**L-7 Database Systems**

**ACCA7002**

**Assignment 1**

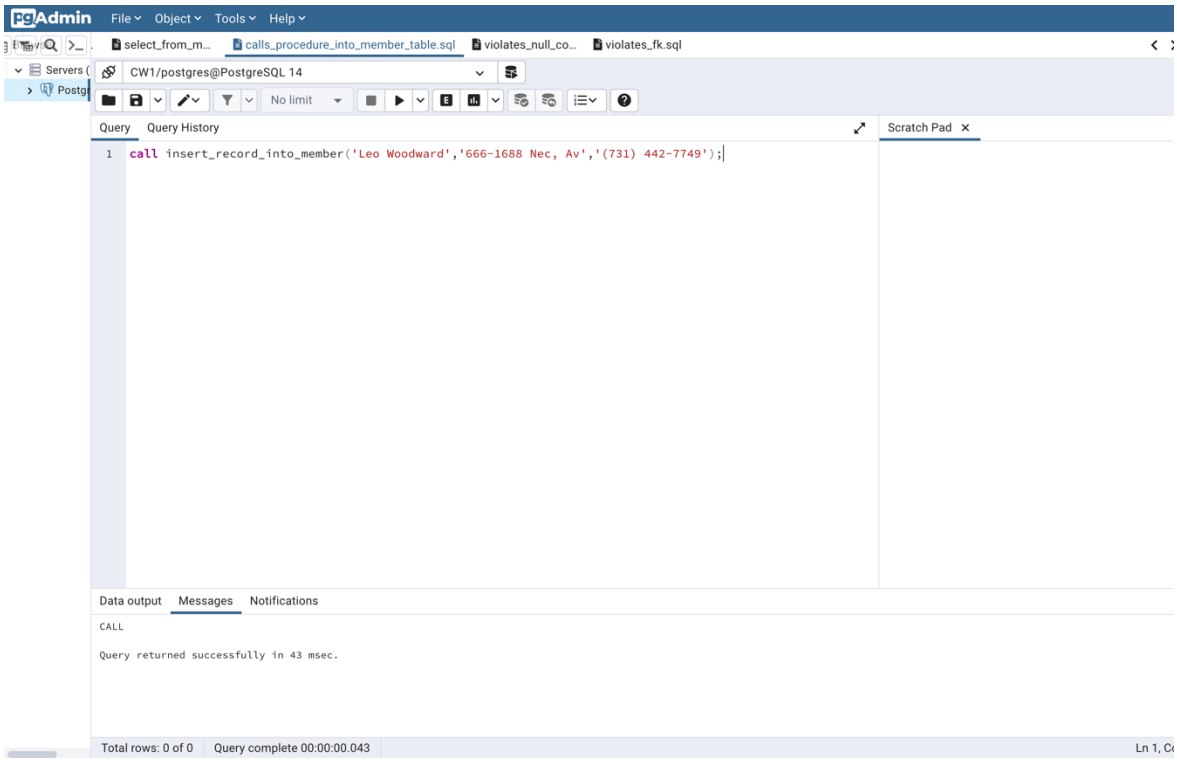
**Madhusudan Ghumare**

**2116948**

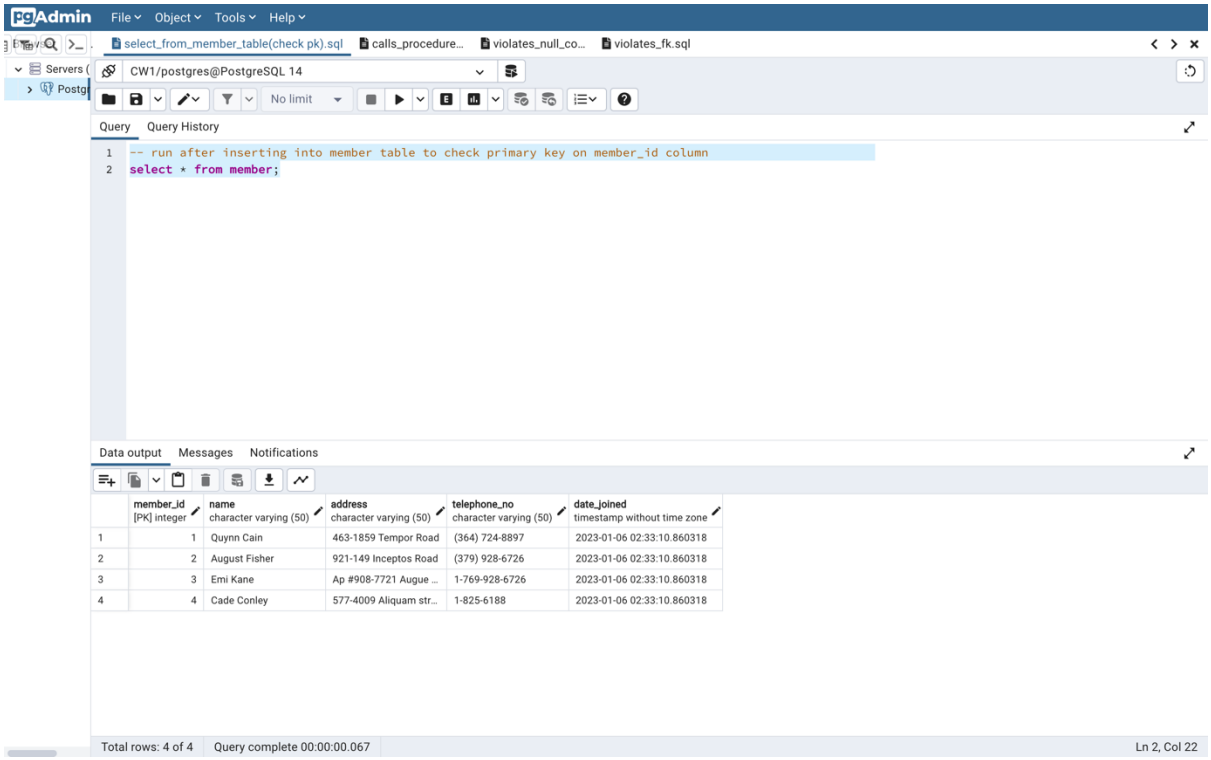
The most important queries in the ALC database are those in ‘member’ and ‘booked_class’ tables. Both of this tables handle core transactions of the club. In the ‘member’ table one of the most important query is checking if a member exists and their membership status. While in the ‘booked_class’ table one frequently used query would be checking if a class is active (if it has more than six members).

