Alex Rogoziński 280522

Programowanie Baz Danych

Laboratorium

Lista 3 - Raport

Spis treści

Zadania - bloki

Zadanie 1

PL/SQL

```
-------------------------------------------------------------------------------------------
-- TASK 1
-------------------------------------------------------------------------------------------
DECLARE
    v_funkcja_kocura Kocury.funkcja%TYPE;
BEGIN
    SELECT
        funkcja
    INTO
        v_funkcja_kocura
    FROM Kocury
    WHERE
        funkcja = UPPER('&funkcja_input')
    FETCH FIRST 1 ROWS ONLY;

    DBMS_OUTPUT.PUT_LINE('Znaleziono: ' || v_funkcja_kocura);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nie znaleziono.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

```
Znaleziono: LOWCZY


PL/SQL procedure successfully completed.
```

Dla `funkcja_input = 'loWczy'`

```
Nie znaleziono.


PL/SQL procedure successfully completed.
```

Dla `funkcja_input = 'honorowa'`

Zadanie 2

T-SQL

```
-------------------------------------------------------------------------------------------
-- TASK 2
-------------------------------------------------------------------------------------------
DECLARE @pseudo_input VARCHAR(15) = 'tyGrYs'

DECLARE
    @pseudo                 VARCHAR(15),
    @imie                   VARCHAR(15),
    @nazwa_bandy            VARCHAR(20),
    @czy_ma_wrogow          VARCHAR(3),
    @czy_wiekszy_od_sredniej VARCHAR(3),
    @w_stadku_od            DATE,
    @przydzial_myszy        INT,
    @srednia_bandy          FLOAT;

IF NOT EXISTS (
    SELECT
        pseudo
    FROM Kocury
    WHERE
        pseudo = UPPER(@pseudo_input)
)
BEGIN
    PRINT 'ERROR: Kot o podanym pseudonimie nie istnieje: ' + UPPER(@pseudo_input);
    RETURN;
END;

SELECT
    @pseudo          = k.pseudo,
    @imie            = k.imie,
    @nazwa_bandy     = b.nazwa,
    @w_stadku_od     = k.w_stadku_od,
    @przydzial_myszy = k.przydzial_myszy
FROM Kocury k
    INNER JOIN Bandy b ON b.nr_bandy = k.nr_bandy
WHERE
    k.pseudo = UPPER(@pseudo_input);

IF EXISTS (
    SELECT
        pseudo
    FROM Wrogowie_kocurow
    WHERE
        pseudo = UPPER(@pseudo_input)
)
    SET @czy_ma_wrogow = 'TAK';
ELSE
    SET @czy_ma_wrogow = 'NIE';

SELECT
    @srednia_bandy = AVG(12.0 * przydzial_myszy)
FROM Kocury
WHERE
```

```
    nr_bandy = (
        SELECT
            nr_bandy
        FROM Kocury
        WHERE
            pseudo = UPPER(@pseudo_input)
    );

IF 12 * @przydzial_myszy > @srednia_bandy
    SET @czy_wiekszy_od_sredniej = 'TAK';
ELSE
    SET @czy_wiekszy_od_sredniej = 'NIE';

PRINT 'Pseudo:'                                          + SPACE(46) + @pseudo;
PRINT 'Imie:'                                            + SPACE(48) + @imie;
PRINT 'Nazwa bandy:'                                     + SPACE(41) + @nazwa_bandy;
PRINT 'Czy ma wrogow:'                                   + SPACE(39) + @czy_ma_wrogow;
PRINT 'Czy roczny przydzial jest wiekszy od sredniej bandy:' + SPACE(1)  +
@czy_wiekszy_od_sredniej;
PRINT 'Dzien przystapienia do stada:'                    + SPACE(24) +
CAST(DAY(@w_stadku_od) AS VARCHAR);
PRINT 'Miesiac przystapienia do stada:'                  + SPACE(22) + DATENAME(month,
@w_stadku_od);
PRINT 'Rok przystapienia do stada:'                      + SPACE(26) +
CAST(YEAR(@w_stadku_od) AS VARCHAR);
```

```
Started executing query at  Line 1
Pseudo:                                          TYGRYS
Imie:                                            MRUCZEK
Nazwa bandy:                                     SZEFOSTWO
Czy ma wrogow:                                   TAK
Czy roczny przydzial jest wiekszy od sredniej bandy: TAK
Dzien przystapienia do stada:                    1
Miesiac przystapienia do stada:                  January
Rok przystapienia do stada:                      2002
Total execution time: 00:00:00.028
```

Dla `pseudo_input = 'tyGrYs'`

```
Started executing query at  Line 1
ERROR: Kot o podanym pseudonimie nie istnieje: MILEK
Total execution time: 00:00:00.038
```

Dla `pseudo_input = 'milek'`

Zadanie 3

PL/SQL

```
-------------------------------------------------------------------------------
-- TASK 3
-------------------------------------------------------------------------------
DECLARE
    v_przydzial_myszy Kocury.przydzial_myszy%TYPE;
    v_myszy_extra     Kocury.myszy_extra%TYPE;
    v_imie_kota       Kocury.imie%TYPE;
    v_w_stadku_od     Kocury.w_stadku_od%TYPE;
BEGIN
    SELECT
        przydzial_myszy,
        NVL(myszy_extra, 0),
        imie,
        w_stadku_od
    INTO
        v_przydzial_myszy,
        v_myszy_extra,
        v_imie_kota,
        v_w_stadku_od
    FROM Kocury
    WHERE
        pseudo = UPPER('&pseudo_input');

    IF 12 * (v_przydzial_myszy + v_myszy_extra) > 700 THEN
        DBMS_OUTPUT.PUT_LINE('calkowity roczny przydzial myszy >700');

    ELSIF v_imie_kota LIKE '%A%' THEN
        DBMS_OUTPUT.PUT_LINE('imię zawiera litere A');

    ELSIF EXTRACT(MONTH FROM v_w_stadku_od) = 5 THEN
        DBMS_OUTPUT.PUT_LINE('maj jest miesiacem przystapienia do stada');

    ELSE
        DBMS_OUTPUT.PUT_LINE('nie odpowiada kryteriom');
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nie znaleziono.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

```
calkowity roczny przydzial myszy >700


PL/SQL procedure successfully completed.
```

Dla `pseudo_input = 'tygrys'`

```
imię zawiera litere A


PL/SQL procedure successfully completed.
```

Dla `pseudo_input = 'dama'`

```
maj jest miesiacem przystapienia do stada


PL/SQL procedure successfully completed.
```

Dla `pseudo_input = 'Bolek'`

```
Nie znaleziono.


PL/SQL procedure successfully completed.
```

