Project Report - Data Storage Paradigms, IV1351

Natasha Donner

9 January 2023

1 Introduction

The students have previously created a conceptual model, a logical/physical model, and a database i PostgreSQL. In this assignment the student are given the task to develop part of Sound Good Music Website. The requirements are limited to a set of functionalities, namely what's used when instruments are rented. The focus is on database access and not website development, therefore the author will create a command line user interface. Additionally, the author expounds upon the topic of ACID transactions and the manner in which they are processed within the application.

2 Literature Study

The listing below is the literature used for getting the project started and used throughout the project. The sources include understanding databases, database managing systems, and conceptual, logical and psychical modelling.

Different sources:

- General:
  - Canvas webpage. link
  - Fundamentals of database systems 7th edition, Elmasri and Navathe.
  - PostgreSQL documentation.link
  - Astah link
- Task 1
  - Lectures by professor Leif
  - The course literature for Inheritance.
- Task 2
  - Course lecture given by professor Leif
  - Lab lectures

- Task 3
  - Course on SQL language at w3schools.com link
  - Writing sql queries link
- Task 4
  - Transaction lecture
  - Database Applications lecture
  - SQL queries link
  - Maven for IntelliJ link

3 Method

Development Environment
IntelliJ, an Integrated Development Environment (IDE) was utilized to write the Java code for the application that handles the functionalities requested in task 4 for Soundgood Music School. The PostgreSQL Database Management System (DBMS) was utilized, both in the psql shell and pgAdmin 4 graphical user interface (GUI), to write queries that extract and updates data from and to the database.

Queries
The initial step in developing the application that will manage certain renting functionalities for Soundgood Music School is composing the queries that will be used to retrieve and updating the necessary data from the database. These queries are tested using PostgreSQL to ensure their accuracy before they are implemented into the application program.

MVC, Layer patterns, DAO, DTO
The Model View Controller pattern (MVC) is used to create a well-structured and easy-to-use code and the design-focused Layer patterns are implemented to separate layers, packages and classes to organize the application further. Neither controller nor model contains any code related to the view (input and output). The Database access Object(DAO)layer only contains methods that handles the communication between the application and the database. As common practice the methods in the DAO layer is named after the proper convention, method names starts with "create", "find", "update" and "delete". There is no logic in the DAO layer. Data Transfer Object (DTO) is used in the application. The DTO is a design pattern that is used to transfer data between systems or layers in a software application. In the context of the Model-View-Controller (MVC) pattern, a DTO is typically used to transfer data between the model and the view. In our application the view access read-only data from the DTO.

4 Result

The result of the project is a easy-to-use command-line user interface that communicates with the Sound Good Music database. As mentioned in the method section, the application is based on layered patterns, the patterns we used in the application is shown in Figure 1.

The packages controller, integration, model, start, and view represent the various layers of the application, with each package containing a number of classes.

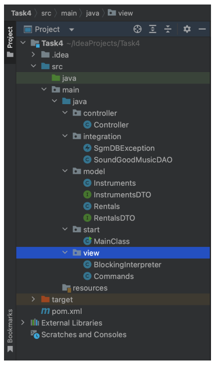


Figure 1: Layered pattern

The functionalities the program implements:

- ls
- rent
- leases
- terminate
- quit

Implementation of commands
The "ls" command can do two things; list all available instruments, and list all available instrument of a specific instrument type. To list all instruments you type: ls all. To list one specific instrument type you type: ls type.

The "rent" command allows you to create a new rental. It requires two pieces of information: the student's ID and the instrument's ID. To rent an instrument, you can first use the ls command to see which instruments are available, and then enter the rent command followed by the student ID and the instrument ID. If the student hasn't reached the maximum of 2 rentals and the instrument is available, the rental will be accepted.

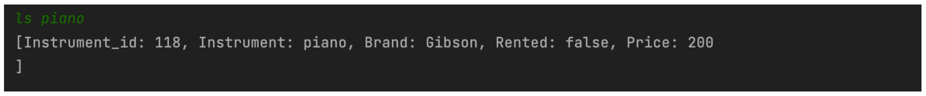
The "leases" command will give you a list of all current and past rentals. To see this list, simply type "leases".

To end a rental, use the "terminate" command along with the student's ID and the instrument's ID. For example, if you want to terminate a rental for student ID 123 and instrument ID 456, you would type: terminate 123 456. When a rental is terminated, all information about it is saved in the database, but in the "rent instrument" table, the "terminated" column will be marked as "true" and the "end at" column will be updated with the current date. Also, in the "instrument in stock" table, the "rented" column will be marked as "false".

The "quit" command allows you to exit the application. Simply type "quit" to close the program.

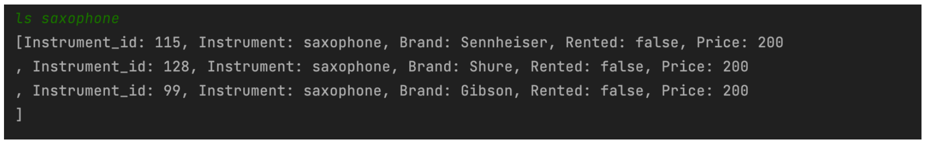
ACID transactions.

