Middle East Technical University | Department of Computer Engineering

CENG 351
Data Management and File Structures

Fall 2023-2024

Programming Assignment 1

Due date: 17 December 2023, Sunday, 23:59
Submission: **via ODTUClass**

1 Introduction

In this assignment, you are asked to create a system by designing queries and implementing pre-defined functions to operate on a database for a factory CengFactory. You will be provided with specific tasks and a well-defined interface. This assignment aims to help you get familiar with connecting and querying to MySQL Server using Java Database Connectivity (JDBC).

All necessary data files, classes, and the interface you will implement are provided to you in the source files. Do not confuse interface with graphical user interface (GUI). Interface is an abstract type in Java used to specify a behavior that classes must implement. (If you remember from the Programming Language Concepts course, a Java interface corresponds to a C++ class and all of its methods are purely virtual.)

You should first implement functions that create the necessary tables corresponding to the schema given in Section 2. Then, you should design queries to accomplish the given tasks. Lastly, you should implement the interface using the queries you have designed as they give the desired results when defined parameters are given. You will not implement the Evaluation class. It will be implemented by us to manipulate the database through the predefined interface and evaluate your implementations. Your task is to build up class(es) that implement(s) the provided interface.

2 Schema

You will use (strictly) the schema given below in the scope of this assignment:

- **Factory**(factoryId:int, factoryName:Text, factoryType:Text, country: Text)
- **Employee**(employeeId:int, employeeName:Text, department:Text, salary: int)
- **Works_In**(factoryId:int, employeeId:int, startDate: Date)
- **Product**(productId:int, productName: Text, productType: Text)
- **Produce**(factoryId:int, productId:int, amount: int, productionCost:int)
- **Shipment**(factoryId:int, productId:int, amount: int, pricePerUnit:int)

Factory table holds the factoryId, factoryName, factoryType, and the country where the factory is located. An example factory is as follows:
`factoryId: 101, factoryName: Nestle Factory, factoryType: Food, country: Sweden`

Employee table holds the employeeId, employeeName, department and the salary of the employees. An example employee is as follows:
`employeeId: 1000, employeeName: John Smith, department: Sales, salary: 100000`

Works_In table holds the factoryId, employeeId, and startDate. An example works_in row is as follows:
`factoryId: 101, employeeId: 1000, startDate: 10-10-2020`

Product table holds the productId, productName, and productType. An example product is as follows:
`productId: 20, productName: Nestle Chocolate, productType: Food`

Produce table holds the factoryId, productId, amount and productionCost. An example produce is as follows:
`factoryId: 101, productId: 20, amount: 1000, productionCost: 10`

Shipment table holds the factoryId, productId, amount and the pricePerUnit. An example shipment is as follows:
`factoryId: 101, productId: 20, amount: 500, pricePerUnit:18`

Your task is to generate a class named `CENGFACTORYDB` (it should belong to package `ceng.ceng351.cengfactorydb`) which implements `ICENGFACTORYDB` interface. You can create any additional classes if necessary. `CENGFACTORYDB` class should be able to accomplish the following tasks:

- Create the database tables
- Insert data into tables
- Find all factories that are located in a particular country.
- Find factories without any working employees.
- Find factories that produce all products for a particular productType
- Find the products that are produced in a particular factory but don't have any shipment from that factory.
- For a given factoryId and department, find the average salary of the employees working in that factory and that specific department.
- For each factory find the most profitable products. Consider the amount in profit calculation
- For each product, find the factories that gather the highest profit for that product
- For each department, find all employees that are paid under the average salary for that department. You consider the employees that do not work earning "0"
- For the products of a given productType, increase the productionCost of every unit by a given percentage
- Deleting all employees that have not worked since the given date.
- Dropping the database tables

Tasks are explained in more detail in the Tasks section. For each task, there is a corresponding method in `ICENGFACTORYDB` interface. You need to implement them in `CENGFACTORYDB` class. Necessary data files (for populating the tables) to accomplish these tasks are provided.

In the `data` folder there are 6 `.txt` files that correspond to each table. You will use these tables when you are inserting data. Data files, interfaces, and classes for fulfilling these tasks will be provided as source files. You can assume all information will be complete and consistent, i.e. all necessary data will be inserted before executing a query. You can find detailed descriptions of the usage of the functions in the provided source files. Your results should not include any repetition. Therefore, please do not forget to use the `DISTINCT` keyword when appropriate in your queries.

Make sure you are using Java (version 14) before starting.

3 Tasks

3.1 Creating the database tables

You will create all the tables according to the schema described in Section 2. You can assume that tables will be created before executing any other database operation. Do not forget to define **foreign keys** while you are creating tables.
Method: `createTables()`
Output: The number of tables that are created successfully.

3.2 Inserting data into tables

You will insert data into appropriate tables.
Methods: `insertFactories()`, `insertEmployees()`, `insertWorksIn()`, `insertProduct()`, `insertProduce()`, `insertShipment()`
Output: The number of rows inserted successfully.