Dla `pseudo_input = 'szAfir'`

Zadanie 4

PL/SQL

```
--------------------------------------------------------------------------------
-- TASK 4
--------------------------------------------------------------------------------
DECLARE
    TYPE r_kot_dane IS RECORD (pseudo Kocury.pseudo%TYPE, nr_bandy Kocury.nr_bandy%TYPE, staz
NUMBER);
    TYPE t_tabela_kocury IS TABLE OF r_kot_dane INDEX BY PLS_INTEGER;
    t_kocury t_tabela_kocury;
    e_brak_rekordow EXCEPTION;
BEGIN
    -- Zbierz dane
    WITH Minimalne_staze AS (
        SELECT
            nr_bandy,
            SYSDATE - MAX(w_stadku_od) AS staz
        FROM Kocury
        GROUP BY
            nr_bandy
    )
    -- Zaladuj dane do tabeli
    SELECT
        k.pseudo,
        k.nr_bandy,
        ROUND(ms.staz, 0)
    BULK COLLECT INTO
        t_kocury
    FROM Kocury k
        INNER JOIN Minimalne_staze ms ON ms.nr_bandy = k.nr_bandy
    GROUP BY
        k.pseudo,
        k.nr_bandy,
        ms.staz
    HAVING
        SYSDATE - MAX(k.w_stadku_od) = ms.staz;

    -- Wypisz na ekran
    FOR i IN t_kocury.FIRST..t_kocury.LAST LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD('Pseudo:',   10) || t_kocury(i).pseudo);
        DBMS_OUTPUT.PUT_LINE(RPAD('Nr bandy:', 10) || t_kocury(i).nr_bandy);
        DBMS_OUTPUT.PUT_LINE(RPAD('Staz:',     10) || t_kocury(i).staz || ' dni');
        DBMS_OUTPUT.NEW_LINE;
    END LOOP;
EXCEPTION
    WHEN e_brak_rekordow THEN
        DBMS_OUTPUT.PUT_LINE('ERROR: Brak rekordow do wyswietlenia.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

```
Pseudo:   RURA
Nr bandy: 2
Staz:     5952 dni

Pseudo:   LOLA
Nr bandy: 1
Staz:     5909 dni

Pseudo:   PUSZYSTA
Nr bandy: 3
Staz:     5509 dni

Pseudo:   MALY
Nr bandy: 4
Staz:     5331 dni



PL/SQL procedure successfully completed.
```

Zadanie 5

PL/SQL

```
-------------------------------------------------------------------------------
-- TASK 5
-------------------------------------------------------------------------------
DECLARE
    CURSOR c_kursor IS
        SELECT
            k.imie,
            k.pseudo,
            k.przydzial_myszy,
            f.max_myszy
        FROM Kocury k
            INNER JOIN Funkcje f ON f.funkcja = k.funkcja
        ORDER BY
            przydzial_myszy DESC;
    r_kot_dane        c_kursor%ROWTYPE;
    v_zmiany          NUMBER := 0;
    v_suma_przydz     NUMBER := 0;
    v_nowy_przydzial  NUMBER;
BEGIN
    LOOP
        -- Policz aktualna sume przydzialow przez zalawdowanie do zmiennej
        -- wynikowej wartosci zwracanej przez prosta kwerende.
        SELECT
            SUM(przydzial_myszy)
        INTO
            v_suma_przydz
        FROM Kocury;
        EXIT WHEN v_suma_przydz > 1050;

        OPEN c_kursor;

        -- Zaktualizuj przydzialy jesli ich suma nie przekracza 1050; najpierw
        -- oblicz 110% poprzedniego a nastepnie sprawdz czy nie przekracza
        -- `max_myszy` dla funkcji kocura i odpowiednio zmien lub zostaw
        -- obliczona przed chwila wartosc.
        -- Zaktualizuj licznik zmian weryfikujac czy zmiana faktycznie nastapila.
        LOOP
            FETCH c_kursor INTO r_kot_dane;
            EXIT WHEN c_kursor%NOTFOUND;

            v_nowy_przydzial := 1.1 * r_kot_dane.przydzial_myszy;
            IF v_nowy_przydzial > r_kot_dane.max_myszy THEN
                v_nowy_przydzial := r_kot_dane.max_myszy;
            END IF;

            IF v_nowy_przydzial != r_kot_dane.przydzial_myszy THEN
                v_zmiany := v_zmiany + 1;
            END IF;

            UPDATE Kocury SET przydzial_myszy = v_nowy_przydzial WHERE pseudo =
r_kot_dane.pseudo;
        END LOOP;
```

```
        CLOSE c_kursor;
    END LOOP;

    -- Wypisz sformatowane dane
    DBMS_OUTPUT.PUT_LINE('Calk. przydzial w stadku ' || v_suma_przydz || '  Zmian - ' ||
v_zmiany);
    DBMS_OUTPUT.NEW_LINE();
    DBMS_OUTPUT.PUT_LINE(RPAD('IMIE', 16) || 'Myszki po podwyzce');
    DBMS_OUTPUT.PUT_LINE(RPAD('-', 15, '-') || ' ' || RPAD('-', 18, '-'));

    FOR kot IN (
        SELECT
            imie,
            przydzial_myszy
        FROM Kocury
        ORDER BY
            w_stadku_od
    ) LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(kot.imie, 16) || LPAD(kot.przydzial_myszy, 18));
    END LOOP;

    ROLLBACK;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        ROLLBACK;
END;
/
```

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

IMIE            Myszki po podwyzce
--------------- ------------------
MRUCZEK                        110
CHYTRY                          55
KOREK                           90
ZUZIA                           70
BOLEK                           87
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


PL/SQL procedure successfully completed.
```

W celu realizacji tego działania należało zastosować pętle zagnieżdżone. W zewnętrznej pętli należy obliczyć aktualną sumę przydziałów myszy i sprawdzić czy nie przekracza ona wartości 1050. W wewnętrznej pętli miejsce ma główna logika obliczania i ustalania nowego przydziału oraz inkrementowany jest licznik dokonanych zmian w przydziałach.