ACID transactions are a set of principles that ensure the accuracy and stability of data in a database. The term ACID stands for Atomicity, Consistency, Isolation, and Durability, and these properties describe the essential features of a successful database transaction. Atomicity means that a transaction is an all-or-nothing proposition: either all of the operations within a transaction are carried out, or none of them are. This ensures that the database remains in a consistent state, even if something goes wrong during the transaction. In the application the author manage atomicity by having auto commit turned off. The transaction is manually committed if all operation successfully carries out. In the method deleteRental we need to use multiple statements to complete the request, therefore these statements are treated as one single unit, hence the all-or-nothing proposition. Consistency means that a transaction must not violate the rules and constraints of the database. Transactions that violate this will not be committed. Isolation ensures that the operations within a transaction do not interfere with the operations of other transactions.Durability means that the effects of a transaction are permanent and cannot be undone, even if there is a system failure. These properties work together to preserve the integrity and reliability of data in a database, allowing multiple transactions to be executed concurrently without affecting the consistency of the data.



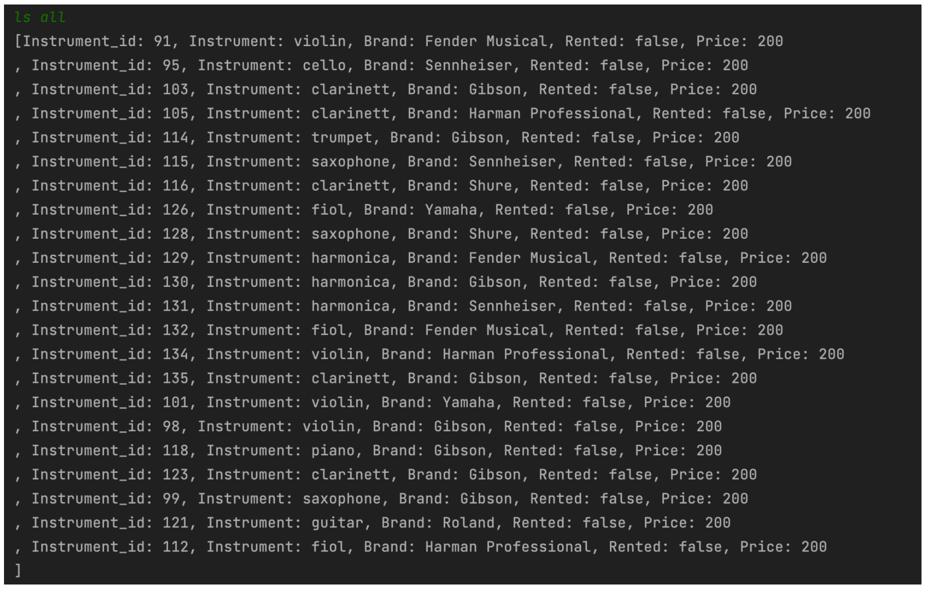
```
ls piano
[Instrument_id: 118, Instrument: piano, Brand: Gibson, Rented: false, Price: 200
]
```

Figure 2: ls piano



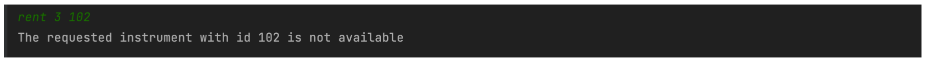
```
ls saxophone
[Instrument_id: 115, Instrument: saxophone, Brand: Sennheiser, Rented: false, Price: 200
, Instrument_id: 128, Instrument: saxophone, Brand: Shure, Rented: false, Price: 200
, Instrument_id: 99, Instrument: saxophone, Brand: Gibson, Rented: false, Price: 200
]
```

