Programming Language Research Assignment 3

Algonquin College
School of Advanced Technology
Computer Programming

Onur Önel
Mazin Abou-Seido

23F_CST8333_360

Submitted November 03, 2023

A technical report submitted to Algonquin College in partial fulfillment of the requirements for a diploma in Computer Programming

Database Integration

The primary aim is uplift the operational capabilities of my existing C++ project. The planned transition is from the current state, which relies on a third-party API https://github.com/vincentlaucsb/csv-parser for CSV file parsing, to a more robust and sophisticated mode of operation involving direct interactions with a MySQL database. This transition is underlying strategic move which is using advanced and comprehensive functionalities that SQL databases offer. By adopting this database-centric approach, I aim several key aspects of the project:

- **Efficiency:** SQL databases are optimized for performance, with well-established methods for data indexing, searching, and retrieval. By refactoring our project to utilize a MySQL database, expecting improvement in the speed and efficiency.

- **Scalability:** If the demands on my project increase, it's essential that the data management system can scale accordingly. MySQL databases are adept at managing a larger volume of data without a significant increase in resource usage. This capability ensures that I can enhance the project's capacity with minimal issues.

- **Performance:** MySQL knowns with advance query optimization and caching. These features improve data manipulation and retrieval, leading to a more efficient and faster-performing application.

- **Reliability:** MySQL offers strong data integrity safeguards such as atomic transactions, which ensure that data operations are completed fully or not at all, consistent state maintenance, and mechanisms for recovering from crashes. Incorporating these aspects bolsters the dependability of the project's data management.

- **Security:** Adopting a MySQL database enhances security provisions, incorporating advanced access controls, encryption, and mechanisms for secure data transmission. These features are indispensable for defending sensitive information against unauthorized access and potential security breaches.

1. https://www.jobsity.com/blog/5-reasons-why-mysql-is-still-the-go-to-database-management-system
2. https://www.mysql.com/industry/faq/
3. https://www.javatpoint.com/mysql-features

