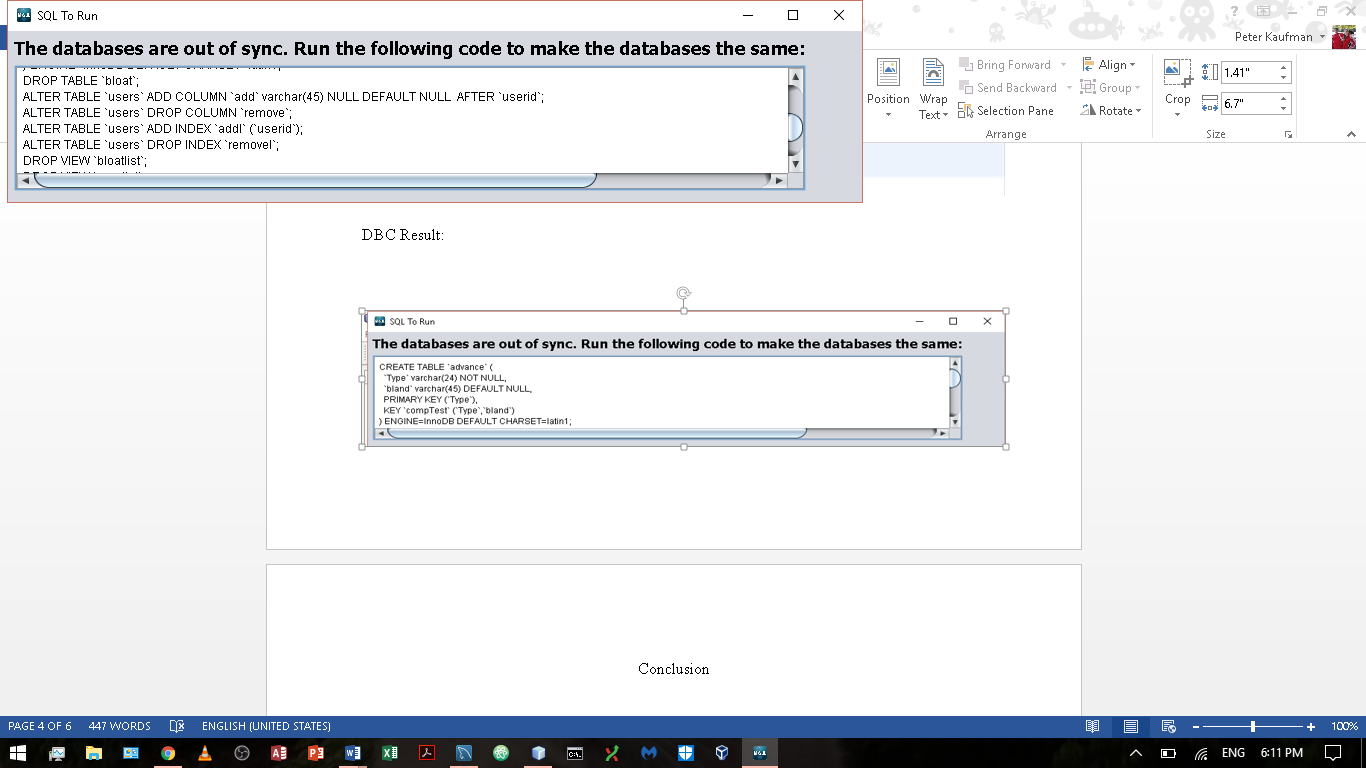


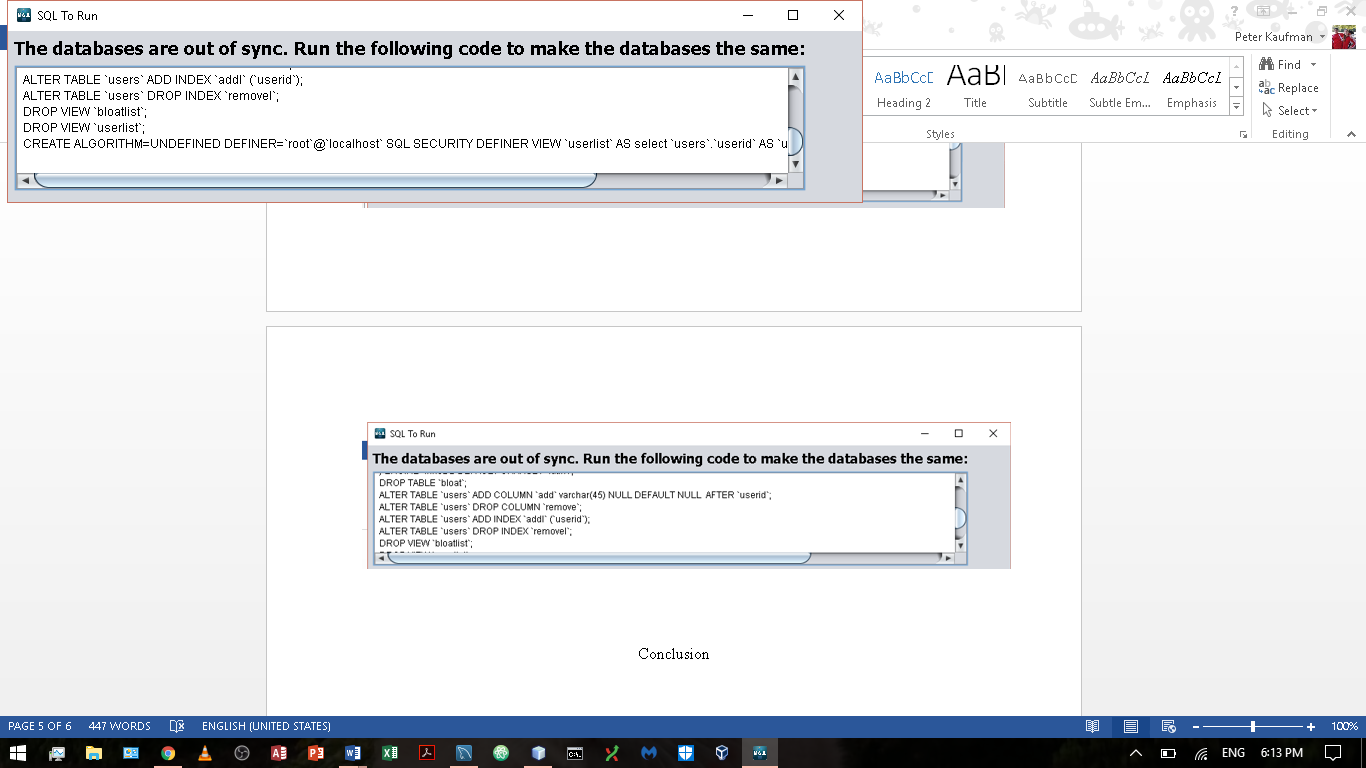
Views



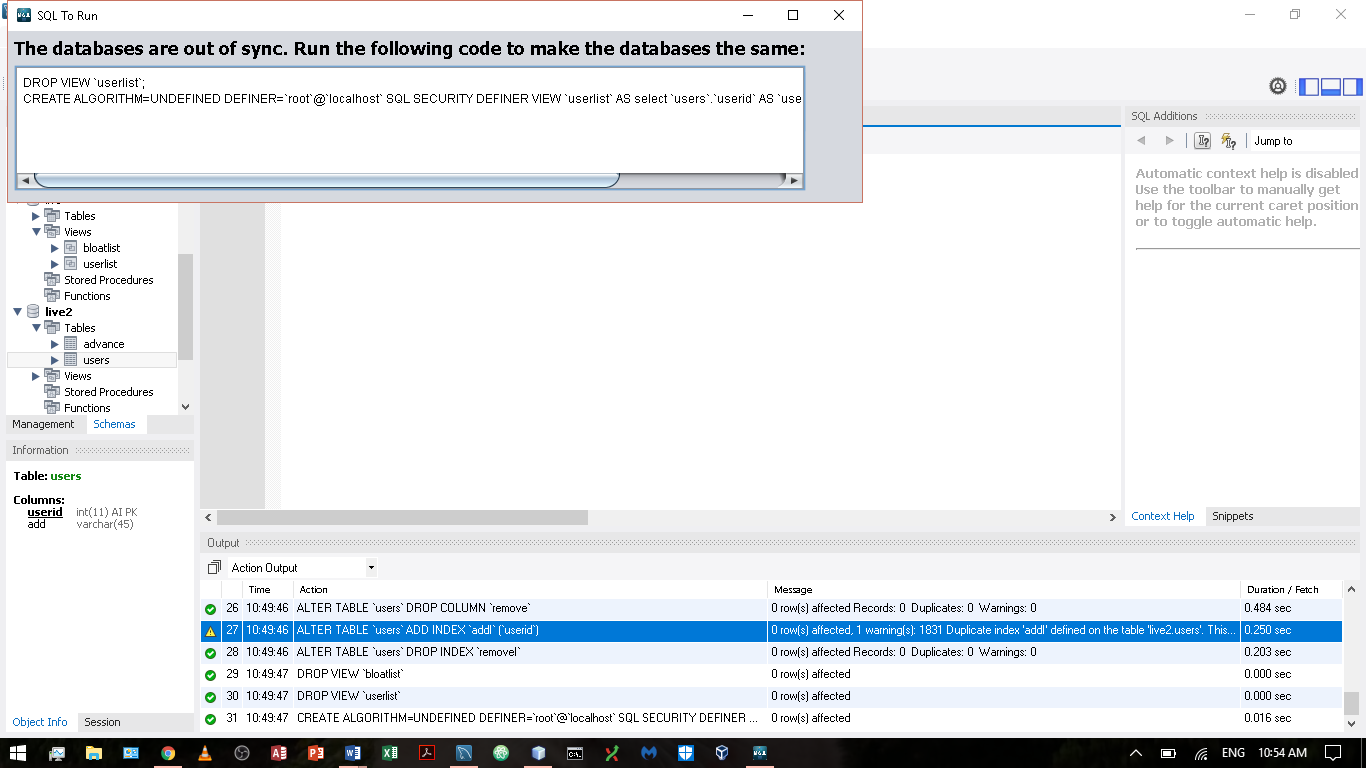
DBC Result:







DBC Result2:



Conclusion

The DBC makes comparing two databases’ tables, columns, indices, and views easy regardless of whether or not the comparison is made using a DB snapshot or two database connections (Aaron). Running the DBC generates the SQL statements, which make two databases the same. The DBC can be used when updating software and bringing a live database up to speed with a dev database.

Works Cited:

Aaron, Rance. Personal Interview. 1 June 2017 – 11 Aug. 2017.

“The Use of Multiple JFrames: Good or Bad Practice?,” *Stack Overflow*, Stack Exchange Inc, 4 Mar. 2012, stackoverflow.com/questions/9554636/the-use-of-multiple-jframes-good-or-bad-practice.