CSCI 5408

DATA MANAGEMENT AND WAREHOUSING

SPRINT -3 DOCUMENT

Group-3
Jay Alpeshkumar Patel (B00969013)
Manish Shankar Jadhav (B00969328)

GitLab Link: https://git.cs.dal.ca/patel38/csci_5408_s24_3/-/tree/main?ref_type=heads

Table of Contents

Architecture Diagram

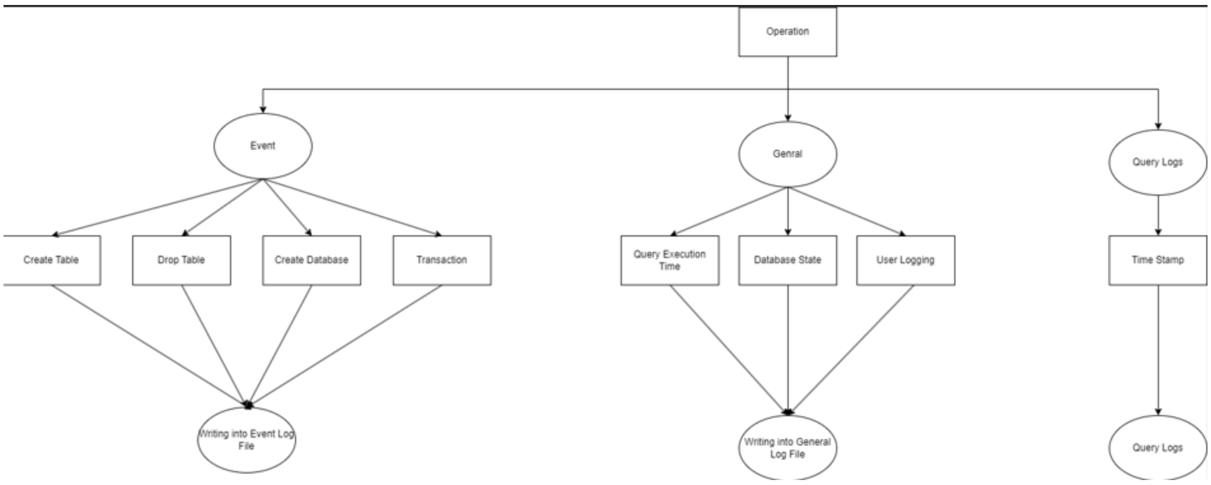


Figure 1: Workflow for the Log Management

Pseudocode

```
class EventLogsService:
  LOG_FILE = "TinyDB/EventLogs.txt"
  method logCreateTable(newTableName: String, userName: String):
        APPEND timestamp
        APPEND logtype as create_table
        APPEND newTableName
        APPEND username
        WRITE INTO LOG_FILE


  method logDropTable(tableName: String, userName: String):
        APPEND timestamp
        APPEND logtype as drop_table
        APPEND tableName
        APPEND username
        WRITE INTO LOG_FILE

  method logCreateDatabase(newDatabaseName: String, userName: String):
        APPEND timestamp
        APPEND logtype as create_database
        APPEND newDatabaseName
        APPEND username
        WRITE INTO LOG_FILE

  method logTransactionStarted(databaseName: String, userName: String):
        APPEND timestamp
        APPEND logtype as start_transaction_log
        APPEND databaseName
        APPEND username
        WRITE INTO LOG_FILE

  method logTransactionCommitted(databaseName: String, userName: String):
        APPEND timestamp
        APPEND logtype as transaction_committed_log
        APPEND databaseName
        APPEND username
        WRITE INTO LOG_FILE

  method logTransactionRollback(databaseName: String, userName: String):
        APPEND timestamp
        APPEND logtype as transaction_rollback_log
        APPEND databaseName
        APPEND username
        WRITE INTO LOG_FILE
  method logErrorMessage(errorMessage: String, userName: String):
        APPEND timestamp
        APPEND logtype as error_log
        APPEND errorMessage
        if username is not null:
              APPEND username
        WRITE INTO LOG_FILE
```

class GeneralLogsService:

```
  LOG_FILE = "TinyDB/GeneralLogs.txt"
  method logQueryExecutionTime(query: String, executionTime: long):

        APPEND timestamp
        APPEND logtype as query_execution_time
        APPEND query
        APPEND EXECUTION TIME
        WRITE INTO LOG_FILE


  method logDatabaseState(dbState: Map<String, Integer>):

        APPEND timestamp
        APPEND logtype as database_state
        APPEND number of tables
        for tableEntry in dbState:
              APPEND Table Name
              APPEND Number of Records
        WRITE INTO LOG_FILE

  method logUserRegistered(username: String):

        APPEND timestamp
        APPEND logtype as user_registered
        APPEND username
        WRITE INTO LOG_FILE


  method logUserLoggedIn(username: String):

        APPEND timestamp
        APPEND logtype as user_logged_in
        APPEND username
        WRITE INTO LOG_FILE
```

