**Курсова работа**

**по Бази данни: Big Data**

**Изготвил:** Петър Димитров Попов

**Класно отделение:** 12282

**Факултетен номер:** 111-7419

Вариант 02\_6.

**Съдържание**

[0. Подготвителни процедури и функции 3](#_Toc1)

[1. Създайте функция GET\_YEARS\_SERVICE за извличане на брой години служба за определен служител. Функцията трябва да очаква аргумент – номер на служител и да връща брой години служба. Добавете грешка, отчитаща невалиден номер на служител. Използвайте функции MONTHS\_BETWEEN и ROUND. 3](#_Toc2)

[2. Създайте PL/SQL блок за извличане на данни за служители от таблица employees с EMPLOYEE\_ID между 100 и 108. Със стойностите, извлечени от таблица employees, попълнете PL/SQL таблица със стаж (брой години служба) по-голям от15. Изведете информация от PL/SQL таблицата. Използвайте функцията GET\_YEARS\_SERVICE. 4](#_Toc3)

[3. Напишете анонимен PL/SQL блок с параметричен курсор (параметър - номер на отдел) за номер на служител, номер на отдел за служители със заплата по-малка от 2700 от таблица emp (копие на таблица employees). Като стойност на параметъра използвайте отдел 30 и променете номера на отдела с нов номер 20 и увеличете заплатите с 5%. 6](#_Toc4)

[4. Създайте съхранена процедура ADD\_DEPARTMENT за добавяне на ред в таблица DEPT (копие на DEPARTMENTS), която да има 4 параметъра. Първият параметър е за department\_id, а следващите са за department\_title, локацията и номера на началника на отдела. Да се прави проверка за наличието на такъв номер на отдел, за наличието на такова име на отдел, за наличието на локацията в таблица locations и наличието на номер на служител в таблица employees както и служителят да не е началник на друг отдел. Ако изискванията не са изпълнени, да се вдигнат съответни изключения и за тези данни да се създаде запис в помощна таблица ERROR\_DEPART със структурата на таблица DEPARTMENTS. 7](#_Toc5)

[а)създаване на таблица DEPT и ERROR\_DEPART 8](#_Toc6)

[б) създаване на функциите за проверка на наличност на записи в отделните таблици 8](#_Toc7)

[в) създаване на процедурата ADD\_DEPARTMENT 10](#_Toc8)

[г) тестване на процедурата ADD\_DEPARTMENT 12](#_Toc9)

###### 0. Подготвителни процедури и функции

Създаване на процедура DROP\_TABLE\_IF\_EXISTS за по-лесно изтриване на таблици, с обработка на грешка при липса на наличието им.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE DROP\_TABLE\_IF\_EXISTS(param\_table\_name IN ALL\_TAB\_COLUMNS.TABLE\_NAME%TYPE)  IS  BEGIN  EXECUTE IMMEDIATE 'DROP TABLE ' || param\_table\_name;  EXCEPTION  WHEN OTHERS THEN  IF *SQLCODE* != -942 THEN  RAISE;  END IF;  END; |

###### 1. Създайте функция GET\_YEARS\_SERVICE за извличане на брой години служба за определен служител. Функцията трябва да очаква аргумент – номер на служител и да връща брой години служба. Добавете грешка, отчитаща невалиден номер на служител. Използвайте функции MONTHS\_BETWEEN и ROUND.

|  |
| --- |
| CREATE OR REPLACE FUNCTION GET\_YEARS\_SERVICE(input\_employee\_id IN NUMBER)  RETURN NUMBER  IS  var\_years\_of\_service NUMBER;  BEGIN  SELECT *round*(*months\_between*(*sysdate*, HIRE\_DATE) / 12.0)  INTO var\_years\_of\_service  FROM EMPLOYEES  WHERE EMPLOYEE\_ID = input\_employee\_id;  RETURN var\_years\_of\_service;  EXCEPTION  WHEN no\_data\_found THEN  DBMS\_OUTPUT.PUT\_LINE('Employee with id ' || input\_employee\_id || ' does not exist!');  RETURN -1;  END; |

Код за тестването на функцията

|  |
| --- |
| DECLARE  var\_employee\_id NUMBER := &emp\_id;  var\_years\_of\_service NUMBER := GET\_YEARS\_SERVICE(var\_employee\_id);  BEGIN  IF var\_years\_of\_service <> -1 THEN  DBMS\_OUTPUT.PUT\_LINE('Years of service of employee number ' || var\_employee\_id || ': ' || var\_years\_of\_service);  END IF;  END ; |