3.3 Find all factories that are located in a particular country.

Method(Input): `getFactoriesForGivenCountry(country)`
Output: factoryId, factoryName, factoryType, country
Order the results by ascending factoryId

3.4 Find factories without any working employees.

Method: `getFactoriesWithoutAnyEmployee()`
Output: factoryId, factoryName, factoryType, country
Order the results by ascending factoryId

3.5 Find factories that produce all products for a particular productType

Method: `getFactoriesProducingAllForGivenType(productType)`
Output: factoryId, factoryName, factoryType, country
Order the results by ascending factoryId

3.6 Find the products that are produced in a particular factory but don't have any shipment from that factory.

Method: `getProductsProducedNotShipped()`
Output: productId, productName, productType Order the results by ascending productId

3.7 For a given factoryId and department, find the average salary of the employees working in that factory and that specific department.

Method: `getAverageSalaryForFactoryDepartment(factoryId, department)`
Output: averageSalary

3.8 For each factory find the most profitable products. Consider the amount in profit calculation

Profit = shipmentAmount * pricePerUnit - producedAmount * productionCost.
Method: `getMostValueableProducts()`
Output: factoryId, productId, productName, productType, profit Order the results by descending profit, ascending factoryId

3.9 For each product, find the factories that gather the highest profit for that product

Profit = shipmentAmount * pricePerUnit - producedAmount * productionCost.
Method: `getMostValueableProducts()`
Output: factoryId, productId, productName, productType, profit Order the results by descending profit, ascending productId

3.10 For each department, find all employees that are paid under the average salary for that department. You consider the employees that do not work earning "0"

Method: `getLowSalaryEmployeesForDepartments()`
Output: employeeId, employeeName, department, salary Order the results by ascending employeeId

3.11 For the products of given productType, increase the productionCost of every unit by a given percentage

Method(Inputs): `increaseCost(productType, percentage)`
Output: number of rows affected

3.12 Deleting all employees that have not started working since the given date.

Method(Input): `deleteNotWorkingEmployees(givenDate)`
Output: number of rows affected

3.13 Dropping the database tables

You will drop all the tables (if they exist).
Method: `dropTables()`
Output: number of tables that are dropped successfully

4 Extra Questions

Note that these questions **will not be graded** and are not in the scope of this course (might be helpful for the ones who are interested in further knowledge). Also, note that you should **investigate further operations** to solve the given questions. You may check MySQL Window Functions, for questions 4.1-4.4 (let me know in case you have any problems.)

4.1 For each department, find the rank of the employees in terms of salary. Peers are considered tied and receive the same rank. After that, there should be a gap.

Method: `calculateRank()`
Output: employeeId, department, salary, rank
Order the deleted rows by ascending department, ascending rank

4.2 For each department, find the rank of the employees in terms of salary. Everything is the same but after ties, there should be no gap.

Method: `calculateRank2()`
Output: employeeId, department, salary, rank
Order the deleted rows by ascending department, ascending rank

4.3 For each factory, calculate the most profitable 4^{th} product.

Method: `calculateFourth()`
Output: factoryId, productId, profit
Order the deleted rows by ascending factoryId

4.4 Determine the salary variance between an employee and another one who began working immediately after the first employee (by startDate), for all employees.

Method: `calculateVariance()`
Output: employeeId, startDate, salary, salaryDifference
Order the deleted rows by ascending startDate, ascending employeeId

4.5 Create a method that is called once and whenever a Product starts losing money, deletes it from Produce table.

Method: `deleteLosing()`
Output: Nothing.

5 Regulations

- **Programming Language:** Java (version 14).
- **Database:** An account on the MySQL server on a remote machine will be created for each of you and an e-mail including credentials and connection configuration will be sent to your metumail. You must use the JDBC driver to connect to the database. Your final submission must connect to the MySQL server on the remote machine. So, make sure that the connection information is correct before submitting your homework.
- **Attachments:** Necessary source files and JDBC driver is provided.
- **Input:** All strings will be case-sensitive and they will not include any non-English characters. Note that they may include punctuation.
- **Cheating:** We have a zero tolerance policy for cheating. People involved in cheating will be punished according to university regulations.
- **Evaluation:** It is GUARANTEED that input files are correctly formatted and sample data will be given to you. There will be no surprises about the data, similar (and larger) data will be used while evaluating the homework. Your program will be evaluated automatically using the "black-box" technique so make sure to obey the specifications. **Please, note that you have to accomplish tasks only within your SQL queries, not with any other Java programming facilities.**

6 Submission

Submission will be done via ODTUClass. Create a compressed file named `ceng.tar.gz` that contains `CENGFACTORYDB.java` and all other classes created by yourself. You will not submit the interface and class files provided by us. So, be sure you do not modify them during implementation. The compressed file should contain a directory tree same as the package tree. That is, you should compress the directory named `ceng` which contains a directory named `ceng351` which contains a directory named `cengfactorydb` which contains your source files. The directory tree should look as follows:

```
ceng
└── ceng351
    └── cengfactorydb
        ├── CENGFACTORYDB.java
        ├── AnotherClassIfYouNeedIt1.java
        ├── AnotherClassIfYouNeedIt2.java
        ├── ...
        └── AnotherClassIfYouNeedItN.java
```

Important note: Make sure your project compiles outside of the IDE by adapting and executing the following command:

```
javac -classpath .:/path/to/mysql-connector-java-8.0.11.jar CENGFACTORYDB.java
ICENGFACTORYDB.java Evaluation.java FileOperations.java QueryResult.java Factory.java
Employee.java Works_In.java Product.java Produce.java Shipment.java
AnotherClassIfYouNeedItN.java
```