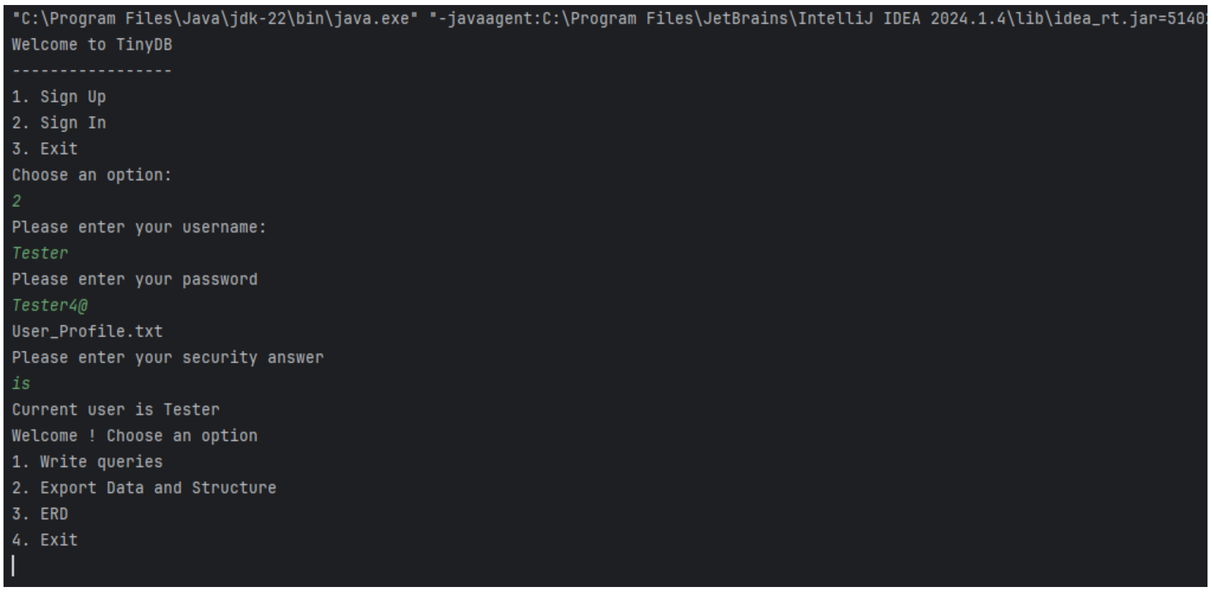
```
class QueryLogsService:
  LOG_FILE = "TinyDB/QueryLogs.txt"\

  method logQuerySubmission(query: String, queryTimeStamp: String):

        APPEND timestamp
        APPEND logtype as query_log
        APPEND query
        WRITE INTO LOG_FILE
```

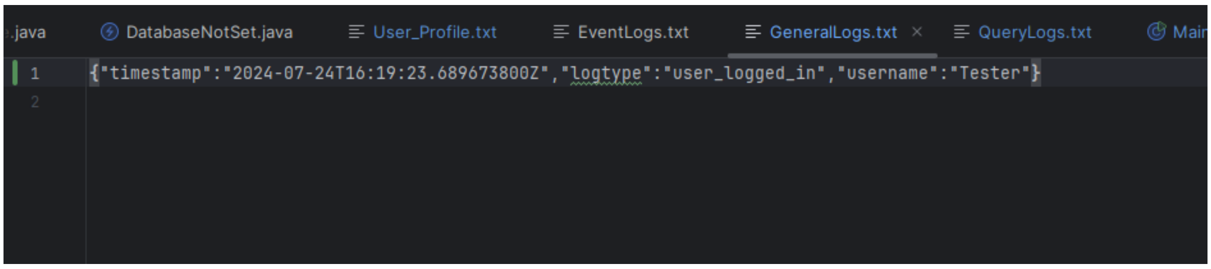
Test Cases and Evidence of Testing

In this scenario, we are logging into the TINY_DB. Upon logging in, a log entry should be created, and this entry will be stored in the general logs.



```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.1.4\lib\idea_rt.jar=5140
Welcome to TinyDB
----------------
1. Sign Up
2. Sign In
3. Exit
Choose an option:
2
Please enter your username:
Tester
Please enter your password
Tester4@
User_Profile.txt
Please enter your security answer
is
Current user is Tester
Welcome ! Choose an option
1. Write queries
2. Export Data and Structure
3. ERD
4. Exit
|
```

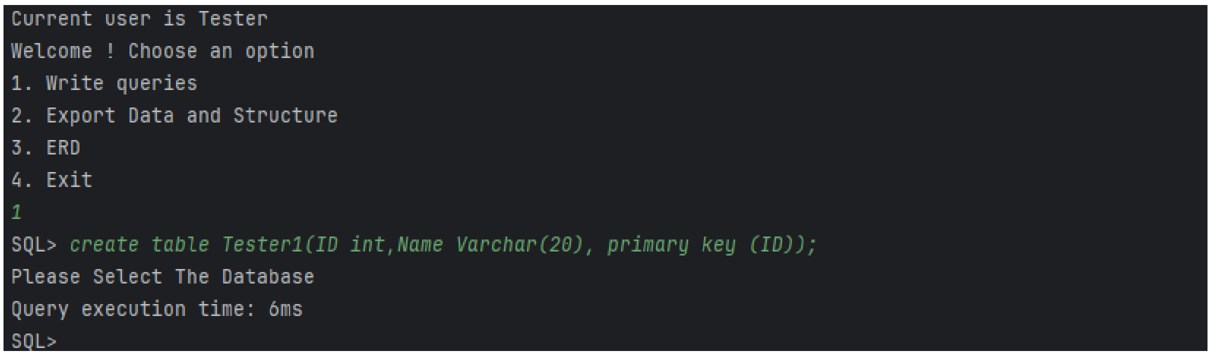
Figure 2: Logging into TINY_DB



```
{"timestamp":"2024-07-24T16:19:23.689673800Z","logtype":"user_logged_in","username":"Tester"}
```

Figure 3: General Log File

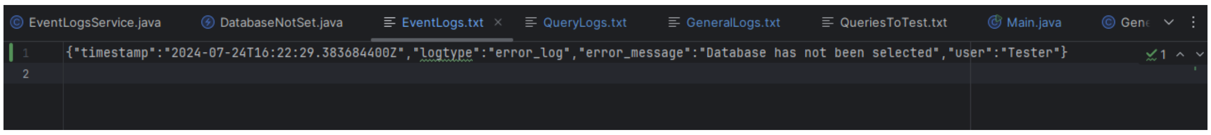
In this scenario, we're demonstrating an error that occurs when we run a query without first using the USE database command.



```
Current user is Tester
Welcome ! Choose an option
1. Write queries
2. Export Data and Structure
3. ERD
4. Exit
1
SQL> create table Tester1(ID int,Name Varchar(20), primary key (ID));
Please Select The Database
Query execution time: 6ms
SQL>
```