Zadanie 6

PL/SQL

```
-------------------------------------------------------------------------------
-- TASK 6
-------------------------------------------------------------------------------
DECLARE
    v_numer NUMBER := 1;
BEGIN
    -- Naglowek
    DBMS_OUTPUT.PUT_LINE(RPAD('Nr', 4) || RPAD('Pseudonim', 11) || 'Zjada');
    DBMS_OUTPUT.PUT_LINE(RPAD('-', 20, '-'));

    -- Zawartosc w petli z kursorem
    FOR rekord IN (
        SELECT
            pseudo,
            przydzial_myszy + NVL(myszy_extra, 0) AS zjada
        FROM Kocury
        ORDER BY
            zjada DESC
        FETCH FIRST 5 ROWS ONLY
    ) LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(v_numer, 4) || RPAD(rekord.pseudo, 11) ||
RPAD(LPAD(rekord.zjada, 4), 5));
        v_numer := v_numer + 1;
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

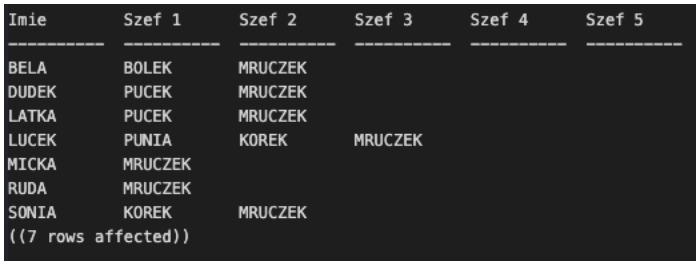
```
Nr  Pseudonim  Zjada
--------------------
1   TYGRYS      136
2   LYSY         93
3   ZOMBI        88
4   LOLA         72
5   PLACEK       67


PL/SQL procedure successfully completed.
```

Zadanie 7

A.

T-SQL

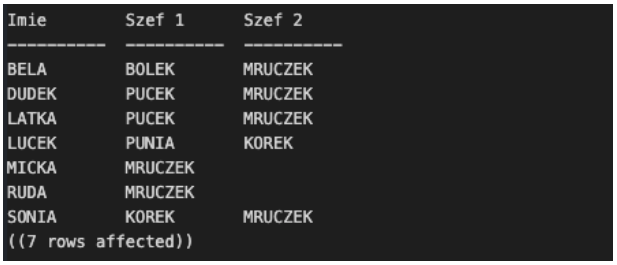
```
-------------------------------------------------------------------------------------------
-- TASK 7
-------------------------------------------------------------------------------------------
-- a)
DECLARE
    @przelozeni_num INT = 4,
    @dynamic_sql    NVARCHAR(MAX),
    @index          INT = 1,
    @index_nvarchar NVARCHAR(3);

-- Poczatek kwerendy
SET @dynamic_sql = 'SELECT k0.imie AS Imie';

-- Wyznaczenie danych do wypisania
WHILE @index <= @przelozeni_num
BEGIN
    SET @index_nvarchar = CAST(@index AS NVARCHAR);
    SET @dynamic_sql += N', ISNULL(k' + @index_nvarchar + N'.imie, '''') AS [Szef ' +
@index_nvarchar + N']';
    SET @index += 1;
END;

SET @dynamic_sql += N' FROM Kocury k0';

SET @index = 1;

-- Wyznaczenie zlaczen
WHILE @index <= @przelozeni_num
BEGIN
    SET @index_nvarchar = CAST(@index AS NVARCHAR);
    SET @dynamic_sql += N' LEFT JOIN Kocury k' + @index_nvarchar + N' ON k' + @index_nvarchar
                     + N'.pseudo = k' + CAST(@index - 1 AS NVARCHAR) + N'.szef';

    SET @index += 1;
END;

-- Dodanie warunkow poczatkowych i porzadku
SET @dynamic_sql += N'
    WHERE
        k0.funkcja IN (''KOT'', ''MILUSIA'')
    ORDER BY
        k0.imie;
    ';

-- Wywolanie
EXEC sp_executesql @dynamic_sql;
```



Imie	Szef 1	Szef 2	Szef 3	Szef 4	Szef 5
BELA	BOLEK	MRUCZEK			
DUDEK	PUCEK	MRUCZEK			
LATKA	PUCEK	MRUCZEK			
LUCEK	PUNIA	KOREK	MRUCZEK		
MICKA	MRUCZEK				
RUDA	MRUCZEK				
SONIA	KOREK	MRUCZEK			

((7 rows affected))

Dla liczby przełożonych = 5



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

((7 rows affected))

Dla liczby przełożonych = 2

Zadanie 8

PL/SQL

```
-------------------------------------------------------------------------------
-- TASK 8
-------------------------------------------------------------------------------
DECLARE
    v_nr_bandy Bandy.nr_bandy%TYPE := &nr_bandy_input;
    v_nazwa    Bandy.nazwa%TYPE    := UPPER('&nazwa_input');
    v_teren    Bandy.teren%TYPE    := UPPER('&teren_input');
    v_errmsg   VARCHAR(55)         := '';
    v_errcnt   NUMBER              := 0;

    e_invalid_nr_bandy EXCEPTION;
    e_invalid_input    EXCEPTION;
BEGIN
    IF v_nr_bandy <= 0 THEN
        RAISE e_invalid_nr_bandy;
    END IF;

    FOR banda IN (SELECT * FROM Bandy) LOOP
        IF v_nr_bandy = banda.nr_bandy THEN
            v_errmsg := v_errmsg || TO_CHAR(v_nr_bandy);
            v_errcnt := v_errcnt + 1;
        END IF;

        IF v_nazwa = banda.nazwa THEN
            IF v_errcnt > 0 THEN
                v_errmsg := v_errmsg || ', ';
            END IF;
            v_errmsg := v_errmsg || v_nazwa;
            v_errcnt := v_errcnt + 1;
        END IF;

        IF v_teren = banda.teren THEN
            IF v_errcnt > 0 THEN
                v_errmsg := v_errmsg || ', ';
            END IF;
            v_errmsg := v_errmsg || v_teren;
            v_errcnt := v_errcnt + 1;
        END IF;

        IF v_errmsg IS NOT NULL THEN
            RAISE e_invalid_input;
        END IF;
    END LOOP;

    INSERT INTO Bandy VALUES (v_nr_bandy, v_nazwa, v_teren, NULL);
    DBMS_OUTPUT.PUT_LINE('Dodano bande: [' || v_nr_bandy || ', ' || v_nazwa || ', ' || v_teren
|| ']');
    ROLLBACK;
EXCEPTION
    WHEN e_invalid_nr_bandy THEN
        DBMS_OUTPUT.PUT_LINE('Numer bandy musi byc wiekszy od 0: ' || v_nr_bandy);
    WHEN e_invalid_input THEN
        DBMS_OUTPUT.PUT_LINE(v_errmsg || ': juz istnieje');
```

```
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        ROLLBACK;
END;
/
```

```
2, CZARNI RYCERZE, POLE: juz istnieje


PL/SQL procedure successfully completed.
```

Dla danych [2, czarni rycerze, pole]

```
1, SZEFOSTWO: juz istnieje


PL/SQL procedure successfully completed.
```

Dla danych [1, szefostwo, ogrod]

