MALMÖ UNIVERSITET

Online Store

Application in C# with pgAdmin4

Martin Guvå

Monika Sliauteryte

VT2023 - DA297A

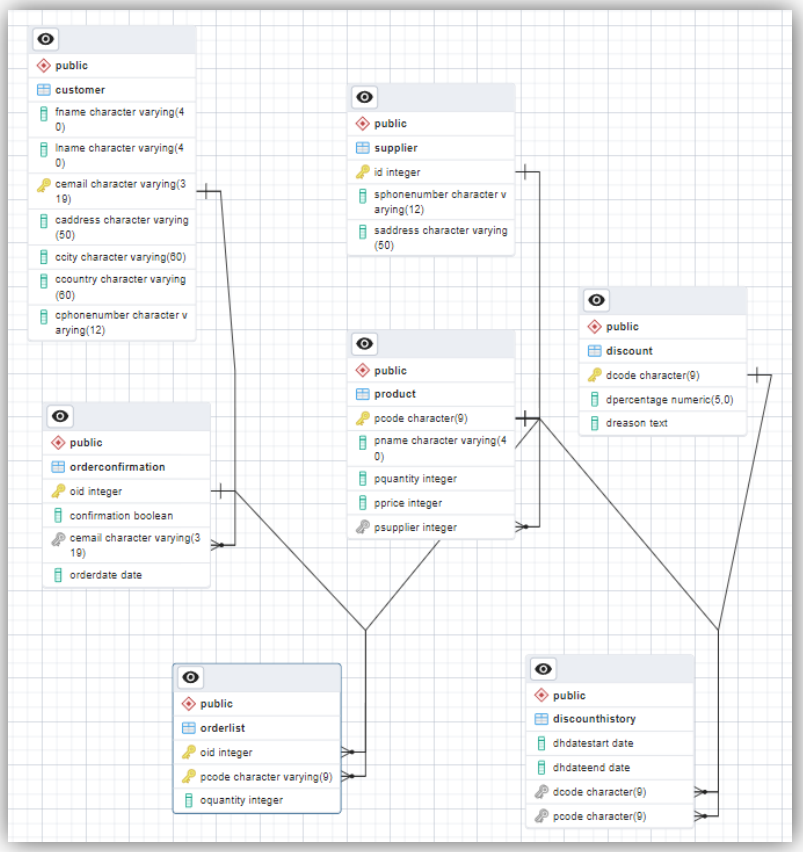
Innehåll

Intro

An online store asks for an application to keep track of their products, prices, discounts, sales status, and customers.

Database Design

An overall graphical diagram representing database design can be seen in picture 1. We chose 7 tables to make whole architecture.



- Customer (first name, last name, email (PK), Address, City, Country, Phone nr). Table handles information about customer on registration and is later used as to create and identify orders.
- Supplier (Id (PK), Phone number, Address).This table handles information about supplier and holds important information about every supplier. This table has relation to products as many products can belong to one supplier.
- Product (Code (PK), Name, Quantity, Price, Supplier (FK to Supplier). Table responsible to manage products and products belonging to suppliers. One product per unique Id can belong to only one supplier. Products can have same names, prices, quantities.
- Discount (Code (PK), Percentage, Reason). Table used for discount information and is used later while adding discounts to particular product. Discount information can vary, we chose to include percentage and be it permanent in our case. Date when discount is used varies depending on product and can be assigned later.
- Discount History (Start date, End date, Discount Code (FK to Discount), Product Code (FK to Product). Table hols information about applied discounts on particular product. Start and end date is assigned per product, discount is chosen from discount table.
- Order Confirmation (Id (PK), Confirmation, Email (FK to Customer), Order Date). Table holds base information about order when it was created. It has default number which is automatically assigned during order creation process. The same is valid for date – default as Today.
- Order List (Order Id(FK to Order confirmation), Product code (FK to Product), Quantity, PK(Order Id, Product Code). Table keeps track of all orders and their products.