* Тест със съществуващо ID - 100



* Тест със несъществуващо ID - 300



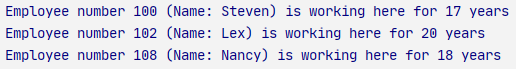
###### 2. Създайте PL/SQL блок за извличане на данни за служители от таблица employees с EMPLOYEE\_ID между 100 и 108. Със стойностите, извлечени от таблица employees, попълнете PL/SQL таблица със стаж (брой години служба) по-голям от15. Изведете информация от PL/SQL таблицата. Използвайте функцията GET\_YEARS\_SERVICE.

|  |
| --- |
| DECLARE  TYPE EMPLOYEES\_TABLE\_TYPE IS TABLE OF employees%ROWTYPE INDEX BY BINARY\_INTEGER;  employees\_table EMPLOYEES\_TABLE\_TYPE;  var\_employee\_row employees%ROWTYPE;  var\_employees\_table\_indexes NUMBER := 0;  BEGIN  FOR counter IN 100..108  LOOP  SELECT *\**  INTO var\_employee\_row  FROM EMPLOYEES  WHERE EMPLOYEE\_ID = counter;  -- for debugging --  DBMS\_OUTPUT.PUT\_LINE(var\_employee\_row.HIRE\_DATE);  DBMS\_OUTPUT.PUT\_LINE(GET\_YEARS\_SERVICE(var\_employee\_row.EMPLOYEE\_ID));  -------------------  IF GET\_YEARS\_SERVICE(var\_employee\_row.EMPLOYEE\_ID) > 15 THEN  var\_employees\_table\_indexes := var\_employees\_table\_indexes + 1;  employees\_table(var\_employees\_table\_indexes) := var\_employee\_row;  END IF;  END LOOP;  FOR counter IN 1..var\_employees\_table\_indexes  LOOP  DBMS\_OUTPUT.PUT\_LINE('Employee number ' || employees\_table(counter).EMPLOYEE\_ID ||  ' (Name: ' || employees\_table(counter).FIRST\_NAME || ')' ||  ' is working here for ' || GET\_YEARS\_SERVICE(employees\_table(counter).EMPLOYEE\_ID) ||  ' years');  END LOOP;  END; |

* Проследяващи логове



* Резултат



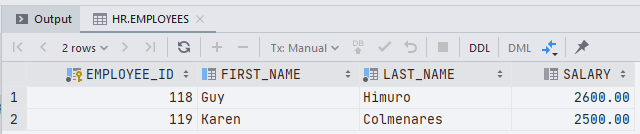
###### 3. Напишете анонимен PL/SQL блок с параметричен курсор (параметър - номер на отдел) за номер на служител, номер на отдел за служители със заплата по-малка от 2700 от таблица emp (копие на таблица employees). Като стойност на параметъра използвайте отдел 30 и променете номера на отдела с нов номер 20 и увеличете заплатите с 5%.

|  |
| --- |
| BEGIN  DROP\_TABLE\_IF\_EXISTS('emp');  END;  CREATE TABLE emp AS (SELECT *\**  FROM EMPLOYEES);  DECLARE  CURSOR cursor\_c (param\_department\_id emp.department\_id%TYPE) IS  SELECT EMPLOYEE\_ID, DEPARTMENT\_ID  FROM emp  WHERE DEPARTMENT\_ID = param\_department\_id  AND SALARY < 2700  FOR UPDATE OF SALARY, DEPARTMENT\_ID;  var\_result\_row cursor\_c%ROWTYPE;  BEGIN  OPEN cursor\_c(30);  LOOP  FETCH cursor\_c INTO var\_result\_row;  EXIT WHEN cursor\_c%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE(var\_result\_row.EMPLOYEE\_ID);  UPDATE emp  SET SALARY = SALARY \* 1.05,  DEPARTMENT\_ID = 20  WHERE CURRENT OF cursor\_c;  END LOOP;  CLOSE cursor\_c;  COMMIT;  END; |

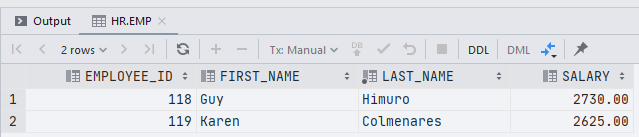
Променени са заплатите на служители с ID 118 и 119:



