Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра ПМиК

Практическое задание №6

по дисциплине «Сетевые базы данных»

“Коллекции”

Выполнил:
студент гр. МГ-211 _____ / Бурдуковский И.А./
подпись

Проверил:
Доцент
кафедры ПМиК _____ / Приставка П.А./

Новосибирск 2023 г.

Оглавление

Задание

Ассоциативные массивы

1. Создать таблицу БД, содержащую информацию об автомобилях, допускаемых к въезду на территорию парковки организации (уникальный признак – гос. номер). Заполнить ее тестовыми данными. Реализовать кэширующую в коллекцию типа ассоциативный массив процедуру поиска въезжающего автомобиля. Коллекция должна обеспечивать учет статусов хранящихся в ней автомобилей – «въехал» и «выехал». Реализовать процедуру поиска автомобиля по гос. номеру, предоставляющую информацию о том въезжал ли искомый автомобиль на территорию в отчетном периоде (период сессии) и его текущем статусе. (Указания: Определение типа коллекции и объявление ассоциативного массива реализовать внутри пакета. Автомобиль считается въезжавшим на территорию, если соответствующий элемент находится в коллекции).

Массивы перменной длины (VARRAY)

2. Определить хранимый в БД тип коллекции (тип уровня схемы) соответствующий массиву переменной длины. Создать в БД таблицу, хранящую информацию о маршруте: название, список населенных пунктов, входящих в маршрут, в порядке следования. Список населенных пунктов хранить в поле таблицы в виде коллекции типа VARRAY. Заполнить таблицу тестовыми данными.Реализовать процедуру добавления нового населенного пункта в заданный маршрут.

Вложенные таблицы (NESTED TABLES)

3. Определить хранимый в БД тип коллекции (тип уровня схемы) соответствующий вложенной таблице. Создать в БД таблицу, содержащую информацию о стране и цветах ее флага. Цвета флага хранить коллекции типа NESTED TABLES непосредственно в поле таблицы. Заполнить ее тестовыми данными (названия стран). Реализовать процедуру внесения цветовой гаммы для заданной страны.

Операции с мультимножествами

4. Реализовать функцию, которая для заданных стран из таблицы п. 3 возвращает список общих цветов. Реализовать процедуру распечатку списка общих цветов.

Выполнение

Ассоциативные массивы

1. Ассоциативные массивы. Создать таблицу БД, содержащую информацию об автомобилях, допускаемых к въезду на территорию парковки организации (уникальный признак – гос. номер). Заполнить ее тестовыми данными.

Реализовать кэширующую в коллекцию типа ассоциативный массив процедуру поиска въезжающего автомобиля. Коллекция должна обеспечивать учет статусов хранящихся в ней автомобилей – «въехал» и «выехал».

Реализовать процедуру поиска автомобиля по гос. номеру, предоставляющую информацию о том въезжал ли искомый автомобиль на территорию в отчетном периоде (период сессии) и его текущем статусе. (Указания: Определение типа коллекции и объявление ассоциативного массива реализовать внутри пакета. Автомобиль считается въезжавшим на территорию, если соответствующий элемент находится в коллекции).

Создание таблицы:

```
CREATE TABLE car (numer VARCHAR2(100) );
ALTER TABLE car
 ADD (CONSTRAINT car_pk_numer PRIMARY KEY (numer));
```

Заполнение таблицы:

```
INSERT INTO car VALUES (‘B555PX39’);
INSERT INTO car VALUES ('C976MM78');
INSERT INTO car VALUES ('K976MM78');
INSERT INTO car VALUES ('M976MM77');
INSERT INTO car VALUES ('O000OO19');
INSERT INTO car VALUES ('X777OY17');
```