Figure 4: Not using "USE DATABASE" Before writing query

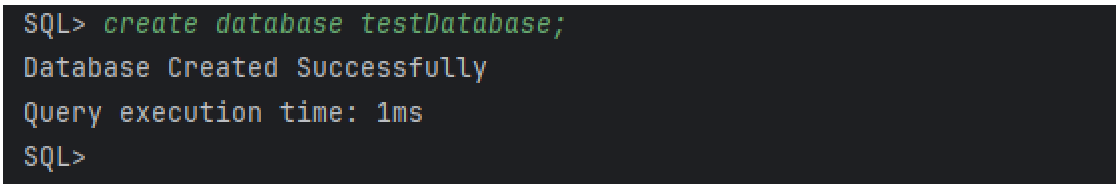
As soon as we run a query without using the USE database command, it will be recognized as an error. The program will then generate a log of this error and store it in the EventLog file.



```
{"timestamp":"2024-07-24T16:22:29.383684400Z","logtype":"error_log","error_message":"Database has not been selected","user":"Tester"}
```

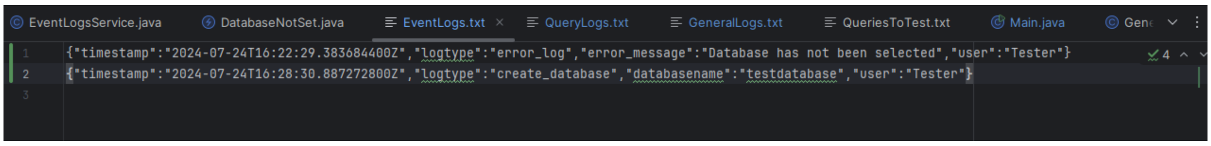
Figure 5: Event Log File (After Create Table Command)

In this scenario, we are creating a database. As soon as the database is created, the event function will be triggered, logging the creation of the database.



```
SQL> create database testDatabase;
Database Created Successfully
Query execution time: 1ms
SQL>
```

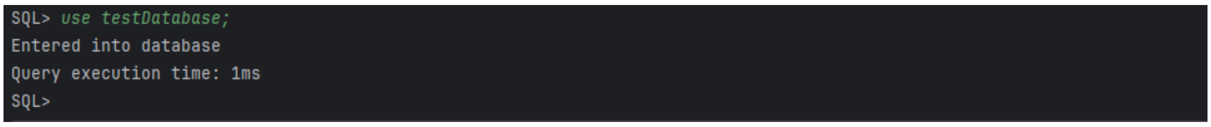
Figure 6: Create database Command



```
{"timestamp":"2024-07-24T16:22:29.383684400Z","logtype":"error_log","error_message":"Database has not been selected","user":"Tester"}
{"timestamp":"2024-07-24T16:28:30.887272800Z","logtype":"create_database","databasename":"testdatabase","user":"Tester"}
```

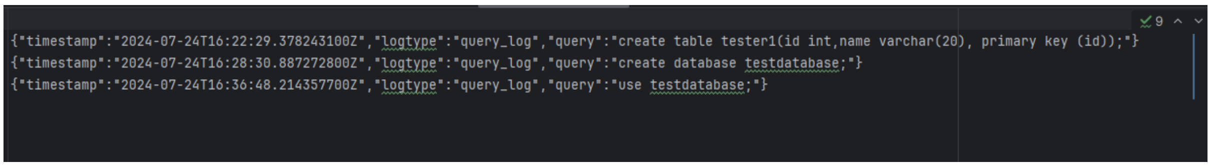
Figure 7: Event Log File (After Create Database Command)

Up until now, we have been dealing with a single type of log file. From now on, multiple log files will be involved. If you execute the USE database command, all related logs will be stored in the query log file, indicating that the query ran successfully. Additionally, database logs will be added to the general logs, providing information about the number of tables and Data in it.



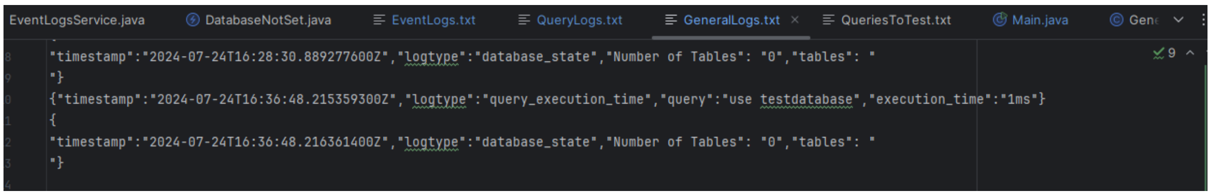
```
SQL> use testDatabase;
Entered into database
Query execution time: 1ms
SQL>
```

Figure 8: “USE DATABASE” Command



```
{"timestamp":"2024-07-24T16:22:29.378243100Z","logtype":"query_log","query":"create table tester1(id int,name varchar(20), primary key (id));"}
{"timestamp":"2024-07-24T16:28:30.887272800Z","logtype":"query_log","query":"create database testdatabase;"}
{"timestamp":"2024-07-24T16:36:48.214357700Z","logtype":"query_log","query":"use testdatabase;"}
```

