ZAJĘCIA VII
FUNKCJE, PROCEDURY, SEKWENCJE

Zad. 1
Stworzyć dwie sekwencje: obie z wartościami początkowymi 1 i krokiem 1 (jedna z sekwencji powinna mieć wartość maksymalną równą 2 oraz mieć cykl). Stworzyć tabelę o trzech kolumnach numerycznych, (dwie pierwsze wchodzą w skład klucza głównego). Wykorzystując uprzednio utworzone sekwencje wstawić przy użyciu pętli `2*n` (założenie: `n>2`) wierszy do tabeli. Przykład (`n=3`):

```
1     1     2
1     2     3
2     1     3
2     2     4
3     1     4
3     2     5
```

Zad. 2
Stworzyć cztery sekwencje. Pierwsza powinna spełniać następujące warunki: wartość początkowa 5, krok 3, wartość maksymalna 12 (druga z ustaloną wartością minimalną), trzecia: wartość początkowa 100, krok -10, wartość maksymalna 110, wartość minimalna 10, czwarta: wartość początkowa -5, krok -10, wartość minimalna -40. Wypisać 5 kolejnych wartości sekwencji pierwszej i drugiej i 11 sekwencji trzeciej i czwartej. (Wszystkie sekwencje z cyklem).

WYWOŁANIE FUNKCJI

```
variable zm number;
execute :zm:=zad3('PROJEKT NR 1', 2300);
print :zm
```

Zad. 3
Stworzyć funkcję, która dla podanego projektu (nazwa projektu parametr) zwróci ilu pracowników, którzy realizowali dany projekt, zarabia więcej niż wartość drugiego parametru funkcji. Obsłużone powinny być sytuacje: (1) nie ma projektu o podanej nazwie, (2) jest wiele projektów o podanej nazwie. Sytuacjom tym powinien towarzyszyć odpowiedni komunikat.

Zad. 4
Stworzyć funkcję, która zwróci jaki procent średnich zarobków stanowią zarobki pracowników, którzy mają podwładnych.

WYWOŁANIE PROCEDURY

```
variable procent number;
execute zad5(3, :procent);
print :procent
```

Zad. 5

Stworzyć procedurę, która dla danego pracownika (id_pracownika parametrem procedury) sprawdzi z jakim procentem ogółu pracowników nigdy nie współpracował (wypisanie na ekran).

Zad. 6

Stworzyć procedurę, która dla danego projektu wyznaczy: minimalną i maksymalną stawkę za godzinę dla danego projektu oraz jaka jest różnica pomiędzy budżetem a kwotą, którą wydano na realizację danego projektu.

POMOC

1. Tworzenie funkcji

```
CREATE [OR REPLACE] FUNCTION n_funkcji[(parametry)] RETURN typ_danych IS
[DEKLARACJE]
BEGIN
      INSTRUKCJE
[EXCEPTION
      obsluga wyjatkow]
END [nazwa funkcji];
/
```

Zwracanie wartości: polecenie `RETURN(wartość);`

2. Tworzenie procedur

```
CREATE [OR REPLACE] PROCEDURE n_procedury[(parametry)] IS
[DEKLARACJE]
BEGIN
      INSTRUKCJE
[EXCEPTION
      obsluga wyjatkow]
END [nazwa procedury];
/
```

3. Wyświetlenie błędów

```
SHOW ERRORS
```

4. Parametry

```
parametr::= nazwa [IN | OUT | IN OUT] typ_danych [ {:= | DEFAULT} wyrażenie]
```

Domyślnie: `IN`

`IN`
- jedynie przekazanie wartości do programu (nie można zmieniać).

`OUT`
- wartość parametru jest (po prawidłowym wykonaniu podprogramu) zwracana do jednostki wołającej,
- próba odwołania się do parametru (w podprogramie) przed przypisaniem: błąd (niezainicjalizowana zmienna).

`IN OUT`
- możliwe jest przekazywanie wartości do podprogramu i zwracanie wartości do wołającej jednostki.