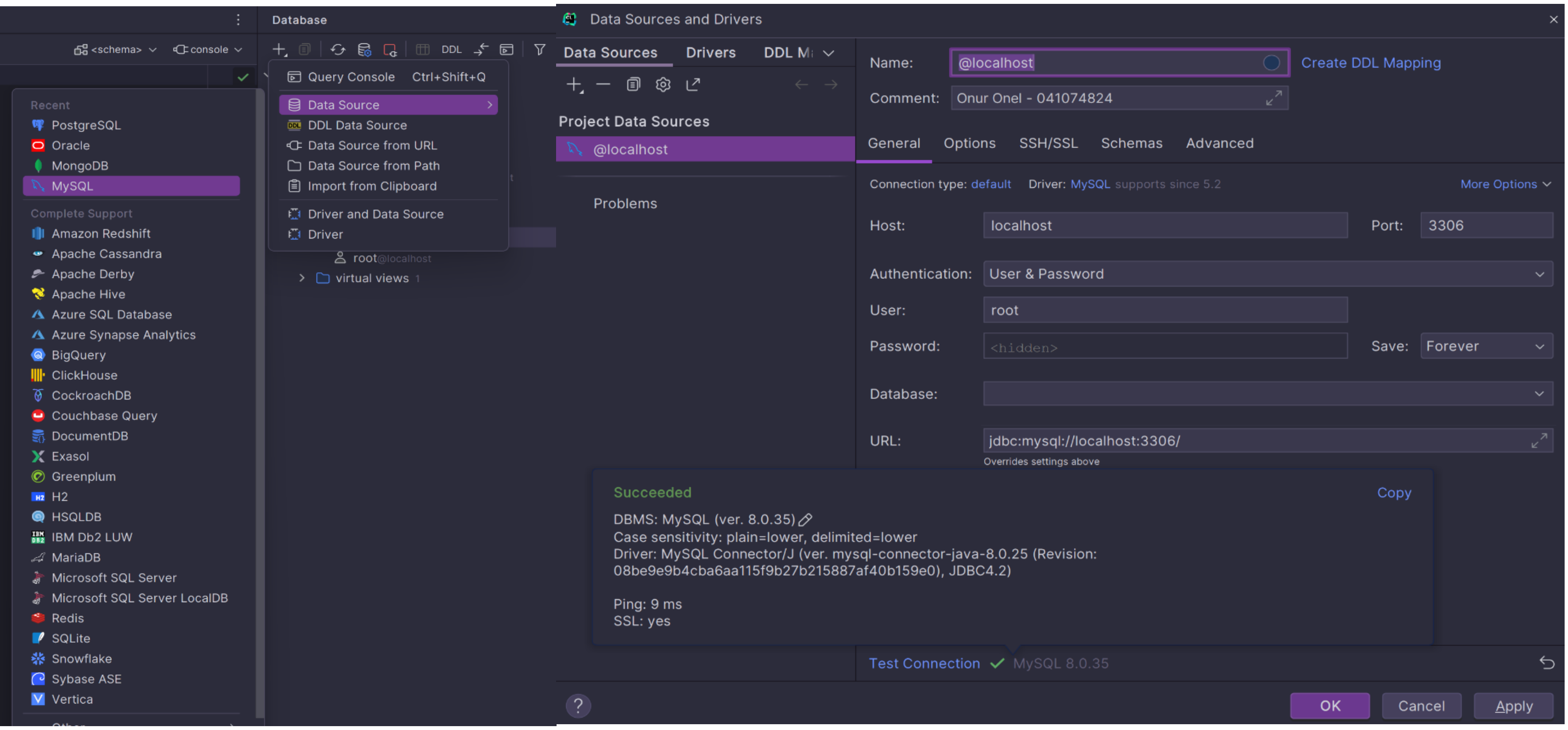
Refactoring

I will start by examining the existing codebase, pinpointing code blocks interfacing with the CSV files through CSV Parser. The objective is implementing interactions to a MySQL based workflow, involving the following tasks:

1. Constructing 'DatabaseHandler' class to encapsulate basic database connection protocols.
2. Improve 'DatabaseHandler' with comprehensive CRUD (Create, Read, Update, Delete) operations.
3. Revising the 'addRecord' method for database interaction.
4. Transition 'deleteRecordById' to employ database commands
5. Changing the 'updateRecord' function to modify data within the database
6. Modifying 'readRecord' to retrieve data via SQL queries
7. Remove old file I/O code to streamline the codebase

Integration of MySQL within CLion

The integration includes adding MySQL client library into the CLion project which requires on an update on compiler through the CMakeLists.txt file. This setup establishes linkage between the project and the MySQL server while ensuring best practices for secure connections.



