SWE 265P - REVERSE ENGINEERING

H2 Database
HOMEWORK 1 REPORT

TEAM RUNTIME TERROR

Changes

1. We added more details into the introduction like the features of the system and build information as we felt this would be a good start in understanding the system which was not present in the earlier version
2. We have included the complete UML diagram of the system - both a compressed image file and a UML file generated using the simpleUML plugin on IntelliJ.
3. We have updated the descriptions for both the features in order to add specific details about how we found the implementation of those features

Introduction

H2 is an embedded relational database management system(RDBMS) which is written in Java. It is a browser-based console application Some of the main features of this database are [1]:

1. It uses a very fast open-source JDBC API
2. It can be operated in embedded and server modes. Embedded mode is used when we don't need a distributed environment or we use server mode otherwise.

The project uses Maven as the build tool. The project makes use of JDK 1.8. All the other versions of Java are not supported currently since they do not have tools.jar.

[1]Source: https://www.h2database.com/html/main.html

Selected Features

1. Create Table

All relational databases store their data in tables, which are made up of rows and columns. Every column is an attribute, and the rows represent different datasets. The system that we chose is complex, and thus through this and the next feature, we wanted to understand the system.

First, we started off by searching for the string “createtable” in the codebase using the Find in Path functionality on IntelliJ. This returned a lot of results, the most important one being the CreateTable class as it represents the CREATE TABLE command. CreateTable class has a property of type CreateTableData which represents the data that is used to create the table like schema, column names, table name, whether table would be temporary or global, should be persisted or not, etc. In the CreateTable class, we searched for the usages of createTable using the Find functionality as we believed there should be a reference to some method that actually creates the table. Then we found the update method in this class which calls the createTable method in the Schema class. We then traced forward by going to the implementation of this method in the Schema class. The createTable method in the Schema class, in turn, calls the createTable method and on looking for the implementation we found its implementation in the MVTableEngine class. On tracing ahead, we found that this method then calls the createTable method in Store class which is an inner class in MVTableEngine class. The Store class maintains a Concurrent HashMap of table names and table objects. In the createTable method of the Store class, we create an instance of the Table object and store the table in the ConcurrentHashMap and then return the table. Now an object table of type MVTable is returned through a return hierarchy to the initial CreateTable class thus creating a table.

2. Setting User privileges

All database must ensure secure access to them, and prevent access to administrative tasks. The administrative tasks like dropping schemas, and creating schemas, must only be performed by the admin. This feature would help us identify where in the code, does the setting of privileges takes place.

First, we started off by searching for the string “user” in the codebase using the Find in Path feature on IntelliJ because we believed all the user related meta would be present there. We found the User class which has a setAdmin method. The User class represents each User of the h2 database. For example, User A could be an admin and have master privileges whereas the other users might just have select query privileges.

We then checked for the usages of setAdmin to better understand where the admin privileges are being set for the user. On doing FindUsages on “setAdmin”, we got three potential matches, and we started looking into them. The openSession method in Engine class was our first choice. We chose this first, because every time the h2 database is fired up, openSession would create a session for the logged user, and thus it would also check the privileges of that user, in order to set his grants and rights. There we found out if the database is not created or is new, the current user would be granted admin privileges(since the database is new, and an admin would only be the one that is responsible for creating a database). And this user will also be set as the master user for the database in consideration.

The next place where we looked at was CreateUser (which represents the statement CREATE USER), since every time a User is created, their privileges must also be set so that if they try to access something which is not permitted, h2 would be able to identify it, and not grant them access. The parsing of this statement takes place in a file called Parser.java in which we create an instance of CreateUser. Now based on the parsed tokens if we find an admin token, then we use the setAdmin method in CreateUser with a TRUE parameter making the user admin to the database.

Lastly, we looked at AlterUser class as it represents the ALTER USER command. Here we found out that the Admin flag of a user can be toggled between admin and non-admin. Following which we update the metadata for the particular user.