Figure 3: ls saxophone



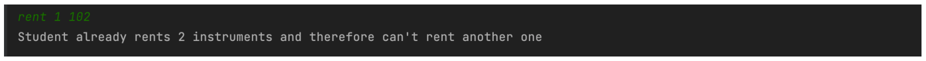
```
ls all
[Instrument_id: 91, Instrument: violin, Brand: Fender Musical, Rented: false, Price: 200
, Instrument_id: 95, Instrument: cello, Brand: Sennheiser, Rented: false, Price: 200
, Instrument_id: 103, Instrument: clarinett, Brand: Gibson, Rented: false, Price: 200
, Instrument_id: 105, Instrument: clarinett, Brand: Harman Professional, Rented: false, Price: 200
, Instrument_id: 114, Instrument: trumpet, Brand: Gibson, Rented: false, Price: 200
, Instrument_id: 115, Instrument: saxophone, Brand: Sennheiser, Rented: false, Price: 200
, Instrument_id: 116, Instrument: clarinett, Brand: Shure, Rented: false, Price: 200
, Instrument_id: 126, Instrument: fiol, Brand: Yamaha, Rented: false, Price: 200
, Instrument_id: 128, Instrument: saxophone, Brand: Shure, Rented: false, Price: 200
, Instrument_id: 129, Instrument: harmonica, Brand: Fender Musical, Rented: false, Price: 200
, Instrument_id: 130, Instrument: harmonica, Brand: Gibson, Rented: false, Price: 200
, Instrument_id: 131, Instrument: harmonica, Brand: Sennheiser, Rented: false, Price: 200
, Instrument_id: 132, Instrument: fiol, Brand: Fender Musical, Rented: false, Price: 200
, Instrument_id: 134, Instrument: violin, Brand: Harman Professional, Rented: false, Price: 200
, Instrument_id: 135, Instrument: clarinett, Brand: Gibson, Rented: false, Price: 200
, Instrument_id: 101, Instrument: violin, Brand: Yamaha, Rented: false, Price: 200
, Instrument_id: 98, Instrument: violin, Brand: Gibson, Rented: false, Price: 200
, Instrument_id: 118, Instrument: piano, Brand: Gibson, Rented: false, Price: 200
, Instrument_id: 123, Instrument: clarinett, Brand: Gibson, Rented: false, Price: 200
, Instrument_id: 99, Instrument: saxophone, Brand: Gibson, Rented: false, Price: 200
, Instrument_id: 121, Instrument: guitar, Brand: Roland, Rented: false, Price: 200
, Instrument_id: 112, Instrument: fiol, Brand: Harman Professional, Rented: false, Price: 200
]
```

Figure 4: ls all



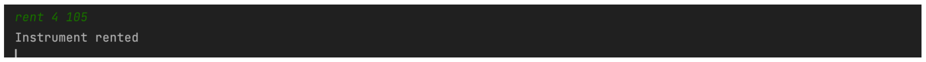
```
rent 3 102
The requested instrument with id 102 is not available
```

Figure 5: Requested instrument not available



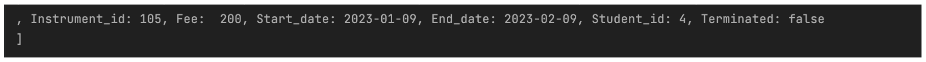
```
rent 1 102
Student already rents 2 instruments and therefore can't rent another one
```

Figure 6: Student already rents two instruments.



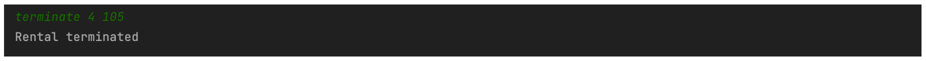
```
rent 4 105
Instrument rented
```

Figure 7: Instrument rented



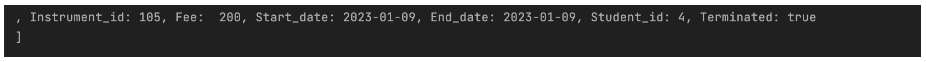
```
, Instrument_id: 105, Fee:  200, Start_date: 2023-01-09, End_date: 2023-02-09, Student_id: 4, Terminated: false
]
```

Figure 8: All information of the rented instrument



```
terminate 4 105
Rental terminated
```

Figure 9: Rental terminated



```
, Instrument_id: 105, Fee:  200, Start_date: 2023-01-09, End_date: 2023-01-09, Student_id: 4, Terminated: true
]
```

Figure 10: All information of the terminated rental

5 Discussion

The primary objective in designing the application for this assignment was to produce well-structured, easily understood code. This involved adhering to the conventions presented for naming, package organization, architectural patterns, and other guidelines provided throughout the course. The use of the MVC pattern facilitates the creation of a well-structured and maintainable application, as it clearly defines the roles and responsibilities of each component. Additionally, the use of layer patterns helps to further modularize the system by organizing packages, classes, and other components into logical layers based on their functionality.

All requirements from the task description is fulfilled. Terminate rentals is implemented as described in the method description.

The rollback functionality is implemented through exception handling in the program to ensure atomicity. In exception handling, a rollback is the process of undoing any changes made to the database during a transaction in the event that an exception is thrown. Transactions are a way to group multiple database operations together so that they are either all committed (applied to the database) or all rolled back (undone from the database) as a single unit of work.

If two deletion actions are executed simultaneously, it may result in the termination of the rental being initiated twice. This could lead to the termination being erroneously removed once and no change being made on the second occasion. To prevent this, the termination procedure within the controller could be improved by incorporating a check to determine if the rental has already been terminated. If it has, a message could be returned indicating that the rental has already been terminated.

In the event that two creation actions are performed simultaneously, it may lead to a situation where data isolation is compromised. This can be mitigated by implementing lock mechanisms which would enhance performance. However, the author has not yet had the opportunity to address this matter

The following link contains the git repository for the entire project.

Link to github repository: https://github.com/natashadonner/IV1351-Project