Databases and information management

Lab 2

Piotr Syga

October 21, 2022

1. Create `warehouse` database. Create two users[1] – *WAdmin* and *U*, where *U* is your name concatenated with 3 last digits of your index number. Set the passwords for both users (remember about securing the passwords). Give *WAdmin* full grants to the `warehouse` database and no privileges outside of it. Give *U* sufficient privileges to select from the whole `warehouse` and `northwind` databases, to modify the contents of the tables in `warehouse` database, but no privileges to create or modify tables nor views.
   For the rest of the tasks switch between all the users in your DBMS so that you're able to do the tasks, yet use the lowest level of access possible. Note which user was used for which task.

2. Create the tables below in the `warehouse` database. Please note that underline denotes the primary key of the table (it may be a composite PK):

   - `people` (id:int, firstname:varchar(40), lastname:varchar(40), title:varchar(10), address: varchar(70), city:varchar(30), country:varchar(30), role:enum('customer','employee','other'), notes:varchar(90)),
   - `orders` (id:int, orderdate:date, duedate:date, shipped:bool, overdue:bool, ship:varchar(40), address:varchar(70), city:varchar(30), country:varchar(30), product: varchar(70), quantity: int, value: decimal(10,2)),
   - `stuff` (productname:varchar(70), category:varchar(40), unitprice: decimal(10,2), instock: int),

   Additionally, include the below specifics and set the proper constraints:

   - IDs are autoincremented artificial keys.
   - All IDs start from 1, no matter what ID was indicated in the `northwind` database.

[1] @'localhost' or @'%'

- Notes for people should include `northwind` notes for employees and information about the company for customers.
- Shipped field indicates if the order was already shipped.
- Overdue indicates if the order was shipped after the required date.
- Orders' values are the total price paid for given products (the price may include some discounts).
- The quantity, number of units InStock and unit prices cannot be negative.
- `Product` in `orders` refers to `productname` in `stuff` as a foreign key.
- The `firstname`, `lastname` and `category` cannot be NULLs.

3. Import the data from `northwind` database to `warehouse`. Do not import data concerning the orders that were not shipped, the orders containing (and the products themselves) from the *Confections* category nor data about the customers that are based in *Germany* or don't have a country of origin set.

4. Add a column to `orders` that will keep the information on the employee responsible for the order. Set a Foreign Key constraint. Think about the way to cope with people of different roles.

5. Add at least 10 new entries to each of the tables.

6. Create a view that stores the information on all the countries and cities present in the database.

7. Create a trigger that controls that new orders are only possible if all the stuff in them is currently in stock.

8. Create a trigger (or modify one of existing ones) that in the case of deleting an order that was not shipped yet, restocks the stuff that was not shipped.

9. Add an `age` column to `people`. Create a trigger that controls that none of the value is below 18 (put NULL in such cases). In cases that allow it, import the correct data from `northwind`, for other people insert random value between 0 and 100. Verify the correctness of the trigger.

10. Create a trigger (or modify one of existing ones) that after creating a new order, subtracts the items in the order from the stock.

11. Create a procedure that given two input parameters – the first and last name of a person, returns an output parameter that stores their complete address. Make sure that the user's input is verified.

12. Create a procedure that refills the stock of the product passed as the input parameter. Please try to adapt the procedure so that passing multiple products is possible. Make sure that the user's input is verified.

13. Create a function that given person's country and role, returns the number of people of specific role based in given country. Make sure that the user's input is verified.

14. Create a function that for given product, returns the date of the order where that product was sold for the highest total price. Make sure that the user's input is verified.

15. Create a prepared statement that for a given category calculates the number of different items in it and their average price. The category name should be given during `EXECUTE`. Check if the input is safe.

16. Create a prepared statement that allows calculating the value of function *aggr* on column *c* in `stuff` table. The `MIN` and `MAX` *aggr* functions should be possible for all columns, whereas `AVG` for all the numeral columns. The evaluation of the passed parameters should be delayed. Check if the input is safe.

17. Create an index allowing us to speed up the searches by `lastname`.

18. Create a procedure that given a category name and a budget generates random orders of products in the given category. Use transactions so that the budget is kept (whenever you add a product that would surpass the limit, rollback the transaction). Add the properly created orders to the database.

19. Download WebGoat app[2] and run it locally. Pass through *Injection Flaws*:

    - *SQL Injection (introduction)* – optional
    - *SQL Injection (advanced)*
    - *SQL Injection (mitigation)*

    Select one of the topics (advanced or mitigation) and create a short report describing the performed tasks and the conclusions drawn from them. Send the report (in pdf format) to the lecturer at least 24h before the deadline.

[2]Note that the later versions may require a proper Java version, e.g. 11