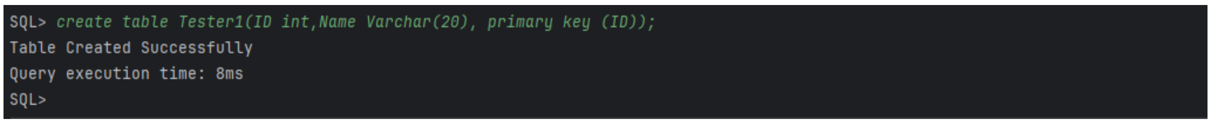
Figure 9: Query Log File (After use database Command)



```
"timestamp":"2024-07-24T16:28:30.889277600Z","logtype":"database_state","Number of Tables": "0","tables": "
"}
{"timestamp":"2024-07-24T16:36:48.215359300Z","logtype":"query_execution_time","query":"use testdatabase","execution_time":"1ms"}
{
"timestamp":"2024-07-24T16:36:48.216361400Z","logtype":"database_state","Number of Tables": "0","tables": "
"}
```

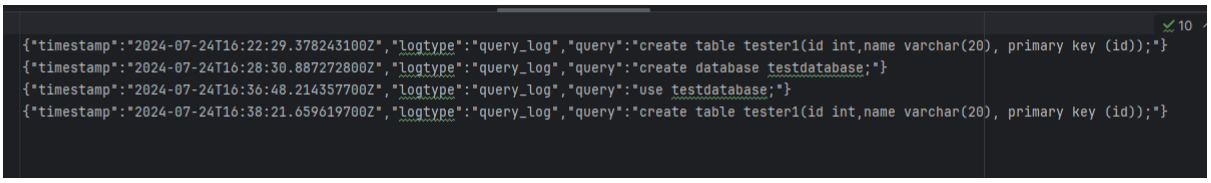
Figure 10: General Log File (After use Database Command)

In this scenario, we are creating the table 'Tester1'. After the table is created, the query log will receive an entry indicating that the query ran successfully. Additionally, the event logs will record that a table has been added to the database. In the general logs, the database schema will be updated, showing that there is currently one table with 0 rows.



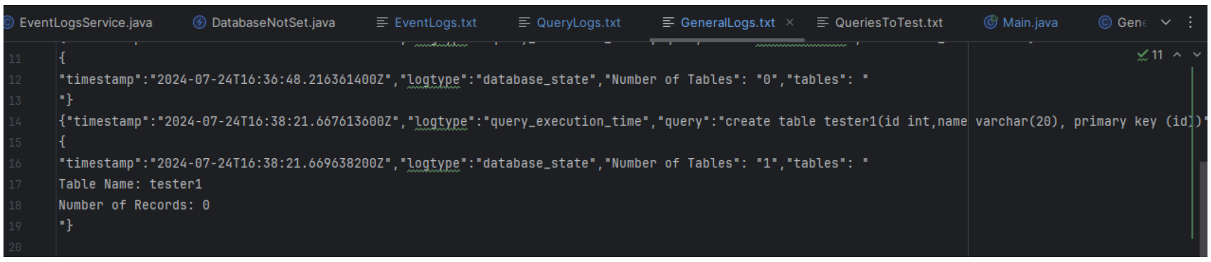
```
SQL> create table Tester1(ID int,Name Varchar(20), primary key (ID));
Table Created Successfully
Query execution time: 8ms
SQL>
```

Figure 11: Create Table command



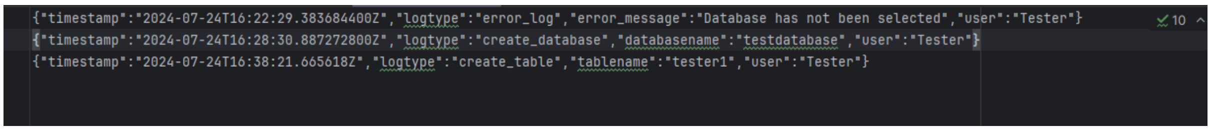
```
{"timestamp":"2024-07-24T16:22:29.378243100Z","logtype":"query_log","query":"create table tester1(id int,name varchar(20), primary key (id));"}
{"timestamp":"2024-07-24T16:28:30.887272800Z","logtype":"query_log","query":"create database testdatabase;"}
{"timestamp":"2024-07-24T16:36:48.214357700Z","logtype":"query_log","query":"use testdatabase;"}
{"timestamp":"2024-07-24T16:38:21.659619700Z","logtype":"query_log","query":"create table tester1(id int,name varchar(20), primary key (id));"}
```

Figure 12: Query Log File (After Create Table command)



```
{
"timestamp":"2024-07-24T16:36:48.216361400Z","logtype":"database_state","Number of Tables": "0","tables": "
"}
{"timestamp":"2024-07-24T16:38:21.667613600Z","logtype":"query_execution_time","query":"create table tester1(id int,name varchar(20), primary key (id))"
{
"timestamp":"2024-07-24T16:38:21.669638200Z","logtype":"database_state","Number of Tables": "1","tables": "
Table Name: tester1
Number of Records: 0
"}
```

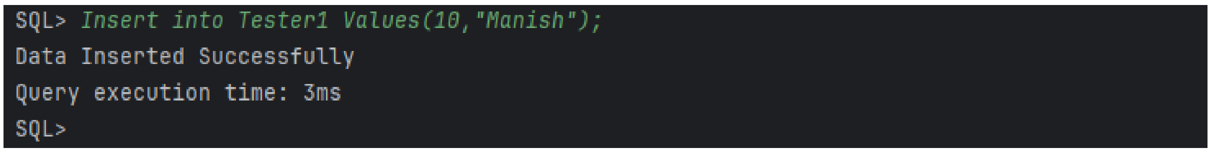
Figure 13: General Log File (After Create Table command)



```
{"timestamp":"2024-07-24T16:22:29.383684400Z","logtype":"error_log","error_message":"Database has not been selected","user":"Tester"}
{"timestamp":"2024-07-24T16:28:30.887272800Z","logtype":"create_database","databasename":"testdatabase","user":"Tester"}
{"timestamp":"2024-07-24T16:38:21.665618Z","logtype":"create_table","tablename":"tester1","user":"Tester"}
```