5. Sekwencje

```
DESC USER_SEQUENCES;
SELECT * FROM USER_SEQUENCES WHERE SEQUENCE_NAME='NAZWA_SEKW';

CREATE SEQUENCE seqname
      [ INCREMENT BY increment ]
      [ MINVALUE minvalue ]
      [ MAXVALUE maxvalue ]
      [ START WITH start ]
      [ CACHE cache | NOCACHE ]
      [ CYCLE ];
```

CACHE

- tyle kolejnych wartości sekwencji (w jednym cyklu) jest w pamięci podręcznej,
- musi być>1,
- domyślnie jest=20.

Zmiana parametrów sekwencji

```
ALTER SEQUENCE nazwa_sek
zmiany;
```

Przykład:

```
ALTER SEQUENCE sekwencja MAXVALUE 300;
      -- nowa wartość MAXVALUE=300
```

Odwoływanie się do sekwencji

- NEXTVAL, CURRVAL: pseudokolumny,
- nazwa_sek.NEXTVAL, nazwa_sek.CURRVAL,
- aby było możliwe odczytanie wartości sekwencji pierwszy raz należy się odwołać do NEXTVAL,
- jeśli w jednym poleceniu odwołamy się do NEXTVAL, to każde kolejne odwołanie się do NEXTVAL w tym samym poleceniu zwróci tę samą wartość,
- pobranie kolejnej wartości: nazwa_sek.NEXTVAL.

Ograniczenia w użyciu NEXTVAL i CURRVAL

MOŻNA UŻYWAĆ:

a) po słowie VALUES przy wstawianiu wierszy (klauzula INSERT),
- po słowie SELECT,
- po słowie SET przy zmianie wartości (klauzula UPDATE).

NIE MOŻNA UŻYWAĆ:

- w podzapytaniu,
- w zapytaniach do perspektyw,
- w zapytaniach z użyciem DISTINCT, GROUP BY, ORDER BY,
- po słowie WHERE,
- jako wartości domyślnej (DEFAULT) przy tworzeniu lub zmienianiu definicji tabeli (CREATE, ALTER),
- w warunku CHECK.

```
SELECT seqname.nextval FROM dual; -- *
SELECT seqname.currval FROM dual;
```

* (to polecenie musi wystąpić jako pierwsze)

6. Zmienne

Zmienne związane (*bind variables*)

Używane są do przechowywania danych lub wyników.

b) deklaracja
```
variable nazwa_zmiennej typ_danych;
```
c) wypisanie dostępnych zmiennych
```
variable
```
d) odwołanie się do zmiennych: poprzedza sie ':'
e) wypisanie wartości zmiennej
```
print nazwa_zmiennej;
```

Wywoływanie funkcji, procedur SQL+

a) funkcji
```
execute :zmienna:=nazwa_funkcji;
```
b) procedury
```
execute nazwa_procedury;
```

Wywoływanie funkcji, procedur PL/SQL

a) funkcji
```
zmienna:=nazwa_funkcji;
```
--(jeśli zmienna związana, to jej nazwa poprzedzona ':')
b) procedury
```
nazwa_procedury;
```

Zmienne użytkownika

a) deklaracja
```
DEFINE zmienna=wartość_początkowa;
```
--zmienna ta ma typ danych `VARCHAR`
b) usunięcie zmiennej
```
UNDEFINE zmienna;
```
c) odwołanie się do zmiennej: `&zmienna`

Przykład:

```
define nazw_def='Kowalski';

SELECT * FROM Pracownik WHERE nazwisko='&nazw_def';

define id_def=1;

SELECT * FROM Pracownik WHERE id_pracownika=&id_def;
```

Jeśli nie ma wartości to użytkownik zostanie poproszony o jej podanie, wartość ta zostanie wykorzystana do wykonania jednego, bieżącego polecenia a później przy kolejnym odwołaniu `&zmienna` znowu użytkownik zostanie poproszony o podanie wartości zmiennej.

`&&zmienna`: wartość pobrana i zapamiętana