Constrains in tables contains Primary Keys and Foreign Keys (following rules that PK is unique and not null) and all important information must be presented (example: quantity of products). Important (not all) constrains to mention includes:

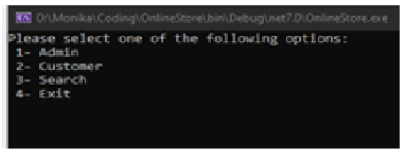
- quantity of product cannot be smaller than 0;
- discount end date cannot be earlier than discount start date (and vs);
- default values on dates;
- …

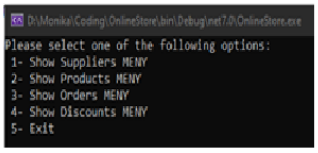
Database connectivity and application architecture

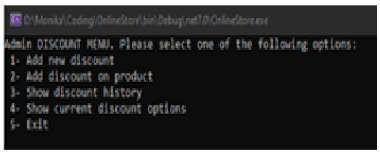
To connect DB to application we used Npgsql library which enabled us to use our db servers with passwords and usernames. Every method in Meny classes is calling databaseconnect class methods. Methods are reused when it was needed and possible (like some search menus for Admin and Customer).

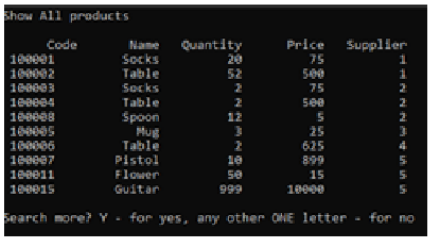
User interface

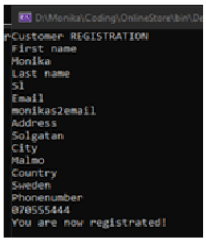
We used simple Console application and programed in C# using Visual Studio. It has 3 main menus: Admin, Customer and Guest. Admin / Customer then have separated menus according to task they can perform. There are few examples:

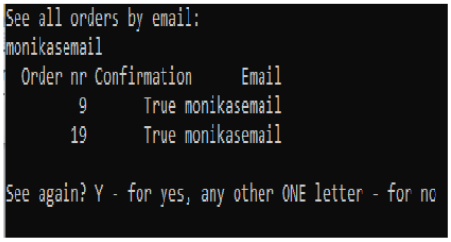






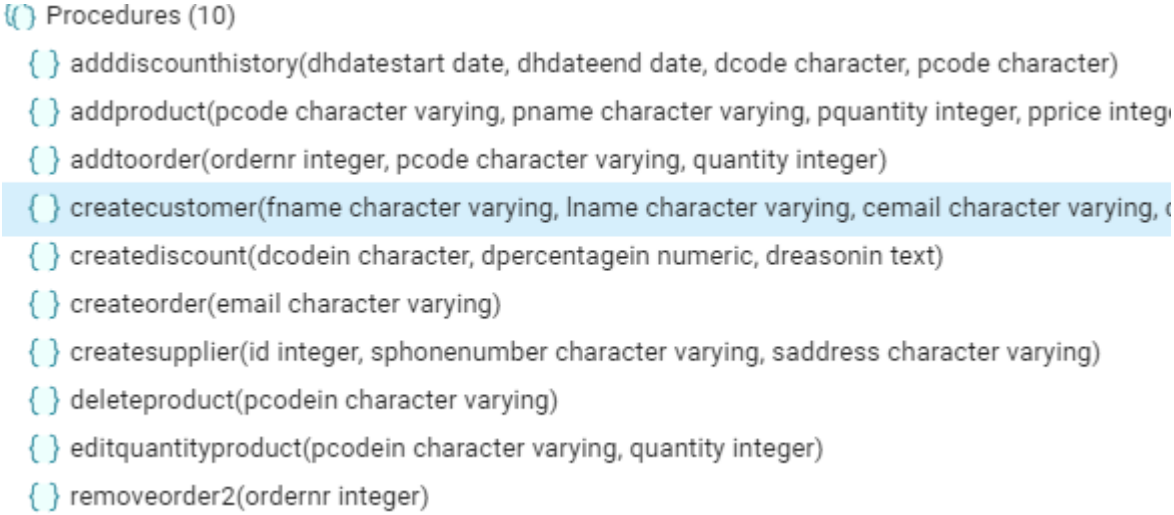






Procedures, Functions, Index and Trigger

We use a lot of stored procedures and functions to make implementation easy. As we will go deeper on them in other paragraph, here is an overview.



Triggers

Triggers were used to make product quantity redundant. They were implemented on product table as Orders were made – products added to orders; as well as orders were deleted.