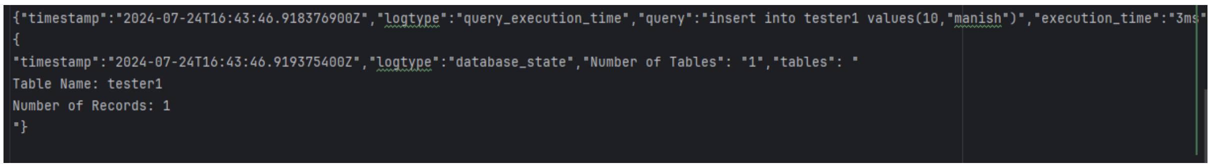
Figure 14: Event Log File (After Create Table command)

Now we are inserting data into the table. As soon as the data is inserted, the query log will include an entry indicating that the query was successfully executed. Additionally, the general logs will be updated with a new entry showing the number of tables and reflecting that the number of records has increased to 1.



```
SQL> Insert into Tester1 Values(10,"Manish");
Data Inserted Successfully
Query execution time: 3ms
SQL>
```

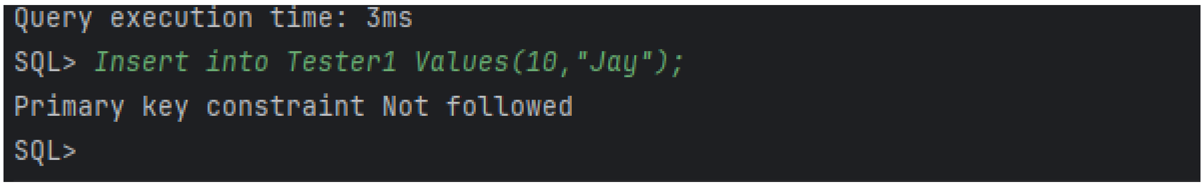
Figure 15: Insert Query command



```
{"timestamp":"2024-07-24T16:43:46.918376900Z","logtype":"query_execution_time","query":"insert into tester1 values(10,"manish")","execution_time":"3ms"
{
"timestamp":"2024-07-24T16:43:46.919375400Z","logtype":"database_state","Number of Tables": "1","tables": "
Table Name: tester1
Number of Records: 1
"}
```

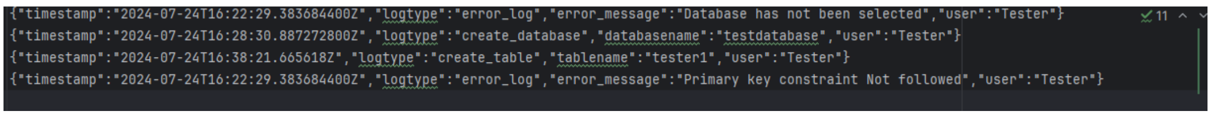
Figure 16: General Log (After Insert Query)

Now we are attempting to insert a duplicate key into the 'Tester' table. This will cause a failure due to the primary key constraint. In this case, the event file will record an error log indicating that the primary key constraint has not been followed.



```
Query execution time: 3ms
SQL> Insert into Tester1 Values(10,"Jay");
Primary key constraint Not followed
SQL>
```

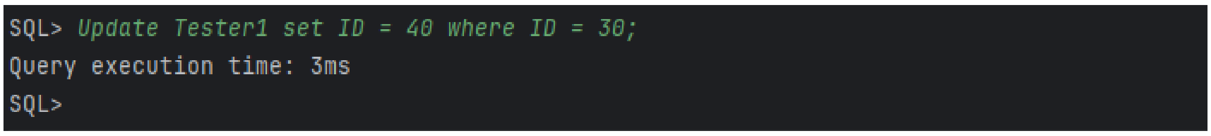
Figure 17: Inserting Duplicate Value



```
{"timestamp":"2024-07-24T16:22:29.383684400Z","logtype":"error_log","error_message":"Database has not been selected","user":"Tester"}
{"timestamp":"2024-07-24T16:28:30.887272800Z","logtype":"create_database","databasename":"testdatabase","user":"Tester"}
{"timestamp":"2024-07-24T16:38:21.665618Z","logtype":"create_table","tablename":"tester1","user":"Tester"}
{"timestamp":"2024-07-24T16:22:29.383684400Z","logtype":"error_log","error_message":"Primary key constraint Not followed","user":"Tester"}
```

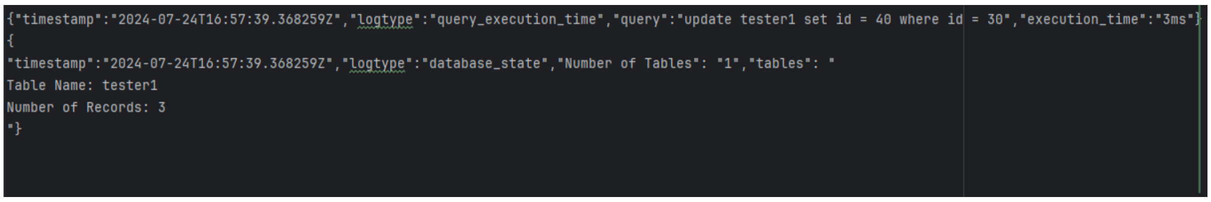
Figure 18: Event Log File (Insert Query for Duplicate Value)

Now we are updating a row based on the primary key. As a result, the query log will receive an entry indicating that the update query was performed successfully. In the general logs, the database schema will be updated to reflect the number of tables and the number of rows, which will remain the same since the update operation does not change the row count.



```
SQL> Update Tester1 set ID = 40 where ID = 30;
Query execution time: 3ms
SQL>
```