Реализация задания:

```
CREATE OR REPLACE PACKAGE parking
IS
   PROCEDURE set_status
         (num IN car.pnum%TYPE, value_status IN BOOLEAN);
   PROCEDURE search_by_number
         (num IN car.pnum%TYPE);
END parking;
/

CREATE OR REPLACE PACKAGE BODY parking
IS
```

```
TYPE carType IS TABLE OF BOOLEAN
        INDEX BY car.pnum%TYPE;
incomers carType;

PROCEDURE set_status
        (num IN car.pnum%TYPE, value_status IN BOOLEAN)
IS
        FUNCTION check_car RETURN BOOLEAN IS
                value_number car.pnum%TYPE;
        BEGIN
      IF incomers.EXISTS(num) THEN
        RETURN TRUE;
      END IF;
                SELECT pnum INTO value_number
                        FROM car WHERE pnum = num;
                RETURN TRUE;
        EXCEPTION
                WHEN NO_DATA_FOUND THEN
                        RETURN FALSE;
                END;

BEGIN
        IF check_car THEN
      incomers(num) := value_status;
                IF NOT incomers.EXISTS(num) THEN
                        DBMS_OUTPUT.PUT_LINE('Въезд разрешен, но у водителя не
получилось заехать: ' || num || '.');
                ELSE
          IF value_status THEN
            DBMS_OUTPUT.PUT_LINE('Заехал: '  || num || '.');
          ELSE
            DBMS_OUTPUT.PUT_LINE('Выехал: ' || num || '.');
                   END IF;
                END IF;
        ELSE
                DBMS_OUTPUT.PUT_LINE('Въезд запрещен: ' || num || '.');
        END IF;
END set_status;

PROCEDURE search_by_number (num IN car.pnum%TYPE)
IS
BEGIN
        IF incomers.EXISTS(num) THEN
                IF incomers(num) THEN
```

```
                DBMS_OUTPUT.PUT_LINE('Автомобиль ' || num || ' находится на
парковке.');
            ELSE
                DBMS_OUTPUT.PUT_LINE('Автомобиль ' || num || ' выехал с
парковки.');
            END IF;
        ELSE
            DBMS_OUTPUT.PUT_LINE('Автомобиль ' || num || ' никогда не был на этой
парковке.');
        END IF;
    END search_by_number;
END parking;
```

Проверка:

```
BEGIN
    parking.set_status('K976MM78', TRUE);
    parking.set_status('B555PX39', TRUE);
    parking.set_status('K976MM78', FALSE);
    parking.search_by_number('K976MM78');
    parking.search_by_number('B555PX39');
    parking.search_by_number('C976MM78');
END;
```

Вывод:

Заехал: K976MM78.
Заехал: B555PX39.
Выехал: K976MM78.
Автомобиль K976MM78 выехал с парковки.
Автомобиль B555PX39 находится на парковке.
Автомобиль C976MM78 никогда не был на этой парковке.

Массивы перменной длины (VARRAY).

2. Определить хранимый в БД тип коллекции (тип уровня схемы) соответствующий массиву переменной длины.

Создать в БД таблицу, хранящую информацию о маршруте: название, список населенных пунктов, входящих в маршрут, в порядке следования. Список населенных пунктов хранить в поле таблицы в виде коллекции типа VARRAY. Заполнить таблицу тестовыми данными. Реализовать процедуру добавления нового населенного пункта в заданный маршрут.

Создание типа и таблицы:

```
CREATE OR REPLACE TYPE LOCALITY_T IS VARRAY(50) OF VARCHAR2(50);
DROP TABLE routes;
CREATE TABLE routes (
 title VARCHAR2(100),
 locality LOCALITY_T
 );
```

Заполнение таблицы:

```
BEGIN
 INSERT INTO routes VALUES ('Новосибирск-Томск',
LOCALITY_T('Новосибирск', 'Юрга', 'Томск'));
 INSERT INTO routes VALUES ('Новосибирск-Барнаул',
LOCALITY_T('Новосибирск', 'Бердск', 'Черепаново', 'Барнаул'));
 INSERT INTO routes VALUES ('Омск-Красноярск',
LOCALITY_T('Омск', 'Татарск', 'Барабинск', 'Кемерово', 'Красноярск'));
END;
```

Скрипт:

```
CREATE OR REPLACE PROCEDURE add_locality (value_in IN VARCHAR2,
title_in
   IN VARCHAR2, pos IN INTEGER)
IS
   lists LOCALITY_T;
   current VARCHAR2(50);
   next VARCHAR2(50);
BEGIN
   SELECT locality INTO lists FROM routes WHERE title = title_in;

   DBMS_OUTPUT.PUT_LINE('Старый маршрут:');
   FOR element IN 1 .. lists.COUNT
   LOOP
      DBMS_OUTPUT.PUT_LINE(lists(element));
   END LOOP;

   IF lists.COUNT < 50 AND (pos >= 1 AND pos <= lists.COUNT + 1) THEN
      lists.EXTEND;
      FOR element IN 1 .. lists.COUNT
      LOOP
         IF element = pos THEN
```

```
            current := lists (element);
            lists (element) := value_in;
          ELSIF element > pos THEN
            next := lists (element);
            lists (element) := current;
            current := next;
          END IF;
        END LOOP;
        UPDATE routes SET locality = lists WHERE title = title_in;
    END IF;
    SELECT locality INTO lists FROM routes WHERE title = title_in;

    DBMS_OUTPUT.PUT_LINE('Новый маршрут:');
    FOR element IN 1 .. lists.COUNT
    LOOP
      DBMS_OUTPUT.PUT_LINE(lists(element));
    END LOOP;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Маршрут с названием ' || title_in || '
  отсутствует');
  END add_locality;/
```