Name: @localhost

Comment: Onur Onel - 041074824

Connection type: default Driver: MySQL supports since 5.2

Host: localhost Port: 3306

Authentication: User & Password

User: root

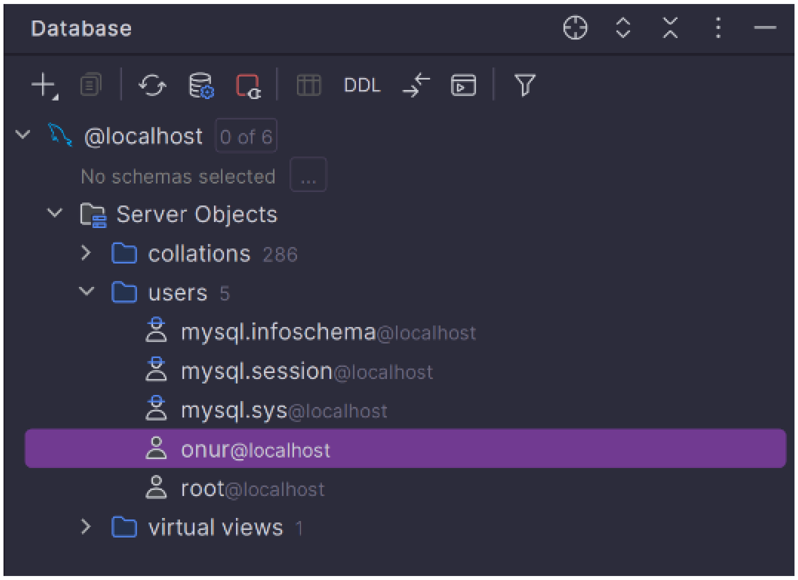
Password: <hidden> Save: Forever

Database:

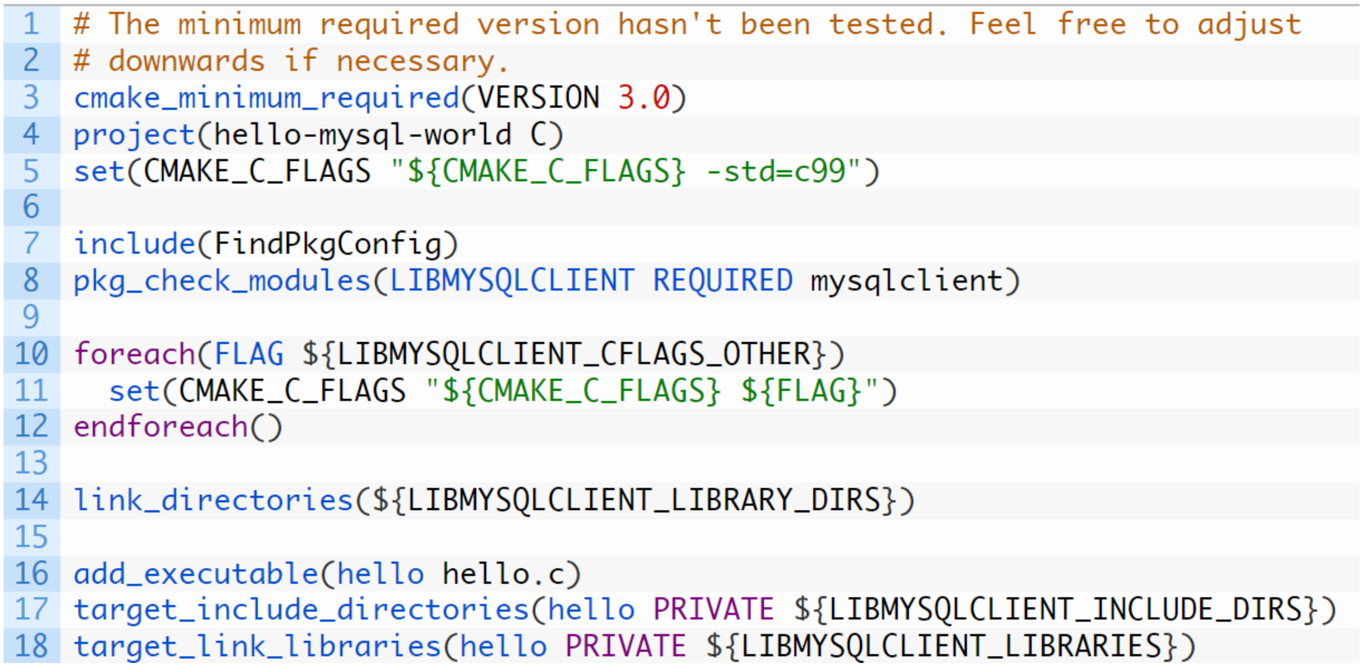
URL: jdbc:mysql://localhost:3306/

```
Succeeded
DBMS: MySQL (ver. 8.0.35)
Case sensitivity: plain=lower, delimited=lower
Driver: MySQL Connector/J (ver. mysql-connector-java-8.0.25 (Revision:
08be9e9b4cba6aa115f9b27b215887af40b159e0), JDBC4.2)

Ping: 9 ms
SSL: yes
```

