ASSIGNMENT 6 – Database Modification

Overview:

The problem statement states that, the northwind database needs to be modified so that the data is maintained and managed in the database when the inventory is ordered from the suppliers.

Classes:

- ConnectDatabase: Class that takes credentials from Identity class and invokes the database connectivity method wherein a connection is established with the database.
- MyIdentity: Class that holds the database name, username and password.
- Product: Class that holds the product details (the tracking id , unit price and quantity)
- OrderException : Class that extends the Exception class and hands the exceptions thrown in the program
- Operations: Class that performs the required operations like shipping the order, issuing the reorder and receive orders

Interface:

- inventoryControl : Interface that holds three methods that are implemented in Operations class.

Functions:

- **ConnectDatabase:**

connectDB : Method that establishes a connection with the database provided by the user along with his credentials and invokes the required methods where summary is fetched.

- **MyIdentity:**

setIdentity : Method that holds the database name, the username and password to access the database.

- **Product:**

Product: Constructor that holds the product details including the tracking id, quantity and unit price.

- **Operations:**

1. Ship_order: Method that ships the given order number and reduces the inventory as per the ordered quantity. Based on the order id given as input, if is checked if the order is already shipped, if not the shipped date is updated as received date -1 in the table and based on the quantity and units in stock the inventory in the products table is reduced. Purposely, the shipped date is not updated as current date as it can cross the received date.

2. Issue_reorders: Method to issue reorders based on the given date. Based on the given date as input, all the products whose units in stock is less than the reorder level is fetched from the database along with the supplier id. All the data fetched is organized into two maps, including the supplier product details and product data. The quantity to reorder is taken the double of the

reorder level and the unit price is taken as the last sale price for that day with 15% markup. All the calculated and fetched data is inserted into two tables, holding the supplier product details and the supplier orders. Later, based on the data inserted on that particular day, the count of the suppliers is returned. This function internally calls the set_price function.

3. set_price: Method that fetches the last sale price from the orders table and arranges the fetched data in a map with key value as product id and value as the last sale price for that product. This function internally calls the arrange data function.

4. arrange_data: Method that accesses the result set obtained from the database and organizes data using two maps, one with key values as supplier id , values as product id and other with key values as product id and values as product details from the Product class, fetched from the database.

5. Receive_order: Method that delivers the order based on the given internal_order_reference given as input updates the inventory with the new stock. Based on the reference number given as input, the stored tables are accessed and for all the orders which can be considered to be received, the arrival date is updated as current date in the tables. At the same time, for each of the product that was ordered under that reference id the inventory is increased in the products table as per the stock.

Approach to the problem:

In order to deal with the discrepancies in the existing database, a basic design is made to connect the scenario.

Since there is no supplier order data at all, a table holding the orders to the suppliers along with the price and quantity is decided to maintain.

Next, every order should be delivered by some logistics, so another table where the delivery id and logistics is created. The supplier can add his linked logistics to this table along with the delivery id so that the by the whom the order is delivered is known.

To normalize the database, the supplier product details table is broken into other table with tracking id supplier id and orderdate as main columns, so that it will be easy to fetch the product details based on the input date for a particular supplier.

Once the design is fixed, the required tables are created with the proper primary keys and foreign keys.

Then based on the requirement categorized under each function, the major queries are formed and executed on workbench to check if the required outcomes are matched.

Later, the java class that implements the interface is formed holding the inline queries and fetching the required outcome for each one of the functions. Respective classes are designed, each of the one holding its own functionality.

Code is completely modularized so that it will be easy to understand the workflow. Internal comments are clear and concise so that the code is understandable.

Each of the module is tested as per the functionality and compared with the required output.

Required exceptional handling is done so that and exception is thrown if there is an error related to that module. Custom exceptions are created so that appropriate exception is thrown when required within the functionality. Also, the sql exceptions are handled at required positions.

Testing/Argument:

Ship_orders:

For the given order id, the inventory should be reduced based on the quantity ordered and units in stock. The product table for those products under that particular order are verified before executing the program and after executing the program, which showed the difference in the stock.

Order id not in the tables : exception is thrown

Order id already shipped : exception is thrown

Issue_reorders:

For the given date, all the order data should be inserted into the new tables. The supplier_orders table and supplier_product_details is checked before and after executing the function to check if the order details for the supplier are inserted into the table. Also the unit price for a product is compared with the last sale price on that day with the 15% markup. And the quantity is taken as double the reorder level. All these are compared to the product in the product table.

Receive_orders:

For the given tracking id, the data is fetched from the respective supplier_product_details table so that inventory is updated in the product table based on the details saved. The product in the product table that is ordered under the given reference id is checked before and after executing the function to check if the units in stock are increased as required in the product table.

Reference id not in the tables : exception is thrown with the reference number

For the given reference id, if the ordered are already received : exception is thrown with the reference number.

The program operates as mentioned in the problem statement and works for the set of existing records. When compared to the practical scenario all the data that needs to be maintained by the supplier is designed in the database and maintained.

The whole design is made comparing with a normal store that generally operates which delivers orders to the customers and receives orders from its suppliers.

Finally, the whole program is following the single responsibility principle where a single class is responsible for the reissuing, shipping and receiving orders for the suppliers.

The code is modularized so that it is understandable and extendable by any other programmer for enhancing the existing design.

The code is easy to read and maintain as it does not make use of much of the memory and satisfies the functionality of the problem statement.

Each of the feature once implemented is tested vigorously under all the circumstances. All the queries are tested on workbench initially until the exact outcome is fetched. Several validations are done so that code is error free

Assumptions:

The shipped date is null for few records, so it is assumed that those are the only ones which needs to be shipped.

For the records whose shipped date is already updated, it is assumed that the inventory is updated as required based on the stock received from the suppliers without maintaining any data.

For the orders whose shipped date is null, once the order is shipped, the date is updated as the orders' required date -1 as the required date for those orders is few years back and it would be no use if we take the current date as shipped date which crosses the required date mentioned in the tables.

The suppliers should update their delivery information in the respective table based on the linked logistics along with their delivery id to keep track of the orders.

The tracking id is autoincrementing in the tables and starting from 1.

The inputs in the issue_reorders method are assumed to be given with respect to the date format where year is of four digits and month and date can be one or two digits.

References:

For understanding the the designing of er diagram : Class videos from collaborate and

https://www.lucidchart.com/pages/er-diagrams

For understanding normalization : Slides from the class along with the text book material.