**Transactions**

Transactions are used to ensure the consistency of a database by allowing multiple related database operations to be treated as a single unit of work. In **file2** I have created different stored procedure to handle transactions. First there are six separate transactions to handle ‘INSERT’ operations for each table, for example insert_record_into_membership_type() inserts values into the membership_type table. Secondly, update_delete_procedure(query text) stored procedure handles all ‘UPDATE’ and ‘DELETE’ in the database.

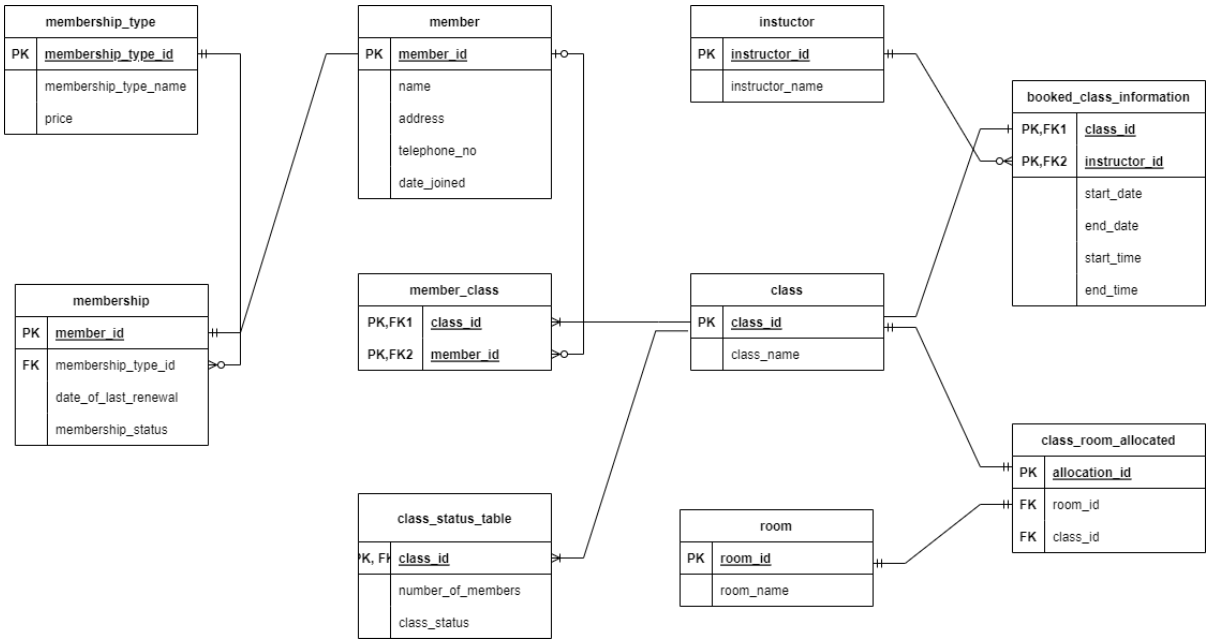



Stored Procedure Into Member




Primary Key Member Table

When a user calls the stored procedures, the transaction ensures that either the operations are completed successfully or are not completed at all. If an operation fails, the transaction is rolled back and the database is returned to its previous state, otherwise the transaction is recorded into the database. This ensures that the database remains consistent