```
SAD: juz istnieje


PL/SQL procedure successfully completed.
```

Dla danych [17, muszkieterzy, sad]

```
Dodano bande: [36, KOMANDOSI, JASKINIA]


PL/SQL procedure successfully completed.
```

Dla danych [36, komandosi, jaskinia]

Ogólne spostrzeżenia po wykonaniu zadań z części *Zadania - bloki*

Pomimo wykonania tylko dwóch zadań w T-SQL i sześciu w PL/SQL, różnice między tymi dialektami były bardzo dobrze widoczne. Składnia PL/SQL jest bardziej rygorystyczna (m.in. pod względem wymagania `BEGIN-END` jako oznaczenia granic bloku), to osobiście podoba mi się ona bardziej niż bardziej swobodna składnia T-SQL, głównie z uwagi na fakt „prowadzenia za rękę" według zasad tego co i w jaki sposób jako programista mogę napisać wewnątrz bloku, choć możliwe, że jest to jedynie efekt braku doświadczenia w pisaniu tego typu kodu. W T-SQL brakuje możliwości interaktywnego podawania wartości dla zmiennych, co nieco utrudnia testowanie kodu.

Zadania - procedury

Zadanie 9

PL/SQL

```
-------------------------------------------------------------------------------------------
-- TASK 9
-------------------------------------------------------------------------------------------
-- Definicja
CREATE OR REPLACE PROCEDURE ZmienPrzydzialDlaFunkcji (
    p_funkcja         IN Kocury.funkcja%TYPE,
    p_przydzial_myszy IN Kocury.przydzial_myszy%TYPE
) IS
BEGIN
    -- Weryfikacja poprawnosci wartosci przydzialu
    IF p_przydzial_myszy IS NULL OR p_przydzial_myszy < 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Nieprawidlowy przydzial myszy: ' || p_przydzial_myszy);
    END IF;

    -- Aktualizacja kocurow o podanej funkcji
    UPDATE Kocury
    SET przydzial_myszy = p_przydzial_myszy
    WHERE
        funkcja = UPPER(p_funkcja);

    -- Sprawdzenie czy kocury o podanej funkcji istnialy
    IF SQL%ROWCOUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Brak kotow o funkcji: ' || p_funkcja);
    END IF;
EXCEPTION
    WHEN VALUE_ERROR THEN
        DBMS_OUTPUT.PUT_LINE('Nieprawidlowy typ danych.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        RAISE;
END ZmienPrzydzialDlaFunkcji;
/

-- Wywolanie
DECLARE
    CURSOR c_kursor IS
        SELECT
            pseudo,
            przydzial_myszy,
            funkcja
        FROM Kocury
        ORDER BY
            funkcja;

    v_funkcja         Kocury.funkcja%TYPE         := '&funkcja_input';
    v_przydzial_myszy Kocury.przydzial_myszy%TYPE := &przydzial_myszy_input;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Przed aktualizacja:');
    FOR kot IN c_kursor LOOP
```

```
        DBMS_OUTPUT.PUT_LINE(RPAD(kot.pseudo, 15) || ' ' || LPAD(kot.przydzial_myszy, 3) || ' '
|| RPAD(kot.funkcja, 10));
    END LOOP;

    ZmienPrzydzialDlaFunkcji(v_funkcja, v_przydzial_myszy);
    DBMS_OUTPUT.NEW_LINE();

    DBMS_OUTPUT.PUT_LINE('Po aktualizacji dla ' || UPPER(v_funkcja) || ':');
    FOR kot IN c_kursor LOOP
        DBMS_OUTPUT.PUT_LINE(RPAD(kot.pseudo, 15) || ' ' || LPAD(kot.przydzial_myszy, 3) || ' '
|| RPAD(kot.funkcja, 10));
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        ROLLBACK;
END;
/
```

```
Procedure ZMIENPRZYDZIALDLAFUNKCJI compiled

No errors.
```

```
Przed aktualizacja:
LYSY             72 BANDZIOR
ZOMBI            75 BANDZIOR
BOLEK            50 DZIELCZY
MALY             40 KOT
ZERO             43 KOT
UCHO             40 KOT
DAMA             51 LAPACZ
RURA             56 LAPACZ
MAN              51 LAPACZ
RAFA             65 LOWCZY
KURKA            61 LOWCZY
SZYBKA           65 LOWCZY
PLACEK           67 LOWCZY
MALA             22 MILUSIA
LASKA            24 MILUSIA
PUSZYSTA         20 MILUSIA
LOLA             25 MILUSIA
TYGRYS          103 SZEFUNIO
```

Przed wywołaniem dla danych [lowczy, 22]

```
Po aktualizacji dla LOWCZY:
LYSY             72 BANDZIOR
ZOMBI            75 BANDZIOR
BOLEK            50 DZIELCZY
MALY             40 KOT
ZERO             43 KOT
UCHO             40 KOT
DAMA             51 LAPACZ
RURA             56 LAPACZ
MAN              51 LAPACZ
RAFA             22 LOWCZY
KURKA            22 LOWCZY
SZYBKA           22 LOWCZY
PLACEK           22 LOWCZY
MALA             22 MILUSIA
LASKA            24 MILUSIA
PUSZYSTA         20 MILUSIA
LOLA             25 MILUSIA
TYGRYS          103 SZEFUNIO
```

Po wywołaniu

T-SQL

```
-------------------------------------------------------------------------------------------
-- TASK 9
-------------------------------------------------------------------------------------------
-- Definicja
CREATE OR ALTER PROCEDURE ZmienPrzydzialDlaFunkcji
    @p_funkcja         VARCHAR(10),
    @p_przydzial_myszy INT
AS
BEGIN
    -- Weryfikacja poprawnosci wartosci przydzialu
    IF @p_przydzial_myszy IS NULL OR @p_przydzial_myszy < 0
        THROW 50000, 'Nieprawidlowy przydzial myszy.', 1;

    -- Aktualizacja kocurow o podanej funkcji
    UPDATE Kocury
    SET przydzial_myszy = @p_przydzial_myszy
    WHERE
        funkcja = UPPER(@p_funkcja);

    IF @@ROWCOUNT = 0
        THROW 50001, 'Brak kotow o podanej funkcji.', 1;
END;
GO

-- Wywolanie
DECLARE
    c_kursor CURSOR FOR
        SELECT
            pseudo,
            przydzial_myszy,
            funkcja
        FROM Kocury
        ORDER BY
            funkcja;
DECLARE
    @v_funkcja         VARCHAR(10) = UPPER('lowczy'),
    @v_przydzial_myszy INT         = 22,
    @v_pseudo_temp     VARCHAR(15),
    @v_przydzial_temp  INT,
    @v_funkcja_temp    VARCHAR(10);
BEGIN TRY
    PRINT 'Przed aktualizacja:';

