Maciej Sawicki

Sprawozdanie - Zadanie 2

1. Polecenie wywołania SQL*Loader pozwalające załadować dane na serwer bazy danych wraz z plikiem

```
Użyty został Program `DataGrip`.
```

2. Polecenia tworzenia tabel, sekwencji, pakietu i wyzwalacza (wraz z poleceniem, które będzie aktywować ten wyzwalacz)

```
CREATE TABLE Position
(
  id   NUMBER(6)     NOT NULL,
  name VARCHAR2(100) NOT NULL UNIQUE,
  PRIMARY KEY (id)
);

CREATE TABLE Speciality
(
  id   NUMBER(6)     NOT NULL,
  name VARCHAR2(100) NOT NULL UNIQUE,
  PRIMARY KEY (id)
);

CREATE TABLE Worker
(
  id            NUMBER(6) NOT NULL,
  name          VARCHAR2(100),
  surname       VARCHAR2(100),
  birth_date    VARCHAR2(100),
  email         VARCHAR2(100),
  hire_date     VARCHAR2(100),
  salary        NUMBER(10),
  position_id   NUMBER(6) NOT NULL,
  speciality_id NUMBER(6) NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (position_id) REFERENCES Position (id),
  FOREIGN KEY (speciality_id) REFERENCES Speciality (id)
);

CREATE SEQUENCE worker_id_generator
  INCREMENT BY 1
  START WITH 1
  NOCYCLE;

CREATE SEQUENCE position_id_generator
  INCREMENT BY 1
  START WITH 1
  NOCYCLE;
```

```
CREATE SEQUENCE speciality_id_generator
  INCREMENT BY 1
  START WITH 1
  NOCYCLE;

--------------------- task2 PACKAGE ----------------------------
CREATE OR REPLACE PACKAGE task2 AS
  PROCEDURE migrate_tables;

  FUNCTION calculate_mean_salary(position_name Position.name%TYPE)
    RETURN NUMBER;
END;

--------------------- task2 PACKAGE IMPLEMENTATION ------------
CREATE OR REPLACE PACKAGE BODY task2
AS
  PROCEDURE migrate_tables AS
    BEGIN
      -- TEMP -> Position
      FOR i IN (SELECT DISTINCT t.POSITION
                FROM TEMP t) LOOP
        INSERT INTO Position VALUES (position_id_generator.nextval, i.POSITION);
      END LOOP;

      -- TEMP -> Speciality
      FOR i IN (SELECT DISTINCT t.SPECIALITY
                FROM TEMP t) LOOP
        INSERT INTO Speciality VALUES (speciality_id_generator.nextval,
i.SPECIALITY);
      END LOOP;

      -- TEMP -> Worker
      FOR i IN (SELECT
                  t.name,
                  t.surname,
                  t.BIRTH,
                  t.email,
                  t.hire_date,
                  t.salary,
                  t.POSITION,
                  t.SPECIALITY
                FROM TEMP t) LOOP
        INSERT INTO Worker
        VALUES (worker_id_generator.nextval,
                i.NAME,
                i.SURNAME,
                i.BIRTH,
                i.EMAIL,
                i.HIRE_DATE,
                i.SALARY,
                (SELECT p.id
                 FROM Position p
                 WHERE p.name = i.POSITION),
                (SELECT s.id
                 FROM Speciality s
                 WHERE s.name = i.SPECIALITY));
      END LOOP;
```

```
      EXCEPTION
      WHEN OTHERS
      THEN NULL;
    END;

  FUNCTION calculate_mean_salary(position_name Position.name%TYPE)
    RETURN NUMBER IS
    average NUMBER;
    BEGIN
      SELECT AVG(w.salary)
      INTO average
      FROM Worker w, POSITION p
      WHERE w.position_id = p.id
            AND p.name = position_name;
      RETURN average;
    END;
END;

------------------  salary_guard TRIGGER  -----------------------
CREATE OR REPLACE TRIGGER salary_guard
  BEFORE UPDATE OF position_id, salary OR INSERT
  ON Worker
  FOR EACH ROW
  DECLARE
    PRAGMA AUTONOMOUS_TRANSACTION;
    position_name  VARCHAR(100);
    average_salary NUMBER;
      salaryException EXCEPTION;
  BEGIN
    SELECT Position.name
    INTO position_name
    FROM Position
    WHERE ID = :new.position_id;

    average_salary := task2.calculate_mean_salary(position_name);
    dbms_output.put_line('average ' || average_salary);
    IF (average_salary * 1.25) < :new.salary
    THEN
      RAISE salaryException;
    END IF;
  END;
```