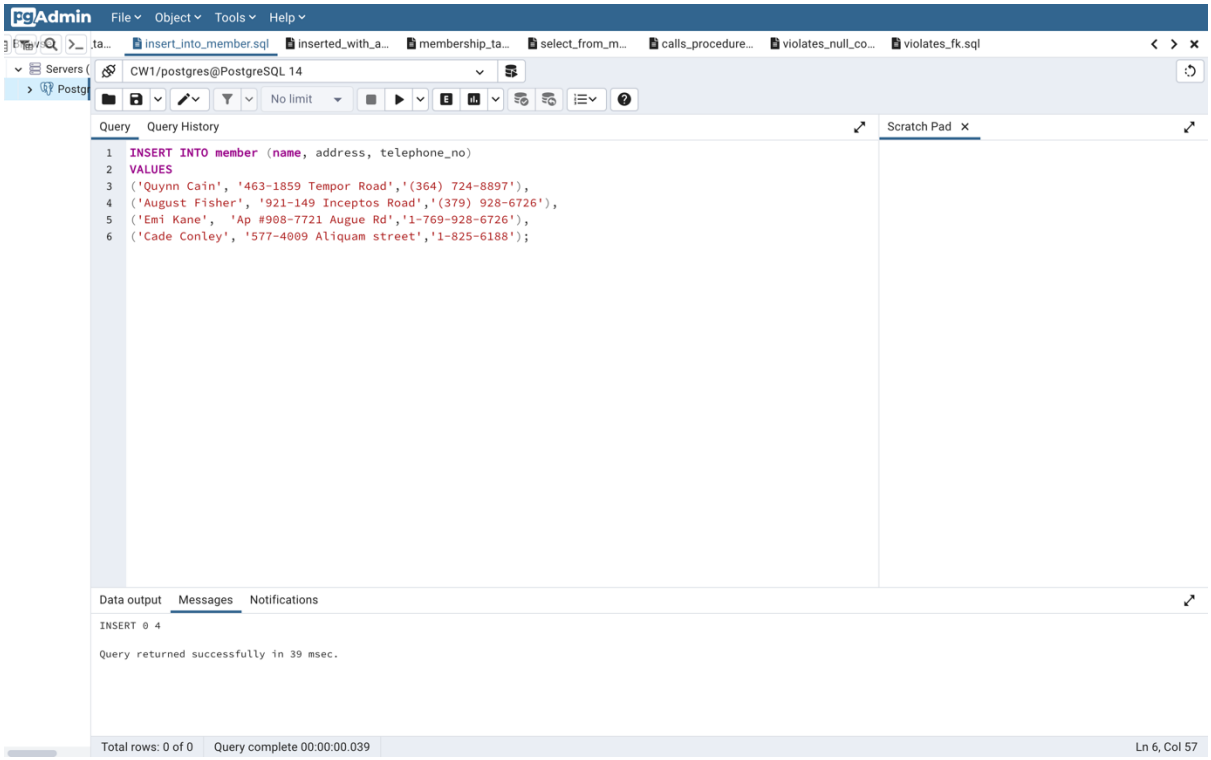


The above ER diagram represents six entities. The top box in each of the entities represents the name of the entity, distinguished by coloring the box green. In every entity there is a underlined attribute which represents the primary key, below it are other attributes of the entity. The entity can be described as follows:

- The first entity, on the top left corner, membership, contains data of membership_type types and their price.
- Member entity represents members,their membership type , membership status and other personal details.
- Booked_class entity represents booked classes. It uses a composite key(class_id and room_id).
- Class entity represent class , the class name and the instructor.
- Instructor entity represents instructors and their name.
- Lastly, room entity represents rooms and their name.

Each of the above entity has a relationship with one or two other entities. Relationships are represented by the lines connected from one to another entity:

- The relationship between membership and member, means a member can only belong to one membership type while a membership type may have several members.
- Between members and booked_class means members may belong to optionally one or many classes.
- An instructor can instruct many classes but a class can only have one instructor.
- A room can only be associated with one class.
- A class can be booked once or severally.




Inserted Data

**Evaluation of design and implementation**

The entity relationship (ER) is an essential part of this design, as it has been in existence for more than 35 years. It was implemented in this project because of its ability to allow modeling for use with databases since it is fairly abstract and is straightforward to understand and explain. It is a deal because it is fairly abstract and straightforward to understand and explain in a database. As a result, this model's relationship is easily converted from ER models. As such, ER models are also called the "ER schema" and are hence represented by ER diagrams. Along with the ER, the ACID was used in this project to maintain consistency in the database and to provide users with easy access to the Abertawe Leisure Centre (ALC).Isolation is another factor that affects transaction integrity, as it is apparent to other users and systems among these four attributes: atomicity, consistency, isolation, and durability. This indicates that a

transaction in a given system should take place so that it is the only transaction accessing the database system's resources (Liu, 2021).Hence, isolation plays a key role in this sector. This is because its level is meant to indicate the degree to which a transaction must be isolated in order to allow for transactions and alterations performed by any other transaction in the database system to take place. As such, transaction levels constitute the following phenomena:

**"Dirty Read": This** level happens in the transaction process when it reads data that has not yet been processed and committed in the database. For instance, in a case where transaction T1 manages to update a row and leaves it uncommitted, transaction T2 will take up this role and manage to read the changed row. As a result, if the first transaction fails, the second transaction will have read data that was never supposed to exist.