    OPEN c_kursor;
    FETCH NEXT FROM c_kursor
    INTO
        @v_pseudo_temp,
        @v_przydzial_temp,
        @v_funkcja_temp;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        PRINT CONCAT(@v_pseudo_temp, ' ', @v_przydzial_temp, ' ', @v_funkcja_temp);
        FETCH NEXT FROM c_kursor
        INTO
```

```
            @v_pseudo_temp,
            @v_przydzial_temp,
            @v_funkcja_temp;
    END;
    CLOSE c_kursor;

    BEGIN TRANSACTION;

    EXEC ZmienPrzydzialDlaFunkcji @v_funkcja, @v_przydzial_myszy;

    PRINT '';
    PRINT 'Po aktualizacja dla ' + UPPER(CAST(@v_funkcja AS VARCHAR) + ':');
    OPEN c_kursor;
    FETCH NEXT FROM c_kursor
    INTO
        @v_pseudo_temp,
        @v_przydzial_temp,
        @v_funkcja_temp;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        PRINT CONCAT(@v_pseudo_temp, ' ', @v_przydzial_temp, ' ', @v_funkcja_temp);
        FETCH NEXT FROM c_kursor
        INTO
            @v_pseudo_temp,
            @v_przydzial_temp,
            @v_funkcja_temp;
    END;
    CLOSE c_kursor;

    DEALLOCATE c_kursor;
    ROLLBACK;
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK;

    PRINT ERROR_MESSAGE();
END CATCH;
GO
```

```
Started executing query at  Line 33
Przed aktualizacja:
LYSY 72 BANDZIOR
ZOMBI 75 BANDZIOR
BOLEK 50 DZIELCZY
MALY 40 KOT
UCHO 40 KOT
ZERO 43 KOT
RURA 56 LAPACZ
MAN 51 LAPACZ
DAMA 51 LAPACZ
KURKA 61 LOWCZY
PLACEK 67 LOWCZY
SZYBKA 65 LOWCZY
RAFA 65 LOWCZY
MALA 22 MILUSIA
PUSZYSTA 20 MILUSIA
LASKA 24 MILUSIA
LOLA 25 MILUSIA
TYGRYS 103 SZEFUNIO
(4 rows affected)
```

Przed wywołaniem dla danych [lowczy, 22]

```
Po aktualizacja dla LOWCZY:
LYSY 72 BANDZIOR
ZOMBI 75 BANDZIOR
BOLEK 50 DZIELCZY
MALY 40 KOT
UCHO 40 KOT
ZERO 43 KOT
RURA 56 LAPACZ
MAN 51 LAPACZ
DAMA 51 LAPACZ
KURKA 22 LOWCZY
PLACEK 22 LOWCZY
SZYBKA 22 LOWCZY
RAFA 22 LOWCZY
MALA 22 MILUSIA
PUSZYSTA 20 MILUSIA
LASKA 24 MILUSIA
LOLA 25 MILUSIA
TYGRYS 103 SZEFUNIO
```

Po wywołaniu

Zadanie 10

PL/SQL

```
-------------------------------------------------------------------------------
-- TASK 10
-------------------------------------------------------------------------------
UNDEFINE pseudo_input;

-- Definicja
CREATE OR REPLACE FUNCTION PodatekPoglowny (p_pseudo IN Kocury.pseudo%TYPE) RETURN NUMBER
IS
    TYPE r_dane IS RECORD (podwladni_cnt NUMBER, wrogowie_cnt NUMBER, plec Kocury.plec%TYPE,
pseudo Kocury.pseudo%TYPE);
    v_dane    r_dane;
    v_podatek NUMBER := 0;
BEGIN
    v_dane.pseudo := UPPER(p_pseudo);

    -- Podatek podstawowy - obliczenie 5% calkowitego przydzialu myszy
    -- dla kota o danym pseudonimie
    SELECT
        ROUND(0.05 * (przydzial_myszy+ NVL(myszy_extra, 0)), 0),
        plec
    INTO
        v_podatek,
        v_dane.plec
    FROM Kocury
    WHERE
        pseudo = v_dane.pseudo;

    -- Podatek od braku podwladnych - policzenie dla ilu kotow kot o danym
    -- pseudonimie jest szefem
    SELECT
        COUNT(*)
    INTO
        v_dane.podwladni_cnt
    FROM Kocury
    WHERE
        szef = v_dane.pseudo;

    IF v_dane.podwladni_cnt = 0 THEN
        v_podatek := v_podatek + 2;
    END IF;

    -- Podatek od braku wrogow - policzenie ile jest wystapien danego
    -- pseudonimu w tabeli `Wrogowie_kocurow`
    SELECT
        COUNT(*)
    INTO
        v_dane.wrogowie_cnt
    FROM Wrogowie_kocurow
    WHERE
        pseudo = v_dane.pseudo;

    IF v_dane.wrogowie_cnt = 0 THEN
        v_podatek := v_podatek + 1;
```

```
    END IF;

    -- Podatek od kocurow (dodany) - podatek w postaci 5 myszy dla kocurow
    -- (plci meskiej)
    IF v_dane.plec = 'M' THEN
        v_podatek := v_podatek + 5;
    END IF;

    RETURN v_podatek;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nie znaleziono kota o podanym pseudonimie: ' || v_dane.pseudo);
        RAISE;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        RAISE;
END PodatekPoglowny;
/

-- Wywolanie
DECLARE
    v_pseudo Kocury.pseudo%TYPE := '&pseudo_input';
BEGIN
    DBMS_OUTPUT.PUT_LINE('Podatek dla ' || UPPER(v_pseudo) || ': ' || PodatekPoglowny(v_pseudo)
|| ' myszy');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nie znaleziono kota o podanym pseudonimie.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

```
Function PODATEKPOGLOWNY compiled

No errors.
```

```
Podatek dla TYGRYS: 12 myszy


PL/SQL procedure successfully completed.
```

Dla `p_pseudo = 'tygrys'` (podwł. 6 i wrog. 2)

```
Podatek dla ZERO: 10 myszy


PL/SQL procedure successfully completed.
```

Dla `p_pseudo = 'zero'` (podwł. 0 i wrog. 0)