* HR.EMPLOYEES



* HR.EMP



###### 4. Създайте съхранена процедура ADD\_DEPARTMENT за добавяне на ред в таблица DEPT (копие на DEPARTMENTS), която да има 4 параметъра. Първият параметър е за department\_id, а следващите са за department\_title, локацията и номера на началника на отдела. Да се прави проверка за наличието на такъв номер на отдел, за наличието на такова име на отдел, за наличието на локацията в таблица locations и наличието на номер на служител в таблица employees както и служителят да не е началник на друг отдел. Ако изискванията не са изпълнени, да се вдигнат съответни изключения и за тези данни да се създаде запис в помощна таблица ERROR\_DEPART със структурата на таблица DEPARTMENTS.

а) създаване на таблица DEPT и ERROR\_DEPART

|  |
| --- |
| BEGIN  DROP\_TABLE\_IF\_EXISTS('DEPT');  END;  CREATE TABLE DEPT AS (SELECT *\**  FROM DEPARTMENTS);  BEGIN  DROP\_TABLE\_IF\_EXISTS('ERROR\_DEPART');  END;  CREATE TABLE ERROR\_DEPART AS (SELECT *\**  FROM DEPARTMENTS  WHERE 1 = 2); |

б) създаване на функциите за проверка на наличност на записи в отделните таблици

|  |
| --- |
| CREATE OR REPLACE FUNCTION DEPARTMENT\_EXISTS(input\_department\_id IN DEPT.DEPARTMENT\_ID%TYPE)  RETURN BOOLEAN  IS  var\_department\_id DEPT.DEPARTMENT\_ID%TYPE;  BEGIN  SELECT DEPARTMENT\_ID  INTO var\_department\_id  FROM DEPT  WHERE DEPARTMENT\_ID = input\_department\_id;  RETURN TRUE;  EXCEPTION  WHEN no\_data\_found THEN  RETURN FALSE;  END;  CREATE OR REPLACE FUNCTION IS\_DEPARTMENT\_NAME\_AVAILABLE(input\_department\_name IN DEPT.DEPARTMENT\_NAME%TYPE)  RETURN BOOLEAN  IS  var\_department\_name DEPT.DEPARTMENT\_NAME%TYPE;  BEGIN  SELECT DISTINCT DEPARTMENT\_NAME  INTO var\_department\_name  FROM DEPT  WHERE DEPARTMENT\_NAME = input\_department\_name;  RETURN FALSE;  EXCEPTION  WHEN no\_data\_found THEN  RETURN TRUE;  END;  CREATE OR REPLACE FUNCTION LOCATION\_EXISTS(input\_location\_id IN LOCATIONS.LOCATION\_ID%TYPE)  RETURN BOOLEAN  IS  var\_location\_id LOCATIONS.LOCATION\_ID%TYPE;  BEGIN  SELECT LOCATION\_ID  INTO var\_location\_id  FROM LOCATIONS  WHERE LOCATION\_ID = input\_location\_id;  RETURN TRUE;  EXCEPTION  WHEN no\_data\_found THEN  RETURN FALSE;  END;  CREATE OR REPLACE FUNCTION EMPLOYEE\_EXISTS(input\_employee\_id IN EMPLOYEES.EMPLOYEE\_ID%TYPE)  RETURN BOOLEAN  IS  var\_employee\_id EMPLOYEES.EMPLOYEE\_ID%TYPE;  BEGIN  SELECT EMPLOYEE\_ID  INTO var\_employee\_id  FROM EMPLOYEES  WHERE EMPLOYEE\_ID = input\_employee\_id;  RETURN TRUE;  EXCEPTION  WHEN no\_data\_found THEN  RETURN FALSE;  END;  CREATE OR REPLACE FUNCTION IS\_EMPLOYEE\_MANAGER(input\_employee\_id IN EMPLOYEES.EMPLOYEE\_ID%TYPE)  RETURN BOOLEAN  IS  var\_employee\_id EMPLOYEES.EMPLOYEE\_ID%TYPE;  BEGIN  SELECT DISTINCT MANAGER\_ID  INTO var\_employee\_id  FROM EMPLOYEES  WHERE MANAGER\_ID = input\_employee\_id;  RETURN TRUE;  EXCEPTION  WHEN no\_data\_found THEN  RETURN FALSE;  END; |