**Non-repeatable read:** this type of read will occur in the process if the transaction manages to read the same row in the database two times. For instance, suppose transaction T1 in our data base manages to read the data due to concurrency; another transaction will be updated by transaction T2, and the updated data will be the same as the data that was committed in the data base. Now if transaction T1 rereads the same data, it will obtain a different value.

**Phantom read:** this type of read will only manage to occur in our database if at all two identical queries are conducted in the process. If this occurs, the row of data will be returned because the two rows are distinct. An example is when the transaction T1 obtains a collection of records that fulfill some search criteria. When this happens, the T2 transaction will manage to create some new data that fits the transaction's T1 search criteria. This will give the users easy access to navigate and search through the courses and other relevant information in the database of Abertawe Leisure Centre (ALC). Additionally, the transaction T1 will manage to acquire a different set of rows if it re-executes the statement that reads the rows.

**The importance of the concepts of:**

**Rollback**

Rollback in our database system is a transactional control language that is very important when it comes to undoing the transactions that have not been stored in the database system. Since the last CMMIT, the rollback command has only been used to undo modifications in the database system. That makes the rollback an operational strategy that is a key factor in database technology because it returns the database to a previous state. Rollback is very important when it comes to the issues of database integrity management, because they do the work of restoring the database system to a clean copy even if erroneous actions are executed.

**Commit**

For our database system, commit plays a great role in allowing the system to permanently store the changes made in the transaction in tables or databases. As such, the data cannot manage to regain its prior state if at all the execution of the commit has taken place.As such, the transaction concept is the distinguishing feature of a database and the reason databases were created. Nevertheless, a commit point is the point at which a set of record updates is formally put into the database, and that set of changes is thereby assured to be safely stored even in the event of a power outage or OS crash by design and redundancy (Yang et al., 2021). A file system in the commit provides the database with a mechanism or guarantee that a file written will still be there after a power outage or other OS crash (Morrow et al., 2022).

**Functions**

Our database management system will enable the organisation (ALC) to administer the programed database in an easy way. It will allow users to interact with the database system to

access the underlying data. Such actions can range from providing requested data to allowing the creation of database schemas that have a significant impact on the database structure (Morrow et al., 2022). The data base will also allow for the configuration of user accounts, access controls, limits, and access scopes. As such, these activities will give the management the chance to deactivate other users, regulate user actions, and manage users in databases. DBMS can ease database backups by offering a simple and intuitive interface for managing backups and snapshots. These backups can even be stored on third-party sites such as cloud storage (Hang & Kim, 2019).The system will manage to monitor the performance of a database using integrated tools and enable users to customise databases by designing optimal indexes.

**Triggers**

The database trigger is critical because it is stored in the database and thus allows users who have the appropriate permissions or privileges to use it. Triggers are located in the database system; as such, it enables the user to write a series of SQL statements that various apps can use. Nevertheless, the triggers allow us to avoid writing duplicate code when numerous programmes must execute the same database operation.

**Procedures**

Database procedures, in most cases known as stored procedures or "procs," are the subroutines that can, in most cases, include one or more SQL statements that accomplish a specified operation. As a result, they play a crucial role in database systems since they can be used for data validation, access control, or to reduce network traffic between clients and DBMS servers. Superior performance is made possible by the subroutine's ability to incorporate complex and extensive business logic (Hang & Kim, 2019) As a result, the database procedures and the function are interchangeable. However, the main difference between the two is how they

manage to invoke Database functions can be used in SQL statements in the same way as any other expression, However, depending on the DBMS, database procedures must be called using the CALL or EXEC verb (Togatorop et al., 2022). As such, the procedure manages to add an interface between the database and the user interface. Since end users can enter or change data but not create procedures, it provides security through data access controls (Hang & Kim, 2019).As such, a stored process within the system protects data integrity by ensuring that information is entered consistently.

**Transactions**

As previously stated in the paper, transactions are a critical series of steps used to perform logical tasks in a database management system. The transaction does the work of indicating that a database has changed. Nevertheless, one of the most important aspects of the DBMS is to protect the user from factors such as data failure. It manages to accomplish this by guaranteeing that all data is restored to a consistent condition when the machine is restarted following a crash (Togatorop et al., 2022). As such, when one programme is running multiple transactions, the programme manages to run several times. Simply said, this means that the user can access many data sets from the database independently of one another. (Liu, 2021). Nevertheless, transactions are used to manage concurrency. It's also utilised to meet acid requirements. It is employed to resolve read/write conflicts.

**Indexing**

An indexing system is a schema object that has an entry for each and every value that appears in the table as the one we created in the data base. The table is meant to cluster indexed columns and give a chance for direct, rapid access to rows. Indexes help the database application find

data quickly without having to read the entire table (Togatorop et al., 2022). Nevertheless, Hang & Kim (2019) state that updating a table with indexes takes longer than updating a table without indexes because the indexes also need an update and will always be searched against by the users. For instance, *create index IX. user_line_item_link (line_item_id) ON user_lilink_item_id;*

**Database Creation**

Creating a schema

A database is constructed from one or more named schemas, each of which contains tables. Data types, functions, and operators are examples of named items found in schemas. The same object name can be used without conflict in many schemas; for example, tables named mytable can exist in both schema1 and myschema. Schemas, unlike databases, are not tightly coupled: if a user has the necessary credentials, they can access objects in any of the schemas linked to the database.