T-SQL

```
-------------------------------------------------------------------------------------------
-- TASK 10
-------------------------------------------------------------------------------------------
-- Definicja
CREATE OR ALTER FUNCTION PodatekPoglowny (@p_pseudo VARCHAR(15)) RETURNS INT
AS
BEGIN
DECLARE
    @v_podwladni_cnt INT,
    @v_wrogowie_cnt  INT,
    @v_podatek       INT         = 0,
    @v_pseudo        VARCHAR(15) = UPPER(@p_pseudo),
    @v_plec          CHAR(1);

    -- Podatek podstawowy - obliczenie 5% calkowitego przydzialu myszy
    -- dla kota o danym pseudonimie
    SELECT
        @v_podatek = ROUND(0.05 * (przydzial_myszy + ISNULL(myszy_extra, 0)), 0),
        @v_plec    = plec
    FROM Kocury
    WHERE
        pseudo = @v_pseudo;

    -- Podatek od braku podwladnych - policzenie dla ilu kotow kot o danym
    -- pseudonimie jest szefem
    SELECT
        @v_podwladni_cnt = COUNT(*)
    FROM Kocury
    WHERE
        szef = @v_pseudo;

    IF @v_podwladni_cnt = 0
        SET @v_podatek += 2;

    -- Podatek od braku wrogow - policzenie ile jest wystapien danego
    -- pseudonimu w tabeli `Wrogowie_kocurow`
    SELECT
        @v_wrogowie_cnt = COUNT(*)
    FROM Wrogowie_kocurow
    WHERE
        pseudo = @v_pseudo;

    IF @v_wrogowie_cnt = 0
        SET @v_podatek += 1;

    -- Podatek od kocurow (dodany) - podatek w postaci 5 myszy dla kocurow
    -- (plci meskiej)
    IF @v_plec = 'M'
        SET @v_podatek += 5;

    RETURN @v_podatek;
END;
GO

-- Wywolanie
```

```
DECLARE
    @pseudo_input VARCHAR(15) = 'tygrys';
BEGIN
    BEGIN TRY
        PRINT CONCAT('Podatek dla ', UPPER(@pseudo_input), ': ',
dbo.PodatekPoglowny(@pseudo_input), ' myszy');
    END TRY
    BEGIN CATCH
        PRINT ERROR_MESSAGE();
    END CATCH
END;
GO
```

```
Started executing query at  Line 161
Podatek dla TYGRYS: 12 myszy
Total execution time: 00:00:00.006
```

Dla `p_pseudo = 'tygrys'` (podwł. 6 i wrog. 2)

```
Started executing query at  Line 161
Podatek dla ZERO: 10 myszy
Total execution time: 00:00:00.004
```

Dla `p_pseudo = 'zero'` (podwł. 0 i wrog. 0)

W celu rozwiązania tego zadania należało policzyć liczbę podwładnych oraz wrogów dla kota o podanym pseudonimie, Na podstawie obliczonych wartości odpowiednio zwiększana była wartość podatku do zapłacenia. Dodana ode mnie forma podatkowa polegała na odprowadzeniu dodatkowych 5 myszy od kocurów (kotów płci męskiej).

Zadanie 11

PL/SQL

a.

```
-------------------------------------------------------------------------------
-- TASK 11
-------------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION GetFunctionCnt RETURN NUMBER
IS
    v_func_cnt NUMBER := 0;
BEGIN
    SELECT
        COUNT(DISTINCT funkcja)
    INTO
        v_func_cnt
    FROM
        Kocury;

    RETURN v_func_cnt;
END;
/

CREATE OR REPLACE FUNCTION GetSeparatorPodsumowanie (
    p_func_cnt IN NUMBER DEFAULT 0
) RETURN CLOB
IS
    v_dynamic_sql      CLOB;
    v_dynamic_func_sep CLOB   := '';
    v_index            NUMBER := 0;
BEGIN
    WHILE v_index < p_func_cnt LOOP
        v_dynamic_func_sep := CONCAT(v_dynamic_func_sep, 'RPAD(''-'', 10, ''-''),');
        v_index := v_index + 1;
    END LOOP;

    SELECT
    '
    SELECT
        RPAD(''Z'', 20, ''-''),
        RPAD(''-'', 6, ''-''),
        RPAD(''-'', 4, ''-''),
    '
    || v_dynamic_func_sep ||
    '
        RPAD(''-'', 7, ''-'')
    FROM dual

    UNION ALL

    SELECT
        ''ZJADA RAZEM'',
        '' '',
        '' '',
        '
    ||
    LISTAGG(
```

```
    '
        LPAD(SUM(DECODE(
            funkcja, ''' || funkcja || ''', przydzial_myszy + NVL(myszy_extra, 0), 0
        )), 10)',
        ', '
    ) WITHIN GROUP (ORDER BY funkcja DESC)
    ||
    '
        , LPAD(SUM(przydzial_myszy + NVL(myszy_extra, 0)), 7)
    FROM Kocury
    '
    INTO
        v_dynamic_sql
    FROM (
        SELECT DISTINCT
            funkcja
        FROM Kocury
    );

    RETURN v_dynamic_sql;
END GetSeparatorPodsumowanie;
/

-- a)
CREATE OR REPLACE PROCEDURE GetSumaSpozyciaA (
    p_rc OUT SYS_REFCURSOR
)
IS
    v_dynamic_sql CLOB;
BEGIN
    SELECT
        '
        SELECT
            RPAD(DECODE(k.plec, ''D'', b.nazwa, ''M'', '' ''), 20)      "NAZWA BANDY",
            RPAD(DECODE(k.plec, ''D'', ''Kotka'', ''M'', ''Kocor''), 6) "PLEC",
            LPAD(COUNT(*), 4)                                           "ILE",
        '
        ||
        LISTAGG(
            '
            LPAD(SUM(DECODE(
                k.funkcja, '''
                || funkcja || ''', k.przydzial_myszy + NVL(k.myszy_extra, 0),
                0
            )), 10) AS ' || funkcja, ', '
        ) WITHIN GROUP (ORDER BY funkcja DESC)
        ||
        '
        ,
            LPAD(SUM(k.przydzial_myszy + NVL(k.myszy_extra, 0)), 7)     "SUMA"
        FROM Kocury k
            INNER JOIN Bandy b ON b.nr_bandy = k.nr_bandy
        GROUP BY
            b.nazwa,
            k.plec

        UNION ALL'
```

```
        || GetSeparatorPodsumowanie(GetFunctionCnt)
    INTO
        v_dynamic_sql
    FROM (
        SELECT DISTINCT
            funkcja
        FROM Kocury
    );

    OPEN p_rc FOR v_dynamic_sql;
END GetSumaSpozyciaA;
/

VAR rc REFCURSOR
EXEC GetSumaSpozyciaA(:rc)
PRINT rc;
```

```
NAZWA BANDY         PLEC   ILE  SZEFUNIO   MILUSIA    LOWCZY     LAPACZ     KOT        DZIELCZY   BANDZIOR   SUMA
-------------------- ------ ---- ---------- ---------- ---------- ---------- ---------- ---------- ---------- --------
                    Kocor     3          0          0         67         56          0          0         93      216
SZEFOSTWO           Kotka     2          0        136          0          0          0          0          0      136
                    Kocor     2          0          0          0          0         43          0         88      131
BIALI LOWCY         Kotka     2          0         55         61          0          0          0          0      116
LACIACI MYSLIWI     Kotka     2          0          0          0         51         40          0          0       91
                    Kocor     3          0          0         65         51         40          0          0      156
                    Kocor     2        136          0          0          0          0         50          0      186
CZARNI RYCERZE      Kotka     2          0         52         65          0          0          0          0      117
Z-------------------- ------ ---- ---------- ---------- ---------- ---------- ---------- ---------- ---------- --------
ZJADA RAZEM                          136        243        258        158        123         50        181     1149

10 rows selected.
```