Index

In consideration that our database is small, we still used index on sorting our product by names. It does not give any visible effect, but it was still implemented.

Application in details

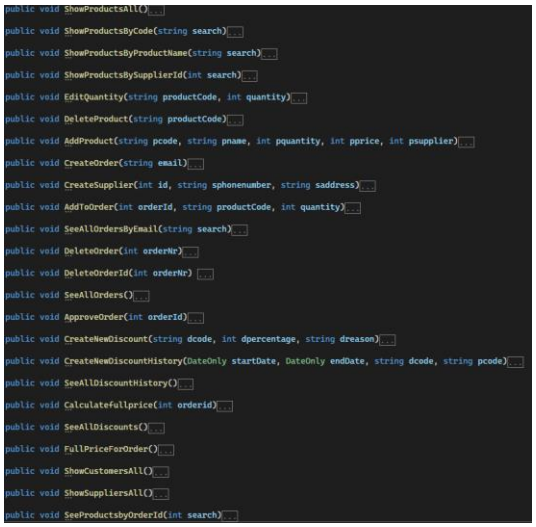
Constrains and input controll in short

It includes all checks on input as well gives messages when BD constrains are failing:

- It is not possible to delete product if product still is in unapproved list
- It is not possible to remove order by customer if order is already approved by admin
- It is not possible to add product to non-existing order;
- It is not possible to add non-existing product to order;
- It is not possible to create customer with same email address
- All values like strings instead of integers makes program to send a user a message that input is not correct
- If order was deleted – all products of that order will be deleted from order list and quantity of products restored (using triggers).
- And more…

All commands

The picture below shows all commands we were using which contains either calling procedure or function or sending SQL command.



Most interesting cases

We want to highlight few interesting cases.

- Search / Show – most search cases we manage to use direct sql sentences to send to db. We used string formatting to show information as a table.

  Ex: ShowProductByProductName is calling simple sql string

  `string sql = $"SELECT * FROM Product WHERE pname = '" + search + "'order by psupplier";` But in this case we have index usage which should (if db would be big) help us to find product by name faster.

- Create – used the same principle on creating customer, supplier, product, order, discount.

  CreateCustomer is calling procedure CreateCustomer().

  `string sql = "call CreateCustomer(@Fname, @Lname, @Cemail, @Caddress, @Ccity, @Ccountry, @CphoneNumber)";` In application we collect input data and send it as procedures arguments. We collect data in format is needed for parameters of the procedure.

  OBS. Solving wrong input we (especially in case of int input – where input was not an integer) using try {} and catch{} commands.

  The same method of control, and user friendly message when it fails, is used if user tries to create anything ex. supplier with the same ID (or any other PK or any other constrain),

- Add – as user adds new products to list or new orders. Was mainly used to call as stored procedure. It was used same try{} and catch{} elements to make sure that program doesn't crash if tables constrains aren't met – the control was handled by constrains on columns in db, but message was handled by application.

  AddProduct() was one of interesting cases where we used triggers on table OrderList and command insert. The trigger function was created to decrease the amount from Product table.

```
begin
if (select pquantity from product where pcode = new.pcode) -
new.oquantity > -1 then
update product
set pquantity = pquantity - new.oquantity
where pcode = new.pcode;
end if;
raise notice '%' , new;
return null;
```

- Delete – used to delete products, orders. The same principle we used to add the deleted amount back to products. Here could be interesting case to use transactions on order delete operation because user choses to delete whole order by order nr, but because it contains products in orderlist table belonging to that particular order, we used 2 methods (2 sql commans) to delete first all products from order list table (and calling trigger to add back products to products list) and then, when it did not contained products on that order – the order was deleted from order confirmation list. Transaction would make sure that all of it is done before commiting.
- Longest commands ☺ To select discounts history we have quite long command.

```
string sql1 = "select p.pcode, p.pname, d.dhdatestart, d.dhdateend,
disc.dreason, p.pprice, disc.dpercentage from product as p join
discounthistory as d on p.pcode = d.pcode join discount as disc on
disc.dcode = d.dcode join orderlist as o on p.pcode = o.pcode";
```

Conclusion

We created simple but full functioning application which online connection to database and used newly gained SQL knowledge to make sure db is redundant.