Create Schema updates the current database's schema with a new one. The schema name needs to be unique compared to any other names for schemas in the current database. Because it contains named objects (tables, data types, functions, and operators) whose names may be similar to those of objects in other schemas, a schema is effectively a namespace. You can access named items by prefixing their names with the name of the corresponding schema or by setting a search route that includes the necessary schemas.

When using an unqualified object name in a CREATE command, the current schema's object is created (the one at the beginning of the search path, as defined by the function current schema). Subcommands for adding objects to the new schema can optionally be included in the Create Schema subcommand. The subcommands are essentially treated the same as

individual commands submitted after the schema is built, with the exception that all objects created if the AUTHORIZATION clause is present will belong to that user.

```
CREATE SCHEMA schema_name [ AUTHORIZATION role_specification ] [ schema_element [ ... ] ]
CREATE SCHEMA AUTHORIZATION role_specification [ schema_element [ ... ] ]
CREATE SCHEMA IF NOT EXISTS schema_name [ AUTHORIZATION role_specification ]
CREATE SCHEMA IF NOT EXISTS AUTHORIZATION role_specification

where role_specification can be:

    user_name
  | CURRENT_ROLE
  | CURRENT_USER
  | SESSION_USER
```

Parameters

*schema_name*

The name of the schema that will be constructed If this is not specified, the schema name is the user name. Because such names are restricted for system schemas, the name cannot begin with pg_.

*user_name*

The name of the user whose role will be in charge of the new schema If not specified, the command is executed by the user. To create a schema owned by another role, you must be a superuser, a direct or indirect member of such a role.

*schema_element*

an SQL statement that defines a schema object to be constructed. Create Schema currently accepts only Create Table, Create View, Create Index, Create Sequence, Create Trigger, and

Grant clauses. Following the creation of the schema, more types of objects can be generated using different commands.

## ERD

The entity-relationship diagram (ERD) is a database design tool that visualises database tables, columns, and inter-relationships. The ERD provides enough information to the database administrator to use while designing and managing the database. We can use this Tool to: create a visual representation of the database tables and their relationships. Automatically align and links the table for better visualisation, we save the diagram and return to it later to work on it, and Create SQL from the database design that is ready to run. As such, the ERD can allow a user to run many instances of the ERD tool in separate tabs at the same time.

## Conclusion

The reports have managed to create a system for Abertawe Leisure Centre (ALC). Additionally, this project has managed to elaborate on some of the most important components of the system, such as the evaluation of design and implementation, transactions, rollbacks, commits, etc.

**Reference**


Hang, L., & Kim, D. H. (2019). Design and implementation of an integrated iot blockchain platform for sensing data integrity. *Sensors*, *19*(10), 2228.

Liu, S. (2021, November). Application of Big Data and Computer 3D Simulation Technology in the Assistance of Training by Database System. In *2021 IEEE International Conference on Emergency Science and Information Technology (ICESIT)* (pp. 710-714). IEEE.

Morrow, N., Mock, N. B., Gatto, A., LeMense, J., & Hudson, M. (2022). Protective Pathways: Connecting Environmental and Human Security at Local and Landscape Level with NLP and Geospatial Analysis of a Novel Database of 1500 Project Evaluations. *Land*, *11*(1), 123.

Togatorop, P., Sitorus, H. A. T., Sirait, R. M., & Manurung, T. (2022). Database Audit System Design and Implementation. *Jurnal Mantik*, *5*(4), 2535-2541.

Urazova, T. (2022). *Building a System for Automated Question Generation and Evaluation to Assist Students Learning UML Database Design* (Doctoral dissertation, UNIVERSITY OF BRITISH COLUMBIA (Okanagan).

