Database programming project: Connecting MySQL database with java and JDBC

Date: 15/05/2023

Group Members:

I.Vilá, UPV/EHU, Univ. of the Basque Country, Spain
J.Wang, UPV/EHU, Univ. of the Basque Country, Spain

ivila006@ikasle.ehu.eus
jwang005@ikasle.ehu.eus

Subject: Databases
Degree: Artificial Intelligence
Academic Year: 2023/2024
School: Faculty of Computer Science

Table of Contents

Abstract 3
1. Introduction 3
2. Project Structure 3
2.1. Classes design and explanation 3
2.2 TRANSACTIONS designed by each member of the group 4
2.3 Extra work 7
3. Notes of the Meetings 8
4. Conclusions 8
5. Bibliography and references 8

Abstract

This document presents the requirements, implementation, and outcomes of a database programming project conducted as part of the Databases' subject. The project focuses on creating and managing relational databases using MySQL and Java with JDBC. The main goal is to implement various transactions to showcase the databases' functionality and what we have learned during the course. Additionally, the project has encouraged collaborative teamwork and documentation practices.

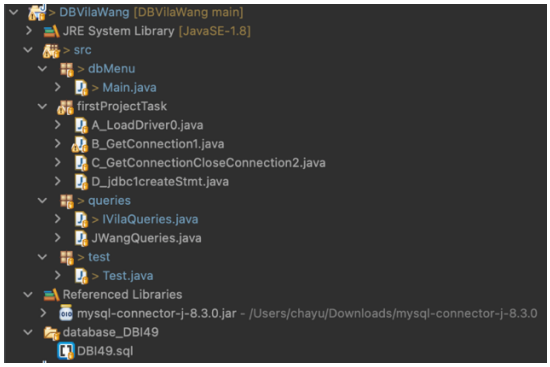
1. Introduction

In this final project, we have had to go through some steps until reaching the end. First, we thought about what transactions we wanted to implement. Then, we implemented those transactions, which was the hardest part due to problems and difficulties we encountered. And in the end, we designed the menu according to what we wanted it to look like. For all these tasks, we used some systems to help us, e.g. GitHub has helped us on sharing the code with each other easily, or also Microsoft Word has enabled us doing the documentation simultaneously.

Link to GitHub repository: https://github.com/JiayuanWX/JDBCProject

2. Project Structure

2.1. Classes design and explanation



The image above this text shows the structure of our Java project. It has two folders, the src/ and the database_DBI49/ folder. This second folder is used to save the whole database we have used for the implementation and testing of the project. Both of us have used the same database.

On the other hand, the main folder, src/, has 4 different packages that are the following ones:

1. dbMenu: This package encapsulates the functionality related to the application's console menu. It contains a main class responsible for generating and managing the user interface, which is very useful to prove each of the queries we have made. For using this menu you'll have to run the class, then the system will print the description of each.
2. firstProjectTask: Here, we put the first steps of learning about how to connect to de database with JBDC. It deals with setting up the database, connecting to it, and doing simple tasks just to ensure that the connection works correctly.
3. queries: This package is composed of two classes, JWangQueries and IVilaQueries. In each class each of us has designed different queries that apply many of the concepts we have learned during the course.
4. test: In this last package, we have made a test class to make sure our queries work correctly and retrieve the expected data without raising any exception. It's where we check if everything is doing what it's supposed to do.

2.2 TRANSACTIONS designed by each member of the group

For the development and creation of the following transactions, we have used COMPANY_TRAVEL, WORLDDB, RESTAURANT, TRAVEL and MOVIES schemas. We have used many different schemas because each schema offers distinct datasets and structures, allowing us to explore a diverse range of scenarios and challenges in our transaction implementations.

2.2.1 Transactions of Iñigo Vilá

2.2.1.1. Retrieve the employee with the highest salary among the TWO employees who have the most female dependents and no male dependents. - *CompanyDB*

```
SELECT e.Ssn, e.Fname, e.Lname, e.Salary
FROM employee e
JOIN (
    SELECT e1.Ssn
    FROM employee e1
    JOIN dependent d ON e1.Ssn = d.Essn
    WHERE d.Sex = 'F'
    GROUP BY e1.Ssn
    HAVING COUNT(*) = (
        SELECT MAX(dependent_count)
        FROM (
            SELECT COUNT(*) AS dependent_count
            FROM dependent
            WHERE Essn = e1.Ssn
            GROUP BY Essn
        ) AS subquery
    )
    ORDER BY COUNT(*) DESC
    LIMIT 2
) AS top_employees ON e.Ssn = top_employees.Ssn
ORDER BY e.Salary DESC
LIMIT 1;
```