3. Polecenia wywołania procedury i funkcji z pakietu (dla funkcji proszę też zamieścić wynik zwracany przez funkcję)

Wywołanie procedury `migrate_tables`

```
CALL task2.migrate_tables();
````

Wywołanie funkcji `calculate_mean_salary(VARCHAR2(100))`

```sql
DECLARE result NUMBER;
BEGIN
```

```
  result := task2.calculate_mean_salary('Sales Associate');
  dbms_output.put_line('Average salary for Sales Associate: ' || result);
END;
```

Wynik:

```
Average salary for Sales Associate: 48428,5714285714285714285714285714285714
```

4. Wynik następujących zapytań

```
SELECT * FROM Worker;

1   Davies  Josh    10/7/1984   joshl@jlee.com  9/15/2004   50000   4   2
2   Tower   Derek   4/19/1989   derek7@jlee.com 9/15/2004   50000   4   1
3   Lender  Xi  12/4/1983   xi3@jlee.com    9/15/2004   50000   4   1
4   Cameron Rotger  1/27/1990   rotger1@jlee.com    9/15/2004   50000   4   1
5   Dobos   Phoebe  2/25/1981   phoebe1@jlee.com    9/15/2004   50000   4   2
6   Reeves  Marisa  9/23/1978   marisa6@jlee.com    9/15/2004   65000   2   2
7   Whitmore    Carla   5/2/1969    carla1@jlee.com 9/15/2004   45000   5   2
8   Gupta   Alain   11/12/1977  alain1@jlee.com 9/15/2004   40000   5   1
9   Kilborn Clive   4/25/1975   clive1@jlee.com 9/15/2004   57000   6   2
10  Bacon   John    5/22/1974   john9@jlee.com  9/15/2004   60000   1   1
11  Szabo   Lucien  12/19/1986  lucien5@jlee.com    9/15/2004   50000   8   1
12  Rango   Gordon  12/6/1985   gordon2@jlee.com    6/13/2005   49000   4   2
13  Aquila  Clancy  3/28/1979   clancy2@jlee.com    6/13/2005   52000   4   2
14  Szabo   Luke    4/19/1987   luke2@jlee.com  6/13/2005   51000   4   1
15  Bauer   Justin  4/14/1984   justin3@jlee.com    6/13/2005   50000   4   1
16  Chen    Julien  8/25/1975   julien4@jlee.com    6/13/2005   60000   2   1
17  Bellows Annie   2/19/1989   annie3@jlee.com 9/7/2007    50000   4   2
18  Rolston Salvador    4/20/1987   salvador4@jlee.com  9/7/2007    50000   4   2
19  Owens   Grant   3/4/1991    grant2@jlee.com 6/1/2011    40000   4   1
20  Porter  Cliff   2/5/1990    cliff3@jlee.com 6/1/2011    40000   4   1
21  Murray  Jon 11/23/1986  jon5@jlee.com   6/1/2011    50000   7   1
22  Goodman Gloria  6/21/1965   gloria5@jlee.com    6/1/2011    60000   3   2
23  Builder Victor  11/13/1991  victor4@jlee.com    7/25/2011   33000   9   2
24  Shah    Kunal   2/5/1993    kunal7@jlee.com 7/25/2011   33000   9   1
25  Couric  Holly   7/26/1992   holly5@jlee.com 7/25/2011   33000   9   1
26  Wells   Adam    7/5/1991    adamw@jlee.com  8/1/2011    46000   4   2
27  Travis  John    6/30/1968   johnt@jlee.com  8/1/2011    52000   8   2
```

```
SELECT * FROM Position;

1   Account Manager
2   Senior Sales Associate
3   Communications Specialist
4   Sales Associate
5   Administrative Assistant
6   Education Analyst
7   IT Analyst
8   Marketing Specialist
9   Sales Trainee
```

```
SELECT * FROM Speciality;

1   Fiction
2   Science and Engineering
```

5. Winoski

Dobrze jest wiedzieć, że istnieje ulepszenie zwykłego SQLa, gdzie mogą być stosowane struktury z programowania w innych językach, tj. pętle, warunki, zmienne, funkcje, itp. Pojawiły się spore problemy, ze zrozumieniem idei niektórych rzeczy takich, takich jak np.: wywołanie funkcji może być tylko wtedy, gdy jej wynik przypisujemy do zmiennej; albo błędy spowodowane używaniem selectow w ciele triggera. Fajnie, że coś takiego jest, ale na ten moment nie widzę swojej przyszłości związanej z tym tematem.