b.

```
-- b)
CREATE OR REPLACE PROCEDURE GetSumaSpozyciaB (
    p_rc OUT SYS_REFCURSOR
)
IS
    v_dynamic_sql CLOB;
BEGIN
    SELECT
        '
        SELECT
            RPAD(DECODE(plec, ''D'', nazwa, ''M'', '' ''), 20)         "NAZWA BANDY",
            RPAD(DECODE(plec, ''D'', ''Kotka'', ''M'', ''Kocor''), 6)  "PLEC",
            LPAD(ile, 4)                                               "ILE",
            '
            ||
            LISTAGG('LPAD(NVL(' || funkcja || ', 0), 10) AS ' || funkcja, ', ') WITHIN GROUP
(ORDER BY funkcja)
            ||
            ', LPAD('
            ||
            LISTAGG('NVL(' || funkcja || ', 0)', ' + ') WITHIN GROUP (ORDER BY funkcja)
            ||
            ', 7)    "SUMA"
        FROM (
            SELECT
                b.nazwa,
                k.plec,
                k.funkcja,
                k.przydzial_myszy + NVL(k.myszy_extra, 0) AS spozycie,
                COUNT(*) OVER (PARTITION BY nazwa, plec)  AS ile
            FROM Kocury k
                INNER JOIN Bandy b ON b.nr_bandy = k.nr_bandy
        )
        PIVOT (
            SUM(spozycie)
            FOR funkcja IN ('
            ||
            LISTAGG(
                '''' || funkcja || ''' AS ' || funkcja,
                ', '
            ) WITHIN GROUP (ORDER BY funkcja)
            ||
            ')
        )

        UNION ALL
        '
        || GetSeparatorPodsumowanie(GetFunctionCnt)
    INTO
        v_dynamic_sql
    FROM (
        SELECT DISTINCT
            funkcja
        FROM Kocury
    );
```

```
    OPEN p_rc FOR v_dynamic_sql;
END GetSumaSpozyciaB;
/

VAR rc REFCURSOR
EXEC GetSumaSpozyciaB(:rc)
PRINT rc;
```

NAZWA BANDY	PLEC	ILE	BANDZIOR	DZIELCZY	KOT	LAPACZ	LOWCZY	MILUSIA	SZEFUNIO	SUMA
BIALI LOWCY	Kotka	2	0	0	0	0	61	55	0	116
	Kocor	2	88	0	43	0	0	0	0	131
CZARNI RYCERZE	Kotka	2	0	0	0	0	65	52	0	117
	Kocor	3	93	0	0	56	67	0	0	216
LACIACI MYSLIWI	Kotka	2	0	0	40	51	0	0	0	91
	Kocor	3	0	0	40	51	65	0	0	156
SZEFOSTWO	Kotka	2	0	0	0	0	0	136	0	136
	Kocor	2	0	50	0	0	0	0	136	186
Z------------------	------	---	---------	---------	---------	---------	---------	---------	---------	--------
ZJADA RAZEM			136	243	258	158	123	50	181	1149

10 rows selected.

a.

```
-------------------------------------------------------------------------------------------
-- TASK 11
-------------------------------------------------------------------------------------------
-- a)
CREATE OR ALTER FUNCTION GetFunctionCnt () RETURNS INT
AS
BEGIN
    DECLARE
        @v_func_cnt INT;

    SELECT
        @v_func_cnt = COUNT(DISTINCT funkcja)
    FROM Kocury;

    RETURN @v_func_cnt;
END;
GO

CREATE OR ALTER FUNCTION GetSeparatorPodsumowanie (
    @p_func_cnt INT
) RETURNS NVARCHAR(MAX)
AS
BEGIN
    DECLARE
        @v_dynamic_sql      NVARCHAR(MAX) = N'',
        @v_dynamic_func_sep NVARCHAR(MAX) = N'',
        @v_index            INT           = 0;

    WHILE @v_index < @p_func_cnt
    BEGIN
        SET @v_dynamic_func_sep += N'REPLICATE(''-'', 10), '
        SET @v_index += 1
    END

    SELECT
        @v_dynamic_sql =
        N'
        SELECT
            RIGHT(''Z'' + REPLICATE(''-'', 19), 20),
            REPLICATE(''-'', 6),
            REPLICATE(''-'', 4),
        '
        + @v_dynamic_func_sep +
        N'
            REPLICATE(''-'', 7)

        UNION ALL

        SELECT
            ''ZJADA RAZEM'',
            '' '',
            '' '',
        '
        +
```

```
        STRING_AGG(
            N'
                RIGHT(
                    REPLICATE('' '', 10) +
                        CAST(SUM(CASE
                                    WHEN funkcja = ''' + funkcja + N'''
                                    THEN przydzial_myszy + ISNULL(myszy_extra, 0)
                                    ELSE 0
                                END) AS VARCHAR(50)),
                10) AS ' + funkcja,
            N', '
        ) WITHIN GROUP (ORDER BY funkcja DESC)
        +
        N'
            , RIGHT(SPACE(7) + CAST(SUM(przydzial_myszy + ISNULL(myszy_extra, 0)) AS
VARCHAR(50)), 7)
        FROM Kocury
        '
    FROM (
        SELECT DISTINCT
            funkcja
        FROM Kocury
    ) f;

    RETURN @v_dynamic_sql;
END;
GO

CREATE OR ALTER PROCEDURE GetSumaSpozyciaA
AS
BEGIN
    DECLARE
        @v_dynamic_sql NVARCHAR(MAX);

    SELECT
        @v_dynamic_sql =
        N'
        SELECT
            CASE k.plec
                WHEN ''D'' THEN b.nazwa
                WHEN ''M'' THEN '' ''
            END                                         [NAZWA BANDY],
            CASE k.plec
                WHEN ''D'' THEN ''Kotka''
                WHEN ''M'' THEN ''Kocor''
            END                                         [PLEC],
            RIGHT(REPLICATE('' '', 4) + COUNT(*), 4) [ILE],'
            +
            STRING_AGG(
                N'
                RIGHT(REPLICATE('' '', 10) +
                CAST(SUM(
                    CASE
                        WHEN k.funkcja = ''' + funkcja + '''
                            THEN k.przydzial_myszy + ISNULL(k.myszy_extra, 0)
                        ELSE 0
                    END
```

```
                ) AS VARCHAR), 10) [' + funkcja + ']',
                ', '
            )
            +
            N',
                RIGHT(REPLICATE('' '', 7) + CAST(SUM(k.przydzial_myszy + ISNULL(k.myszy_extra,
0)) AS VARCHAR), 7) [SUMA]
            FROM Kocury k
                INNER JOIN Bandy b ON b.nr_bandy = k.nr_bandy
            GROUP BY
                b.nazwa,
                k.plec

            UNION ALL
            '
            + dbo.GetSeparatorPodsumowanie(dbo.GetFunctionCnt())
    FROM (
        SELECT DISTINCT
            funkcja
        FROM Kocury
    ) f;

    PRINT @v_dynamic_sql;

    EXEC sp_executesql @v_dynamic_sql;
END;
GO

EXEC GetSumaSpozyciaA;
GO
```