Test Connection ✓ MySQL 8.0.35



I set up the client library for MySQL. My machine now contains all the required files and directories that compiler needs. Next, I am going to make changes to the CMakeLists.txt file. Here, I tell the compiler to link with the MySQL client library and describe the location of the MySQL inclusion files through commands such as find_package, target_include_directories, and target_link_libraries Finally, I need to adjust the compiler parameters. This step guarantees that MySQL is operating securely and effectively running.



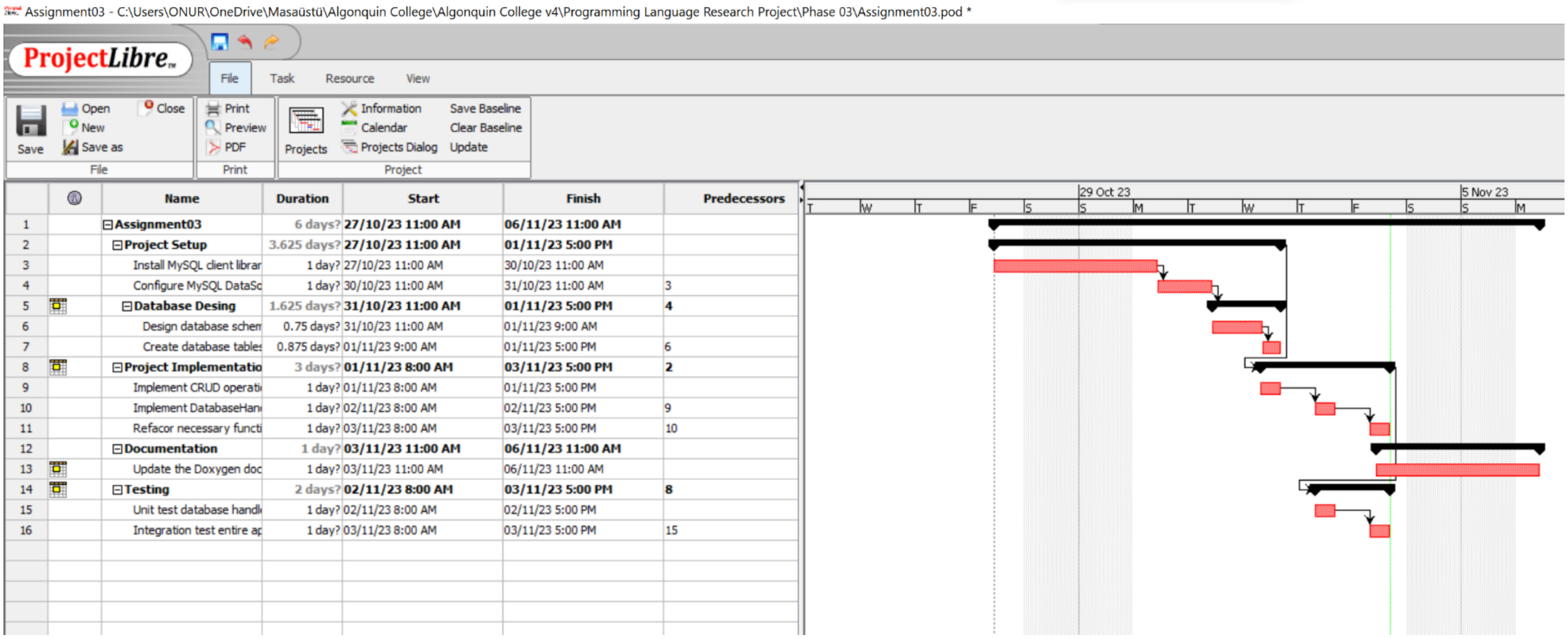
```
# The minimum required version hasn't been tested. Feel free to adjust
# downwards if necessary.
cmake_minimum_required(VERSION 3.0)
project(hello-mysql-world C)
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -std=c99")

include(FindPkgConfig)
pkg_check_modules(LIBMYSQLCLIENT REQUIRED mysqlclient)

foreach(FLAG ${LIBMYSQLCLIENT_CFLAGS_OTHER})
  set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} ${FLAG}")
endforeach()

link_directories(${LIBMYSQLCLIENT_LIBRARY_DIRS})

add_executable(hello hello.c)
target_include_directories(hello PRIVATE ${LIBMYSQLCLIENT_INCLUDE_DIRS})
target_link_libraries(hello PRIVATE ${LIBMYSQLCLIENT_LIBRARIES})
```