Проверка:

```
  BEGIN
    add_locality ('Новосибирск', 'Омск-Красноярск', 4);
  END;
```

Вывод:

Старый маршрут:
Омск
Татарск
Барабинск
Кемерово
Красноярск

Новый маршрут:
Омск
Татарск
Барабинск
Новосибирск
Кемерово
Красноярск

Вложенные таблицы (NESTED TABLES).

3. Определить хранимый в БД тип коллекции (тип уровня схемы) соответствующий вложенной таблице. Создать в БД таблицу, содержащую информацию о стране и цветах ее флага. Цвета флага хранить коллекции типа NESTED TABLES непосредственно в поле таблицы. Заполнить ее тестовыми данными (названия стран). Реализовать процедуру внесения цветовой гаммы для заданной страны.

Создание типа и таблицы:

```
CREATE OR REPLACE TYPE COLORS_T AS TABLE OF VARCHAR2(50);
DROP TABLE country;
CREATE TABLE country (
 title VARCHAR2(100),
 colors COLORS_T
 ) NESTED TABLE colors STORE AS colors_model_tab;
```

Заполнение таблицы:

```
BEGIN
INSERT INTO country VALUES ('Россия', colors_t('белый', 'синий', 'красный'));
 INSERT INTO country VALUES ('Китай', colors_t('красный', 'желтый'));
END;
```

Скрипт:

```
CREATE OR REPLACE PROCEDURE set_color_to_flag (title_in IN
VARCHAR2,
colors_in IN COLORS_T)
IS
   colors COLORS_T;
BEGIN
   SELECT colors INTO colors FROM country WHERE title = title_in;

   DBMS_OUTPUT.PUT_LINE('Старый флаг:');
   FOR element IN 1 .. colors.COUNT
   LOOP
      DBMS_OUTPUT.PUT_LINE(colors (element));
   END LOOP;

   UPDATE country SET colors = colors_in WHERE title = title_in;
   colors := colors_in;

   DBMS_OUTPUT.PUT_LINE('Новый флаг:');
   FOR element IN 1 .. colors.COUNT
   LOOP
      DBMS_OUTPUT.PUT_LINE(colors (element));
   END LOOP;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Нет страны ' || title_in || '!');
END set_color_to_flag;
/
```

Проверка:

```
BEGIN
set_color_to_flag('Китай', COLORS_T('желтый', 'синий', 'красный')); END;
```

Вывод:

```
Старый флаг:
красный
желтый
Новый флаг:
желтый
синий
красный
```

Операции с мультимножествами.

4. Реализовать функцию, которая для заданных стран из таблицы п. 3 возвращает список общих цветов. Реализовать процедуру распечатку списка общих цветов.

Скрипт:

```
CREATE OR REPLACE FUNCTION get_general_colors
   RETURN COLORS_T
AS
   general_colors COLORS_T := COLORS_T ();
   first_colors COLORS_T;
BEGIN
   FOR rec IN (SELECT title from country)
   LOOP
      SELECT colors INTO first_colors FROM country WHERE title = rec.title;
      IF general_colors.COUNT = 0 THEN
         general_colors := first_colors;
      ELSE
         general_colors := general_colors MULTISET INTERSECT first_colors;
      END IF;
   END LOOP;
RETURN general_colors;
END;
/

CREATE OR REPLACE PROCEDURE print_general_colors (list_colors IN
COLORS_T )
AS
BEGIN
   FOR element IN 1 .. list_colors.COUNT
   LOOP
      DBMS_OUTPUT.PUT_LINE(list_colors (element));
```

```
    END LOOP;
  END;
  /
```

Проверка:

```
  BEGIN
  print_general_colors (get_general_colors);
  END;
```

Вывод:

```
  красный
```