2.2.1.2. For each city find the hotel most chosen among all customers, jointly with the number of times that it has been chosen. - *TravelDB*

```
SELECT h1.hotelcity, h1.hotelname, COUNT(distinct htc1.CustomerId) AS
custs
FROM hotel_trip_customer AS htc1
NATURAL JOIN hotel AS h1
GROUP BY h1.hotelcity, h1.hotelname
HAVING COUNT(distinct htc1.CustomerId) >= ALL (
    SELECT DISTINCT COUNT(distinct htc2.CustomerId)
    FROM hotel_trip_customer AS htc2
    NATURAL JOIN hotel AS h2
    WHERE h1.hotelcity = h2.hotelcity
    GROUP BY h2.hotelcity, h2.hotelname
);
```

2.2.1.3. Return the customer ID's, name and count of different trip destinations of those customers that have gone on at least the same number of trips than the employee that works the least hours. – *Company_Travel DB*

```
SELECT CustomerId, c.custname, COUNT(distinct htc1.TripTo) AS
distinctTrips
FROM hotel_trip_customer htc1
NATURAL JOIN customer c
GROUP BY htc1.CustomerId
HAVING COUNT(distinct htc1.TripTo) = (
    SELECT COUNT(DISTINCT TripTo)
    FROM hotel_trip_customer htc2
    JOIN employee_customer ON htc2.CustomerId =
employee_customer.Cust_Id
    WHERE employee_customer.Emp_id IN (
        SELECT e.Ssn
        FROM employee e
        JOIN works_on w ON e.Ssn = w.Essn
        GROUP BY e.Ssn
        HAVING SUM(w.Hours) <= ALL(
            SELECT SUM(w2.Hours) AS total_hours
            FROM employee e2
            JOIN works_on w2 ON e2.Ssn = w2.Essn
            GROUP BY e2.Ssn
        )
    )
);
```

2.2.1.4. INSERT INTO dependent a new dependent for the employee with ssn 999999999 – CompanyDB

```
INSERT INTO dependent (Essn, Dependent_name, Sex, Bdate, Relationship)
VALUES ('999999999', 'Jack Jr.', 'M', '2010-05-15', 'Son')
```

2.2.1.5. UPDATE num nights from 5 to 10 for the customer with customerId 10000025 - *TravelDB*

```
UPDATE hotel_trip_customer SET NumNights = 10 WHERE customerid =
10000025 AND TripTo = 'Hong Kong' AND DepartureDate = '2018-01-05' AND
hotelid = 'h07' AND NumNights = 5
```

2.2.2 Transactions of Jiayuan Wang

2.2.2.1. Retrieve the rating of the restaurants that serve ‘cheesecake’ or ‘pepperonipizza’ which has at least one daily sale of an amount greater than 10000 but less than 20000. – *RestaurantsDB*

```
SELECT DISTINCT r.restaurname, r.city, r.rating

FROM restaurant AS r
WHERE r.restaurname IN (
    SELECT s.restaurname
    FROM serves AS s
    JOIN dishes d ON s.dish = d.dish
    WHERE d.dish = 'cheesecake' OR d.dish = 'pepperonipizza'
)
AND EXISTS (
      SELECT s.restaurname
      FROM sales AS s
```

```
      WHERE r.restaurname = s.restaurname AND s.amount BETWEEN 10000
         AND 20000
)
AND r.restaurname IN (
      SELECT f.restaurname
      FROM frequents f
      JOIN person p ON f.nameId = p.nameId
      WHERE p.nameId IN ('Hillary', 'Alexander')
);
```

2.2.2.2. Retrieve the Fname, Lname, salary, and nº of hotels visited, for employees who are also customers, have not visited 'Biarritz' and earn more or equal to 30000, sorted by salary. – *Company_Travel DB*

```
SELECT e.Fname, e.Lname, e.Salary, COUNT(htc.HotelId) AS 'numHotels'
FROM (employee AS e  RIGHT JOIN employee_customer AS ec ON e.Ssn =
ec.Emp_id)
      LEFT  JOIN  hotel_trip_customer  AS  htc  ON  ec.Cust_Id  =
htc.CustomerId
WHERE NOT EXISTS(
      SELECT *
    FROM hotel_trip_customer AS htc2
    WHERE htc2.TripTo = 'Biarritz' AND htc2.CustomerId = htc.CustomerId
)
GROUP BY e.Fname, e.Lname, e.Salary
HAVING e.Salary >= 30000
ORDER BY e.Salary ASC;
```

2.2.2.3. List the genres where ALL the directors have directed at least a movie of that genre. – *Movies DB*

```
SELECT DISTINCT mg.genre
FROM movies_genres mg
WHERE NOT EXISTS (
    SELECT d.id
    FROM directors d
    WHERE NOT EXISTS (
        SELECT md.movie_id
        FROM movies_directors md
        WHERE md.director_id = d.id
        AND md.movie_id IN (
            SELECT mg2.movie_id
            FROM movies_genres mg2
            WHERE mg2.genre = mg.genre
        )
    )
);
```