Figure 19: Update Command

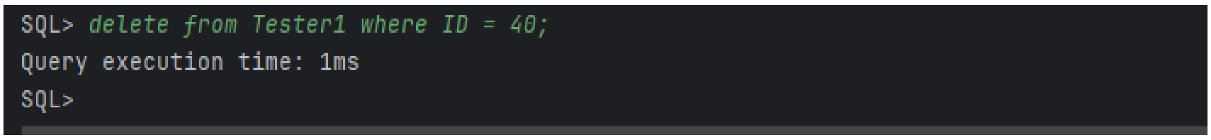


```
{"timestamp":"2024-07-24T16:57:39.368259Z","logtype":"query_execution_time","query":"update tester1 set id = 40 where id = 30","execution_time":"3ms"}
{
"timestamp":"2024-07-24T16:57:39.368259Z","logtype":"database_state","Number of Tables": "1","tables": "
Table Name: tester1
Number of Records: 3
"}
```

Figure 20: General Log File (After Update Command)

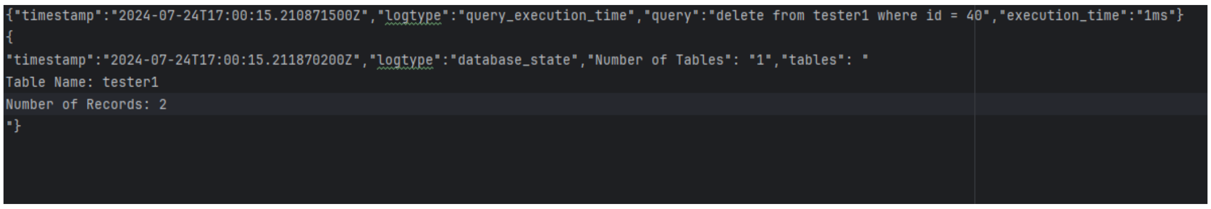
Now we are deleting rows from the table. In this case, logs will be added to two files:

1. In the general log file: An entry will be added showing the updated number of rows in the table. For example, the number of records in the 'Tester' table will be reduced to 2.
2. In the query log file: Similarly as before an entry will be added indicating that the delete query was executed successfully.



```
SQL> delete from Tester1 where ID = 40;
Query execution time: 1ms
SQL>
```

Figure 21: Delete Table Command

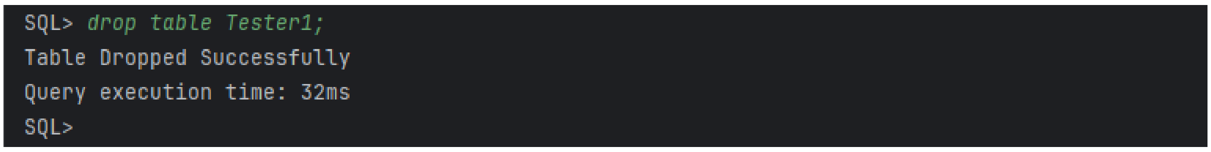


```
{"timestamp":"2024-07-24T17:00:15.210871500Z","logtype":"query_execution_time","query":"delete from tester1 where id = 40","execution_time":"1ms"}
{
"timestamp":"2024-07-24T17:00:15.211870200Z","logtype":"database_state","Number of Tables": "1","tables": "
Table Name: tester1
Number of Records: 2
"}
```

Figure 22: General Log File (After Delete Table)

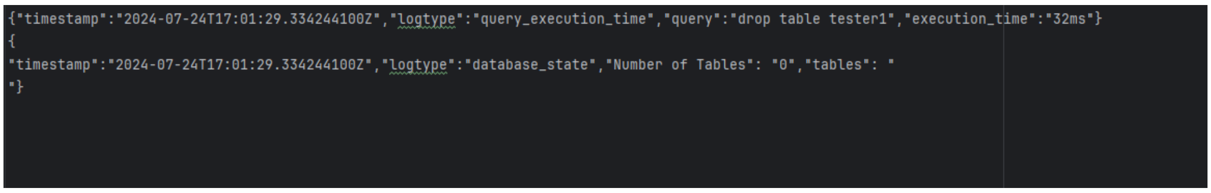
"In this scenario, we are dropping the table. Logs will be added to two files:

1. **In the general log file**: An entry will be added showing the updated schema. For example, the number of tables will now be 0.
2. **In the event log file**: An entry will be added indicating that the table has been dropped."



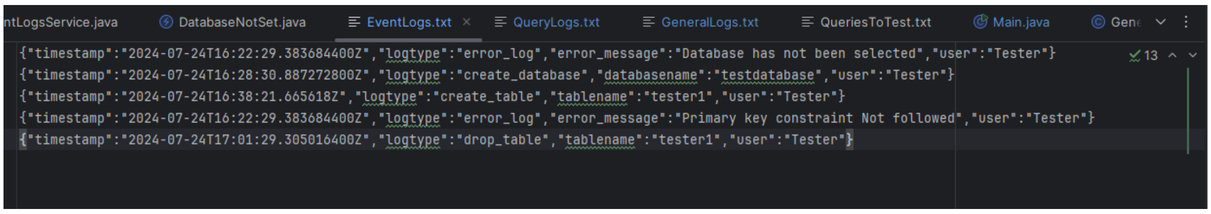
```
SQL> drop table Tester1;
Table Dropped Successfully
Query execution time: 32ms
SQL>
```

Figure 23: Drop Table Command



```
{"timestamp":"2024-07-24T17:01:29.334244100Z","logtype":"query_execution_time","query":"drop table tester1","execution_time":"32ms"}
{
"timestamp":"2024-07-24T17:01:29.334244100Z","logtype":"database_state","Number of Tables": "0","tables": "
"}
```