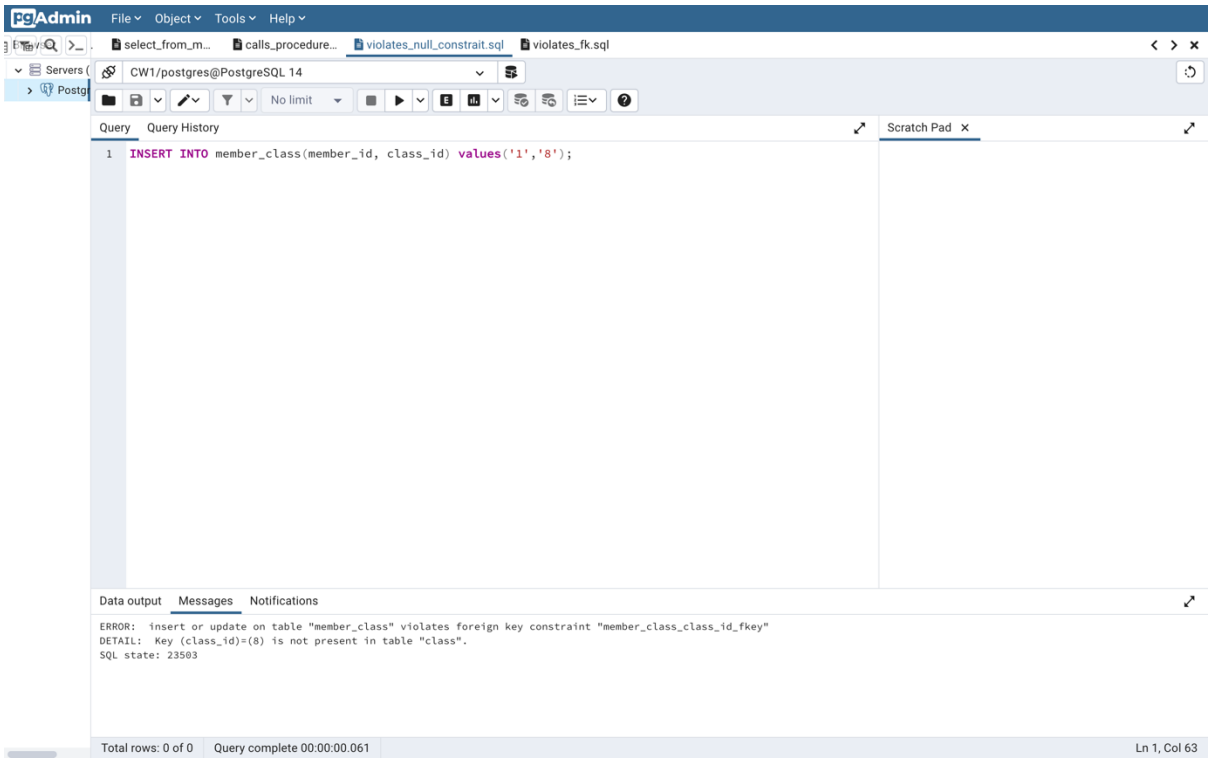
Yang, M., Zhang, M., Leng, T., Dong, W., & Yan, Y. (2021, December). Design and Implementation of Benchmarks for Multimodal Databases. In *2021 IEEE International Conference on Industrial Application of Artificial Intelligence (IAAI)* (pp. 490-494). IEEE.

*43.10. trigger functions* (2022) *PostgreSQL Documentation*. Available at: https://www.postgresql.org/docs/current/plpgsql-trigger.html.

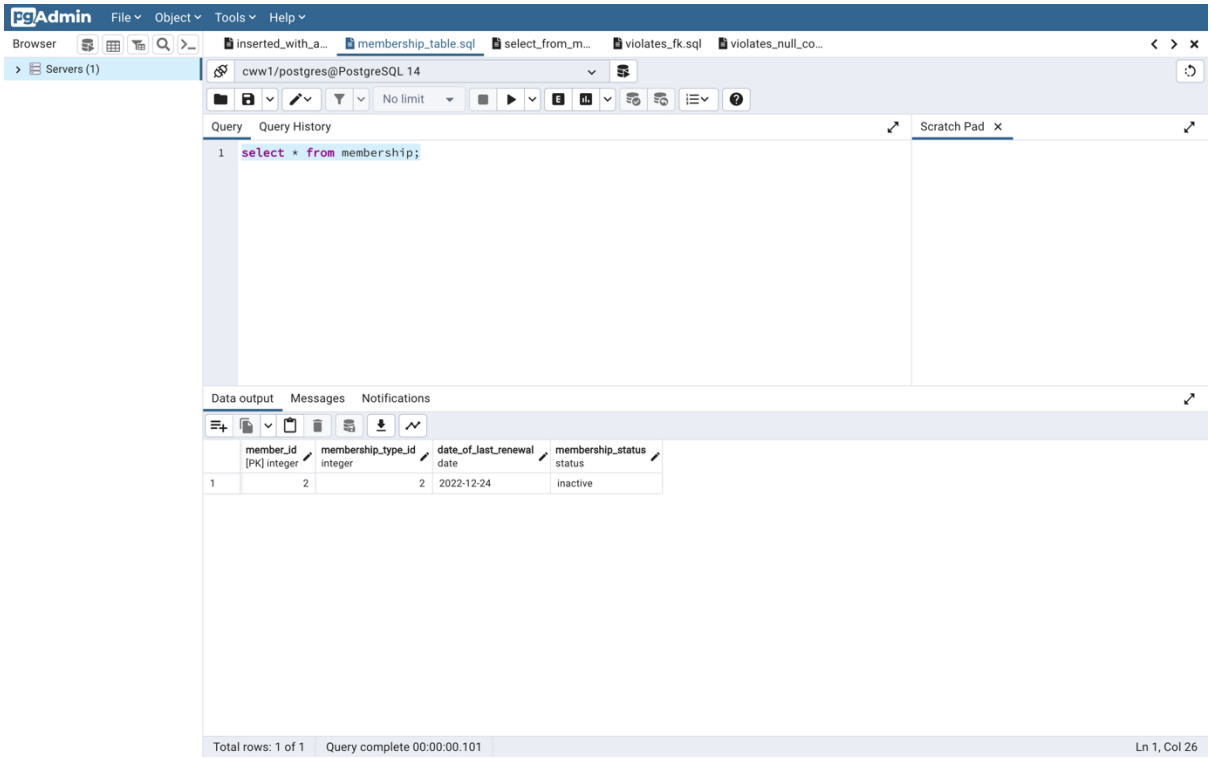
*Set transaction* (2022) *PostgreSQL Documentation*. Available at: https://www.postgresql.org/docs/15/sql-set-transaction.html .