2.2.2.4. INSERT INTO serves of 'Dish2Eat' restaurants the 'tortilla' dish. – *RestaurantsDB*

```
INSERT INTO serves (restaurname, dish, price) VALUES ('Dish2Eat',
'tortilla', '2.50')
```

2.2.2.5. UPDATE languages of the guide of Id:72515633, from English to Chinese and from French to Spanish. – *Travel DB*

```
UPDATE languages SET Lang = 'Chinese' WHERE GuideId = '72515633' AND
Lang = 'English'

UPDATE languages SET Lang = 'Spanish' WHERE GuideId = '72515633' AND
Lang = 'French'
```

2.3 Extra work

2.3.1 Extra Queries of Iñigo

2.3.1.1. For each language show the country with most population ordered by descending country population. (only first 25 results) - *WorldDB*

```
SELECT cl.Language, c.Name AS CountryName, c.Population
FROM countrylanguage cl
JOIN country c ON cl.CountryCode = c.Code
WHERE c.Population =(
    SELECT MAX(c2.Population)
    FROM countrylanguage cl2
    JOIN country c2 ON cl2.CountryCode = c2.Code
    WHERE cl2.Language = cl.Language
);
```

2.3.1.2. Retrieve the name of the clients that have visited all restaurants of the city of New York - *RestaurantsDB*

```
SELECT p.nameId, COUNT(DISTINCT r.restaurname) AS
NewYorkRestaurantsVisited
FROM person p
JOIN frequents f ON p.nameId = f.nameId
JOIN restaurant r ON f.restaurname = r.restaurname
WHERE r.city = 'New York'
GROUP BY p.nameId
HAVING NewYorkRestaurantsVisited = (
    SELECT COUNT(*)
    FROM restaurant
    WHERE city = 'New York'
);
```

2.3.2 Extra Queries of Jiayuan

2.3.2.1. Retrieve the TripTo, DepartureDate, and totalNights of the trips that across all customers have more totalNights than the average. - *TravelDB*

```
SELECT htc.TripTo, htc.DepartureDate, SUM(htc.NumNights) AS totalNights
FROM hotel_trip_customer AS htc
GROUP BY htc.TripTo, htc.DepartureDate
HAVING totalNights > (
    SELECT AVG(bookedNights)
    FROM (
        SELECT SUM(NumNights) AS bookedNights
        FROM hotel_trip_customer
        GROUP BY TripTo, DepartureDate
    ) AS nightsPerTrip
);
```

2.3.2.2. Print the difference of dishesnames between the person that has eaten the most dishes and the least. – *RestaurantDB*

```
SELECT DISTINCT maxE.dish
FROM eats AS maxE
WHERE maxE.nameId = (
    SELECT e1.nameId
    FROM eats AS e1
    GROUP BY e1.nameId
    ORDER BY COUNT(e1.dish) DESC
    LIMIT 1
)
```

```
AND maxE.dish NOT IN (
    SELECT minE.dish
    FROM eats minE
    WHERE minE.nameId = (
        SELECT e2.nameId
        FROM eats e2
        GROUP BY e2.nameId
        ORDER BY COUNT(e2.dish) ASC
        LIMIT 1
    )
);
```

3. Notes of the Meetings

This project has been completely developed through face-to-face meetings, in this way we have been able to share all our knowledge and in case of any doubt or error we have been able to help each other in the search for solutions. The first meeting was used to think about what each of us was going to design and to start with the structure of the project. Then we started to design all the queries and test them to make sure they returned the expected data. Finally, we spent our time to create this documentation that shows how we developed this project.

4. Conclusions

These types of projects are great for us to apply the knowledge acquired during the course. As we have said before during the development of the project, we have had to solve several problems, this search for solutions has helped us to learn more about SQL and the use of JBDC.

In addition, being a group work we have been able to expand our knowledge on GitHub, which is an essential platform for this type of collaborative projects.

As a final detail we would like to point out that besides having implemented each of us two extra queries, to work with more schemas and different types of queries, we have also designed insert/updates that raise different exceptions in order to demonstrate the use of ssavepoints and rollbacks, which are so useful for maintaining the integrity and consistency of the database.

5. Bibliography and references

J. Dolado. (2024). *Exercises TRAVEL Database SQL Solutions* [Slides]. Computer Science Faculty, University of the Basque Country. https://egela.ehu.eus/pluginfile.php/8090753/mod_resource/content/24/W7-EX_%20TRAVEL_SQL_Solutions.pdf

J. Dolado. (2024). *TRANSACTIONS and JDBC* [Slides]. Computer Science Faculty, University of the Basque Country. https://egela.ehu.eus/pluginfile.php/8090845/mod_resource/content/5/W13-01_TransactionsJDBC.pdf

Barry Brown. (2013, 14 abril). *Database Transactions, part 3: ACID and Isolation* [Vídeo]. YouTube. https://www.youtube.com/watch?v=NHKHzwolbKU