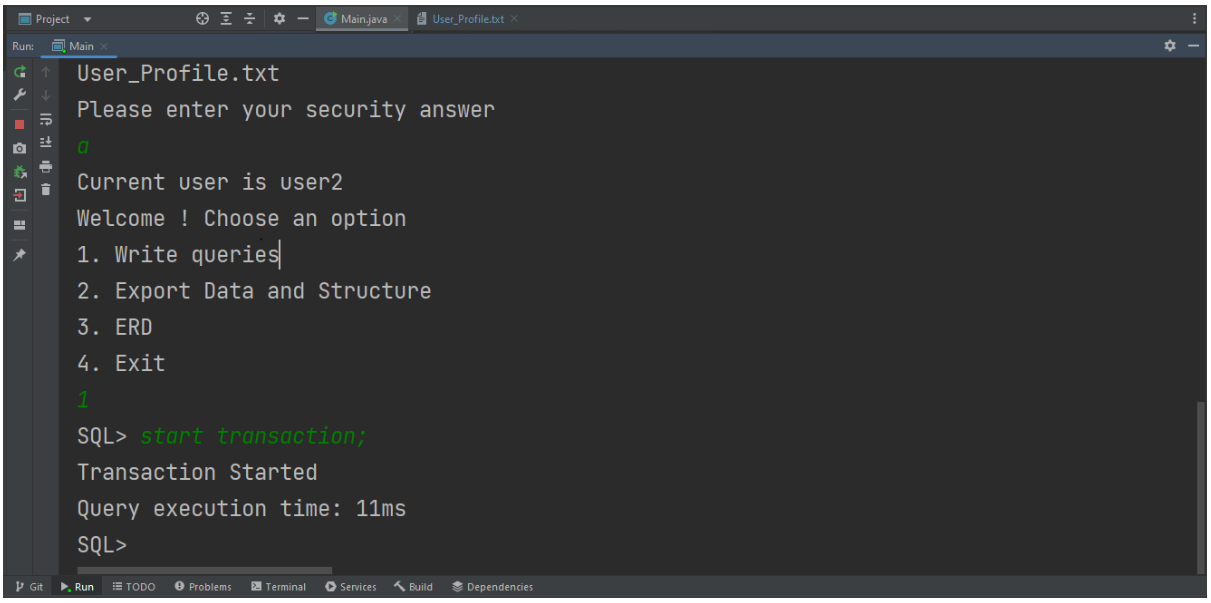
Figure 24: General Log File (After Drop Table Command)



```
{"timestamp":"2024-07-24T16:22:29.383684400Z","logtype":"error_log","error_message":"Database has not been selected","user":"Tester"}
{"timestamp":"2024-07-24T16:28:30.887272800Z","logtype":"create_database","databasename":"testdatabase","user":"Tester"}
{"timestamp":"2024-07-24T16:38:21.665618Z","logtype":"create_table","tablename":"tester1","user":"Tester"}
{"timestamp":"2024-07-24T16:22:29.383684400Z","logtype":"error_log","error_message":"Primary key constraint Not followed","user":"Tester"}
{"timestamp":"2024-07-24T17:01:29.305016400Z","logtype":"drop_table","tablename":"tester1","user":"Tester"}
```

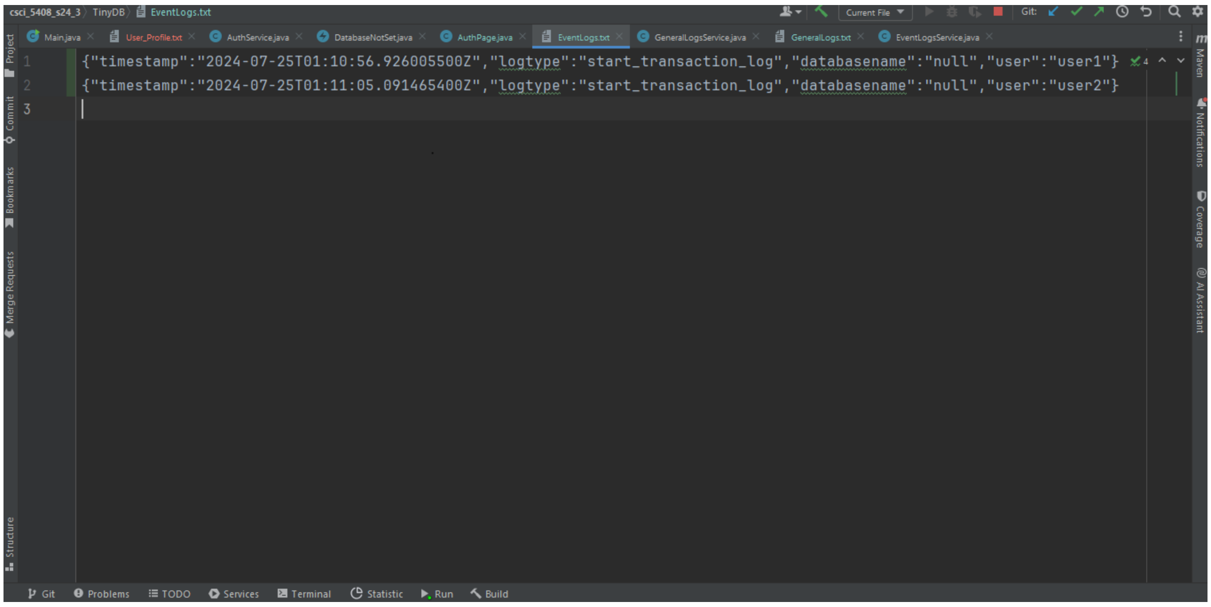
Figure 25: Event Log File (After Drop Table Command)

