Project Laboratory in Database Programming
Task List No. 3

Task 34. Write a PL/SQL block that selects cats with a function provided via keyboard input from the Kocury table. The only effect of the block should be a message informing whether a cat with the given function was found or not. If a cat is found, the corresponding function name should be displayed.

Task 35. Write a PL/SQL block that outputs the following information about a cat with a nickname entered from the keyboard (depending on the actual data):

- 'total annual mouse allocation > 700'
- 'name contains the letter A'
- 'May is the month of joining the pack'
- 'does not meet the criteria

The above information should be displayed according to its hierarchy of importance, meaning the first true information about the cat should be displayed, skipping the checks for the following conditions.

Task 36. Due to high efficiency in catching mice, SZEFUNIO decided to reward his subordinates. He announced that he would increase each cat's individual mouse allocation by 10%, starting with the cat with the lowest allocation and ending with the cat with the highest allocation. He also decided that the process of increasing the allocations should end once the total sum of all the cats' allocations exceeds 1050. If, for any cat, the mouse allocation after the increase exceeds the maximum value allowed for their function (max_myszy from the Functions relation), the mouse allocation after the increase should be set to that value.

Write a PL/SQL block using a cursor that performs this task. The block should run until the total of all allocations truly exceeds 1050 (the number of “salary increases” may be greater than 1, so the increase may be greater than 10%). Display on the screen the total mouse allocations after completing the task, along with the number of modifications in the Cats relation. Finally, rollback all changes.

```
Calk. przydzial w stadku 1057  Zmian - 30

IMIE            Myszki po podwyzce
--------------- ------------------
MRUCZEK                        110
CHYTRY                          55
KOREK                           90
BOLEK                           87
ZUZIA                           70
RUDA                            26
PUCEK                           70
PUNIA                           70
BELA                            29
KSAWERY                         60
MELA                            60
JACEK                           70
BARI                            60
MICKA                           30
LUCEK                           50
SONIA                           24
LATKA                           48
DUDEK                           48
```

Task 37. Write a block that selects the top five cats with the highest total mouse allocation using a FOR cursor loop. The result should be displayed on the screen.

```
Nr  Psedonim   Zjada
--------------------
1   TYGRYS      136
2   LYSY         93
3   ZOMBI        88
4   LOLA         72
5   PLACEK       67
```

Task 38. Write a block that will implement version a. or version b. of task 19 in a universal way (without the need to account for the depth of the tree). The input data should be the maximum number of supervisors to be displayed.

```
Result for amount of supervisors = 5

Imie           |  Szef 1         |  Szef 2         |  Szef 3
------------- --- ------------- --- ------------- --- -------------
RUDA           |  MRUCZEK        |                 |
BELA           |  BOLEK          |  MRUCZEK        |
MICKA          |  MRUCZEK        |                 |
LUCEK          |  PUNIA          |  KOREK          |  MRUCZEK
SONIA          |  KOREK          |  MRUCZEK        |
LATKA          |  PUCEK          |  MRUCZEK        |
DUDEK          |  PUCEK          |  MRUCZEK        |

Result for amount of supervisors = 2

Imie           |  Szef 1         |  Szef 2
------------- --- ------------- --- -------------
RUDA           |  MRUCZEK        |
BELA           |  BOLEK          |  MRUCZEK
MICKA          |  MRUCZEK        |
LUCEK          |  PUNIA          |  KOREK
SONIA          |  KOREK          |  MRUCZEK
LATKA          |  PUCEK          |  MRUCZEK
DUDEK          |  PUCEK          |  MRUCZEK
```

Task 39. Write a PL/SQL block that reads three parameters representing the band number, band name, and hunting territory. The script should prevent the input of already existing values by handling appropriate exceptions. An exceptional situation occurs when the band number is <= 0. In the case of an exception, the block should display an appropriate message on the screen. If the parameters are valid, a new band should be created in the "Bandy" table. The change should be rolled back at the end.

```
Example results (respectively, 3, 2 i 1 argument values already exist in
the database):

2, CZARNI RYCERZE, POLE: juz istnieje (already exists)

1, SZEFOSTWO: juz istnieje (already exists)

SAD: juz istnieje (already exists)
```

Task 40. Remake PL/SQL block from task 39 to a saved database procedure.

Task 41. Define a trigger that ensures that the number of a new band will always be one greater than the highest number of an existing band. Test the trigger by using the procedure from task 40.

Task 42. The MILUSIA functions cats decided to take care of their interests. They hired an IT specialist to plant a virus in the Tygrys system. Now, every time an attempt is made to increase the mouse allocation (a decrease is not allowed at all – such an attempt should be blocked by the proposed mechanism, with exceptions), if the increase is less than 10% of Tygrys's mouse allocation, the Milusies should be "consoled" with a 10% increase in their allocation and an extra mouse increase of 5. Tygrys should be penalized by losing the mentioned 10%. However, if the raise is satisfactory for the Milusies (an increase of 10% or more), Tygrys's extra mouse allocation should be increased by 5.

Propose two solutions to this task that will bypass the basic limitation of the row-level trigger activated by the DML command, i.e., the inability to read or modify the relation on which the operation (DML command) "triggers" this trigger. In the first solution (classic), use multiple triggers and memory in the form of a package specification dedicated to the task, and in the second, use a COMPOUND trigger.

Provide an example of the trigger functionality and then remove the changes introduced by them.

Task 43. Write a block that will implement task 33 in a universal way (without the need to account for the functions held by the cats).

Task 44. Tygrys was concerned about the unexplained depletion of "mouse" stocks. He decided to introduce a head tax that would replenish the pantry. He ordered that each cat must give away 5% (rounded up) of their total "mouse" income. Additionally, from the remaining amount:

- Cats with no subordinates must give away two mice for their incompetence in courting a promotion
- Cats with no enemies must give away one mouse for being too conciliatory.
- Male cats pay an additional tax, the form of which is defined by the task solver.

Write a function with a parameter for the cat's pseudonym, which calculates the required head tax for the cat. This function, together with the procedure from task 40, should be placed in a package, and then used to determine the tax for all cats.

Task 45. Tygrys noticed strange changes in the value of his personal mouse allocation (see task 42). While he was not concerned about increases, decreases were, in his opinion, unacceptable. He motivated one of his spies to act, and thanks to this, he discovered the nefarious practices of MILUSIA cats (task 42). He then ordered his IT technician to create a mechanism that would log in the Dodatki_extra table (see Lectures - part 2) a -10 (minus ten) mouse extra benefit for each Milus when their mouse allocation is increased by an operator other than him (Tygrys). Propose such a mechanism, in place of Tygrys's IT technician. The solution should use the LOGIN_USER function, which returns the username of the operator activating the trigger, and elements of dynamic SQL.

Task 46. Write a trigger that prevents assigning a mouse allocation to a cat outside the range (min_myszy, max_myszy) defined for each role in the Funkcje relation.
Every attempt to exceed the allowed range must also be monitored in a separate relation (who, when, which cat, what operation).

Termin oddania listy – grupa poniedziałkowa: 18.12.2023
– grupy wtorkowe: 19.12.2023

Wrocław 28.09.2023 *Zbigniew Staszak*