Introduction to Databases

Use of AI

The use of artificial intelligence tools **is not allowed in this activity**. The course plan and the UOC's academic integrity and plagiarism have information on what is considered misconduct in assessment activities and the consequences this may have.

PR2: Stored procedures and triggers: why are they needed?

In the first part of the practice, we made queries and modifications on the data using SQL. In this second part, using the same database, we will add logic within the database using stored procedures and triggers.

To correctly execute the second part of the practice, **it is mandatory to create a new database from scratch and insert all initial data using the scripts we provide in this statement** (*create_db.sql* and *inserts_db.sql*, respectively). This step is required as we are introducing the following changes in the database schema compared to the one provided in PR1:

- Creation of a new table called *REPORT_WINE*, with the next columns:
    - *wine_id*: Wine identifier.
    - *wine_name*: Wine name.
    - *alcohol_content*: Wine alcohol content.
    - *category*: Wine category.
    - *price*: Wine price.
    - *prizes*: Number of awards the wine has received.
    - *total_sold*: Total quantity of boxes sold.
    - *orders*: Number of orders in which the wine has been requested.
    - *customer_id*: Identifier of the customer who has placed the most orders for the wine.
    - *customer_name*: Name of the customer who has placed the most orders for the wine.

Important note: SQL language implemented in PostgreSQL may accept different syntax statements that could vary depending on which version you have installed, and these may not be SQL standard. Please avoid (unless otherwise specified) utilizing those types of statements and focus on the ones that are explained in the topics. If you use SQL standard statements, your code will work in any RDBMS.

Exercise 1 (15% weight)

You need to create a stored procedure that, given a wine identifier, provides specific data about it. Specifically, we want its identifier (*wine_id*), the name of the wine (*wine_name*), its alcohol content (*alcohol_content*), its category (*category*), the price (*price*), the number of awards it has received (*prizes*), the total quantity of solded boxes (*total_sold*), the number of orders in which it has been requested (*orders*), and the customer who has placed the most orders for the wine (*customer_id* and *customer_name*). In case of a tie, the customer with the highest number of boxes purchased should be selected, and if there is still a tie, the customer whose name appears first alphabetically should be chosen.

All of this information should be stored in the *REPORT_WINE* table. This table should have been created by executing the *create_db.sql* file, **which you must run before anything else**. If there are already rows in the *REPORT_WINE* table for the wine, the table should be updated with the new values. In addition to storing the data in the *REPORT_WINE* table, the procedure needs to return and display the result of the report.

The user should be informed with a specific message when no wine exists with the given identifier. It should also indicate if the wine has never been requested in an order.

The signature of the required stored procedure and the type it needs to return are as follows:

```
CREATE OR REPLACE FUNCTION update_report_wine(p_wine_id INT)

RETURNS REPORT_WINE_TYPE AS $$
```

where *REPORT_WINE_TYPE* type is:

```
CREATE TYPE REPORT_WINE_TYPE AS (
      t_wine_id INTEGER,
      t_wine_name VARCHAR(100),
      t_alcohol_content DECIMAL(4,2),
      t_category VARCHAR(50),
      t_price DECIMAL(8,2),
      t_prizes INTEGER,
      t_total_sold INTEGER,
      t_orders INTEGER,
      t_customer_id INTEGER,
      t_customer_name VARCHAR(100));
```

Note: Additionally, at the following link, you will find information about the error and message order in PL/PostgreSQL: [PostgreSQL: Documentation: 16: 43.9. Errors and Messages].

Assessment Criteria

- *Proposed solutions that are not executing (for example, due to a syntax error) will not be evaluated.*
- *We will positively value the use of SQL standard (apart from other elements indicated in the exercise).*
- *To obtain the maximum mark:*
    - *The SQL code of your proposal shall be efficient. For example, we'll penalize the use of unnecessary joins.*
    - *The proposed solution shall include tests that cover all possible situations defined in this exercise. For example, they shall cover all potential error scenarios.*
    - *The proposed solution shall include the results of the stored procedure (via screenshot or similar).*

Exercise 2 (18 % weight)

In the table *WINE* we have a column called *stock* that will be used to store **the number of boxes available for each wine**.

You need to create a trigger (or triggers), on any necessary table or tables, so it correctly maintains updated the column *stock* in the table *WINE*, so that inventory remains accurate in real-time based on the customer orders. The user should be informed with a specific message when stock is insufficient for an order.

Concretely, we are asking for this column to always store up to date the values every time there are changes to the database.

Concretely, we are asking for this column to always store up to date values every time there are changes to the database.

You can assume that NO users or programs will directly update the column *stock* in the table *WINE*.

Assessment criteria

- *Proposed solutions that are not executing (for example, due to a syntax error) will not be evaluated.*
- *We will positively value the use of SQL standard (apart from other elements indicated in the exercise).*
- *To obtain the maximum mark:*
  - *The SQL code of your proposal shall be efficient. For example, we'll penalize the use of unnecessary joins.*
  - *The proposed solution shall include tests that cover all possible situations defined in this exercise.*
  - *The proposed solution shall include the results (via screenshot or similar).*