When a transaction is started, it will create an entry in the event log file indicating that the transaction has been initiated by a specific user.



```
User_Profile.txt
Please enter your security answer
a
Current user is user2
Welcome ! Choose an option
1. Write queries
2. Export Data and Structure
3. ERD
4. Exit
1
SQL> start transaction;
Transaction Started
Query execution time: 11ms
SQL>
```

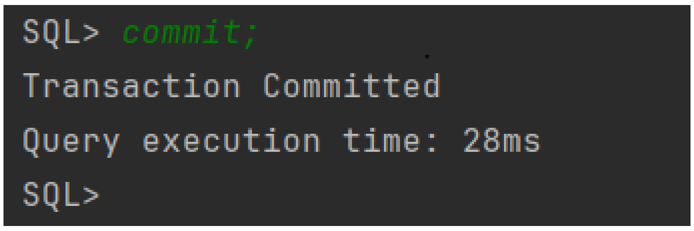
Figure 26: Start Transaction Command



```
{"timestamp":"2024-07-25T01:10:56.926005500Z","logtype":"start_transaction_log","databasename":"null","user":"user1"}
{"timestamp":"2024-07-25T01:11:05.091465400Z","logtype":"start_transaction_log","databasename":"null","user":"user2"}
```

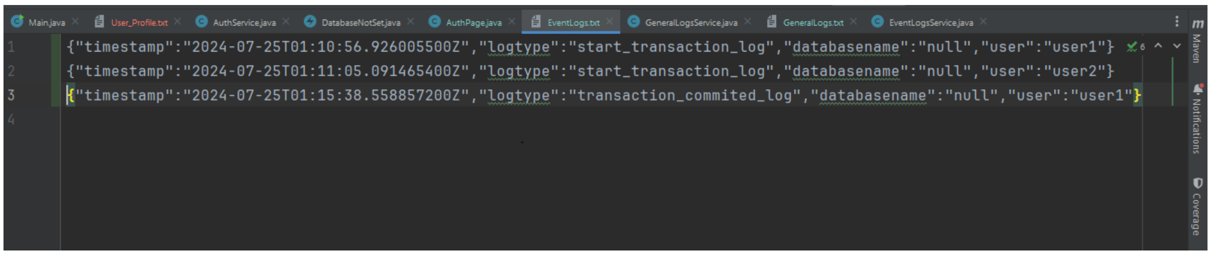
Figure 27: Event Log File (After Start Transaction Command)

When a transaction is committed, an entry will be created in the event log file, indicating that the transaction has been successfully completed by a specific user



```
SQL> commit;
Transaction Committed
Query execution time: 28ms
SQL>
```

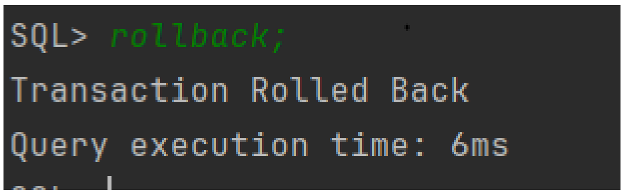
Figure 28: Commit Command



```
{"timestamp":"2024-07-25T01:10:56.926005500Z","logtype":"start_transaction_log","databasename":"null","user":"user1"}
{"timestamp":"2024-07-25T01:11:05.091465400Z","logtype":"start_transaction_log","databasename":"null","user":"user2"}
{"timestamp":"2024-07-25T01:15:38.558857200Z","logtype":"transaction_commited_log","databasename":"null","user":"user1"}
```

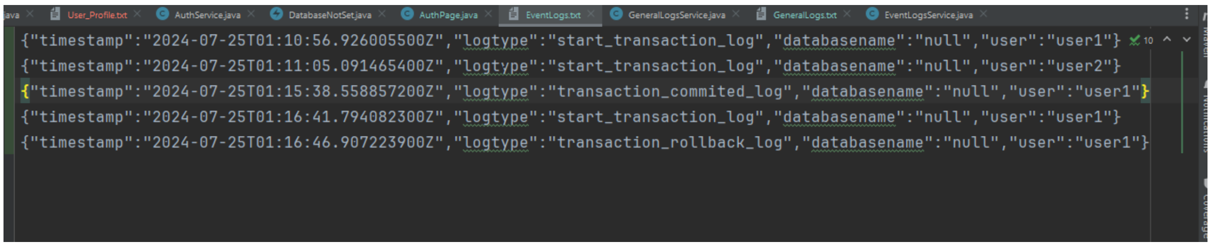
Figure 29: Event Log File (After Commit Command)

When a transaction is rolled back, an entry will be created in the event log file, indicating that the transaction has been reverted by a specific user



```
SQL> rollback;
Transaction Rolled Back
Query execution time: 6ms
```

Figure 30: Rollback Command



```
{"timestamp":"2024-07-25T01:10:56.926005500Z","logtype":"start_transaction_log","databasename":"null","user":"user1"}
{"timestamp":"2024-07-25T01:11:05.091465400Z","logtype":"start_transaction_log","databasename":"null","user":"user2"}
{"timestamp":"2024-07-25T01:15:38.558857200Z","logtype":"transaction_commited_log","databasename":"null","user":"user1"}
{"timestamp":"2024-07-25T01:16:41.794082300Z","logtype":"start_transaction_log","databasename":"null","user":"user1"}
{"timestamp":"2024-07-25T01:16:46.907223900Z","logtype":"transaction_rollback_log","databasename":"null","user":"user1"}
```

Figure 31: Event Log File (After Rollback Command)

Meeting Logs

Date	**Time**	**Attendees**	**Agenda**	**Meeting Type**	**Meeting Recording Link**
20-07-2024	3:30-4:00	Manish, Jay	Discussing The Flow of Final Module along with the presentation Preparation	Online	**Meeting in _General_-20240720_230829-Meeting Recording.mp4**