NAZWA BANDY	PLEC	ILE	BANDZIOR	DZIELCZY	KOT	LAPACZ	LOWCZY	MILUSIA	SZEFUNIO	SUMA
BIALI LOWCY	Kotka	2	0	0	0	0	61	55	0	116
CZARNI RYCERZE	Kotka	2	0	0	0	0	65	52	0	117
LACIACI MYSLIWI	Kotka	2	0	0	40	51	0	0	0	91
SZEFOSTWO	Kotka	2	0	0	0	0	0	136	0	136
	Kocor	2	88	0	43	0	0	0	0	131
	Kocor	3	93	0	0	56	67	0	0	216
	Kocor	3	0	0	40	51	65	0	0	156
	Kocor	2	0	50	0	0	0	0	136	186
Z----------	------	----	----------	----------	----------	----------	----------	----------	----------	-------
ZJADA RAZEM			136	243	258	158	123	50	181	1149

((10 rows affected))

b.

```
-- b)
CREATE OR ALTER PROCEDURE GetSumaSpozyciaB
AS
BEGIN
    DECLARE
        @v_dynamic_sql NVARCHAR(MAX);

    SELECT
        @v_dynamic_sql =
        N'
        SELECT
            CASE plec
                WHEN ''D'' THEN nazwa
                WHEN ''M'' THEN '' ''
            END                                                 [NAZWA BANDY],
            CASE plec
                WHEN ''D'' THEN ''Kotka''
                WHEN ''M'' THEN ''Kocor''
            END                                                 [PLEC],
            RIGHT(REPLICATE('' '', 4) + CAST(ile AS VARCHAR), 4) [ILE],
        '
        +
        STRING_AGG(
            'RIGHT(REPLICATE('' '', 10) + CAST(ISNULL(' + QUOTENAME(funkcja) + ', 0) AS
VARCHAR), 10) ' + QUOTENAME(funkcja),
            ', '
        ) WITHIN GROUP (ORDER BY funkcja)
        +
        ',
            RIGHT(REPLICATE('' '', 7) + CAST(' +
        STRING_AGG('ISNULL(' + funkcja + ', 0)', ' + ') WITHIN GROUP (ORDER BY funkcja) +
        '   AS VARCHAR), 7)                                     [SUMA]
        FROM (
            SELECT
                b.nazwa,
                k.plec,
                k.funkcja,
                k.przydzial_myszy + ISNULL(k.myszy_extra, 0) AS spozycie,
                COUNT(*) OVER (PARTITION BY b.nazwa, k.plec) AS ile
            FROM Kocury k
                INNER JOIN Bandy b ON b.nr_bandy = k.nr_bandy
        ) src
        PIVOT (
            SUM(spozycie)
            FOR funkcja IN ('
            +
            STRING_AGG(
                QUOTENAME(funkcja),
                ', '
            ) WITHIN GROUP (ORDER BY funkcja)
            +
            ')
        ) pvt

        UNION ALL
```

```
        '
        + dbo.GetSeparatorPodsumowanie(dbo.GetFunctionCnt())
    FROM (
        SELECT DISTINCT
            funkcja
        FROM Kocury
    ) f;

PRINT @v_dynamic_sql;
    EXEC sp_executesql @v_dynamic_sql;
END;
GO

EXEC GetSumaSpozyciaB;
GO
```

NAZWA BANDY	PLEC	ILE	BANDZIOR	DZIELCZY	KOT	LAPACZ	LOWCZY	MILUSIA	SZEFUNIO	SUMA
BIALI LOWCY	Kotka	2	0	0	0	0	61	55	0	116
	Kocor	2	88	0	43	0	0	0	0	131
CZARNI RYCERZE	Kotka	2	0	0	0	0	65	52	0	117
	Kocor	3	93	0	0	56	67	0	0	216
LACIACI MYSLIWI	Kotka	2	0	0	40	51	0	0	0	91
	Kocor	3	0	0	40	51	65	0	0	156
SZEFOSTWO	Kotka	2	0	0	0	0	0	136	0	136
	Kocor	2	0	50	0	0	0	0	136	186
Z----------	------	----	----------	----------	----------	----------	----------	----------	----------	--------
ZJADA RAZEM			136	243	258	158	123	50	181	1149

((10 rows affected))

Ogólne spostrzeżenia po wykonaniu zadań z części *Zadania - procedury*

Większość uwag została przeze mnie wspominania już w podsumowaniu poprzedniej części zadań, tutaj dodać mogę jedynie dość oczywiste różnice w składni definiowania procedur i funkcji. Jedyną z mojej perspektywy istotną różnicą (lecz może ona dotyczyć również bloków) jest sposób obsługi wyjątków. W PL/SQL mamy do tego polecenie `EXCEPTION`, w ramach którego można specyfikować działanie w przypadku zgłoszenia dowolnego wyjątku. W T-SQL z kolei korzystać należy z bloków `TRY-CATCH`.

Jeśli chodzi o dynamicznego SQL, pisze się go w obu dialektach niemal identycznie, jednakże sama technika jest dość trudna do realizacji, ponieważ brakuje odpowiedniego sposobu debugowania kodu.