https://dev.mysql.com/blog-archive/the-client-library-part-4-how-to-write-a-simple-mysql-client-in-c-using-cmake-and-pkg-config/

Shifting from CSV files to SQL represents a significant change in how I manage data. Instead of processing files in sequence, I'll be employing SQL queries, which are not only quicker but also offers handling of complex data relationships. Confirming that the new system is reliable and functions correctly is critical. To achieve this, I will perform comprehensive testing using the CATCH2 framework, which has proven effective in previous assignments.

In conclusion, switching to a MySQL database will bring immediate enhancements in our project's data management and pave the way for ongoing improvements. This update is a deliberate move to keep the project up-to-date and scalable, readying it for future technological upgrades.

WBS & Gant Chart



	ⓘ	Name	Duration	Start	Finish	Predecessors
1		⊟Assignment03	6 days?	27/10/23 11:00 AM	06/11/23 11:00 AM	
2		⊟Project Setup	3.625 days?	27/10/23 11:00 AM	01/11/23 5:00 PM	
3		Install MySQL client librar	1 day?	27/10/23 11:00 AM	30/10/23 11:00 AM	
4		Configure MySQL DataSc	1 day?	30/10/23 11:00 AM	31/10/23 11:00 AM	3
5	▦	⊟Database Desing	1.625 days?	31/10/23 11:00 AM	01/11/23 5:00 PM	4
6		Design database schem	0.75 days?	31/10/23 11:00 AM	01/11/23 9:00 AM	
7		Create database tables	0.875 days?	01/11/23 9:00 AM	01/11/23 5:00 PM	6
8	▦	⊟Project Implementatio	3 days?	01/11/23 8:00 AM	03/11/23 5:00 PM	2
9		Implement CRUD operati	1 day?	01/11/23 8:00 AM	01/11/23 5:00 PM	
10		Implement DatabaseHan	1 day?	02/11/23 8:00 AM	02/11/23 5:00 PM	9
11		Refacor necessary functi	1 day?	03/11/23 8:00 AM	03/11/23 5:00 PM	10
12		⊟Documentation	1 day?	03/11/23 11:00 AM	06/11/23 11:00 AM	
13	▦	Update the Doxygen doc	1 day?	03/11/23 11:00 AM	06/11/23 11:00 AM	
14	▦	⊟Testing	2 days?	02/11/23 8:00 AM	03/11/23 5:00 PM	8
15		Unit test database handl	1 day?	02/11/23 8:00 AM	02/11/23 5:00 PM	
16		Integration test entire ap	1 day?	03/11/23 8:00 AM	03/11/23 5:00 PM	15

https://www.projectlibre.com/

After a BIOS reset on my laptop, I made the switch from MS Project to ProjectLibre for my Assignment03. ProjectLibre, being a Java-based application with its familiar user interface and feature set that's quite similar to MS Project's. Using ProjectLibre, I crafted a WBS Gantt chart that detailed project tasks, their dependencies, and timelines. The chart included new features and connected tasks in a way that made the project flow and deadlines clear.