*PostgreSQL reference manual* (2010). Bristol: Network Theory Ltd.

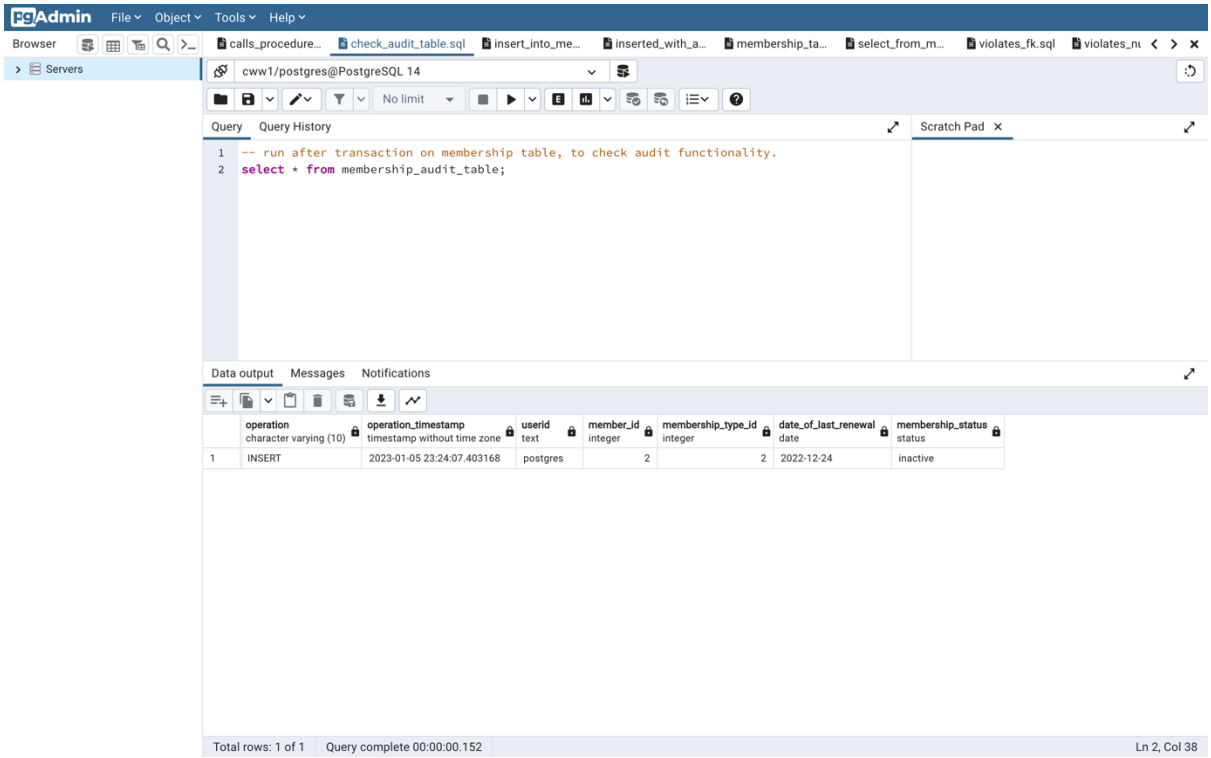
# Appendix



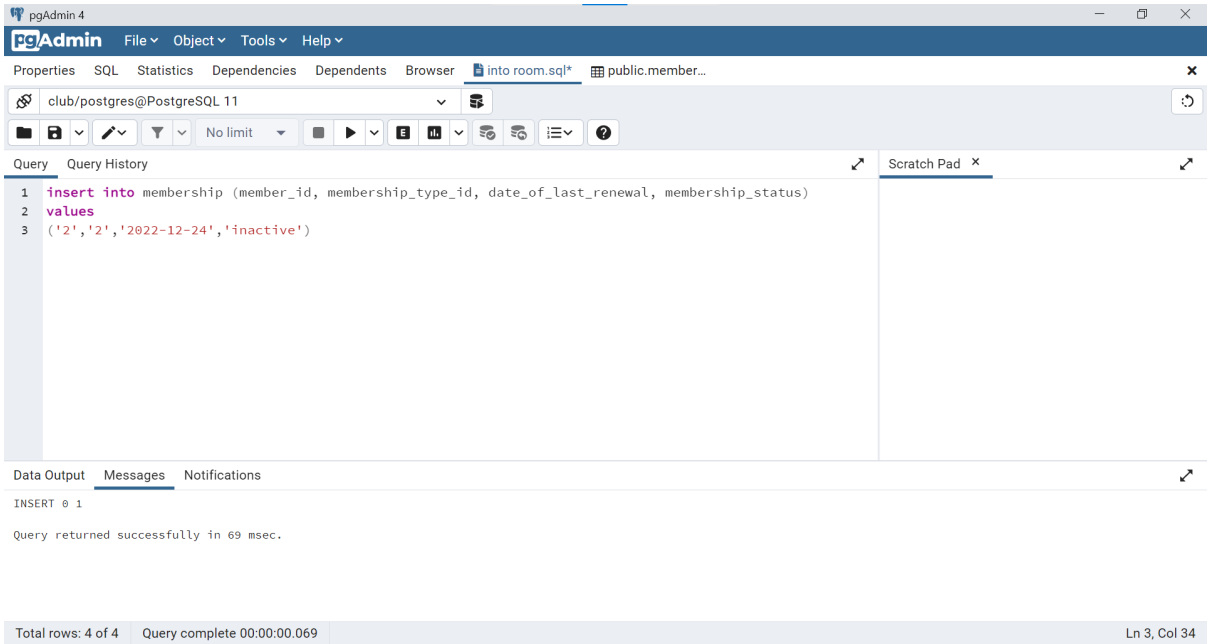

# Check Constraint Member



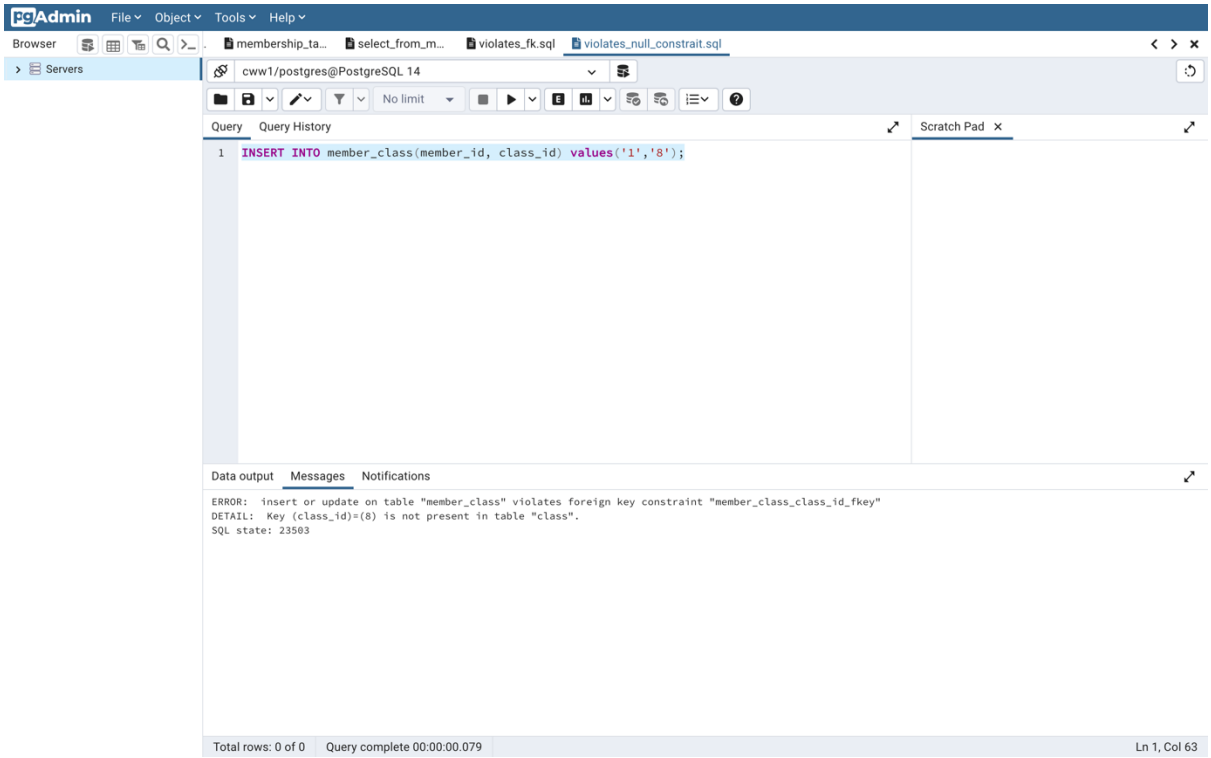

# Membership Table




Membership Audit




Inserted with audit


pgAdmin
File
Object
Tools
Help
Browser
membership_ta...
select_from_m...
violates_fk.sql
violates_null_constrait.sql
Servers
cww1/postgres@PostgreSQL 14
No limit
Query
Query History
Scratch Pad
INSERT INTO member_class(member_id, class_id) values('1','8');
Data output
Messages
Notifications
ERROR: insert or update on table "member_class" violates foreign key constraint "member_class_class_id_fkey"
DETAIL: Key (class_id)=(8) is not present in table "class".
SQL state: 23503
Total rows: 0 of 0
Query complete 00:00:00.079
Ln 1, Col 63

Error