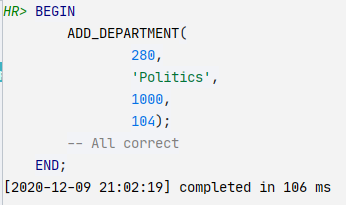
в) създаване на процедурата ADD\_DEPARTMENT

|  |
| --- |
| CREATE OR REPLACE PROCEDURE ADD\_DEPARTMENT(  param\_department\_id IN DEPT.department\_id%TYPE,  param\_department\_title IN DEPT.department\_name%TYPE,  param\_location\_id IN DEPT.location\_id%TYPE,  param\_manager\_id IN DEPT.manager\_id%TYPE)  IS  department\_exists\_exc EXCEPTION;  department\_name\_not\_avail\_exc EXCEPTION;  location\_does\_not\_exist\_exc EXCEPTION;  employee\_does\_not\_exist\_exc EXCEPTION;  employee\_is\_manager\_exc EXCEPTION;  PRAGMA EXCEPTION\_INIT ( department\_exists\_exc, -20001 );  PRAGMA EXCEPTION\_INIT ( department\_name\_not\_avail\_exc, -20002 );  PRAGMA EXCEPTION\_INIT ( location\_does\_not\_exist\_exc, -20003 );  PRAGMA EXCEPTION\_INIT ( employee\_does\_not\_exist\_exc, -20004 );  PRAGMA EXCEPTION\_INIT ( employee\_is\_manager\_exc, -20005 );  BEGIN  IF DEPARTMENT\_EXISTS(param\_department\_id)  THEN  *raise\_application\_error*(-20001, 'Duplicate of PK (Department\_Id) ' || param\_department\_id);  END IF;  IF IS\_DEPARTMENT\_NAME\_AVAILABLE(param\_department\_title) = FALSE  THEN  *raise\_application\_error*(-20002, 'Duplicate of unique column (Department\_Name) ' || param\_department\_title);  END IF;  IF LOCATION\_EXISTS(param\_location\_id) = FALSE  THEN  *raise\_application\_error*(-20003, 'No corresponding FK exist (Location\_Id) ' || param\_location\_id);  END IF;  IF EMPLOYEE\_EXISTS(param\_manager\_id) = FALSE  THEN  *raise\_application\_error*(-20004, 'No corresponding FK exist (Employee\_Id) ' || param\_manager\_id);  END IF;  IF IS\_EMPLOYEE\_MANAGER(param\_manager\_id)  THEN  *raise\_application\_error*(-20005, 'Employee is already manager (Employee\_Id) ' || param\_manager\_id);  END IF;  INSERT INTO  DEPT (department\_id, department\_name, manager\_id, location\_id)  VALUES (param\_department\_id, param\_department\_title, param\_manager\_id, param\_location\_id);  COMMIT;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE(*sqlerrm*);  INSERT INTO  ERROR\_DEPART(department\_id, department\_name, manager\_id, location\_id)  VALUES (param\_department\_id, param\_department\_title, param\_manager\_id, param\_location\_id);  COMMIT;  END; |

г) тестване на процедурата ADD\_DEPARTMENT

* процедурата трябва да работи както е очаквано

|  |
| --- |
| BEGIN  ADD\_DEPARTMENT(  280,  'Politics',  1000,  104);  -- All correct  END; |



* Процедурата трябва да добавя запис в ERROR\_DEPART, с вдигане на съобщение за грешка при дублиран primary key (дублаж на ID на съществуващ отдел)

|  |
| --- |
| BEGIN  ADD\_DEPARTMENT(  100,  'Politics2',  1000,  105);  -- Existing department id  END; |



* Процедурата трябва да добавя запис в ERROR\_DEPART, с вдигане на съобщение за грешка при дублирана уникална колона (дублаж на име на съществуващ отдел)

|  |
| --- |
| BEGIN  ADD\_DEPARTMENT(  290,  'Executive',  1000,  106);  -- Duplicate name  END; |



* Процедурата трябва да добавя запис в ERROR\_DEPART, с вдигане на съобщение за грешка липса на съответстващ foreign key (несъществуващо местоположение)

|  |
| --- |
| BEGIN  ADD\_DEPARTMENT(  300,  'Politics3',  3300,  107);  -- Not existing location  END; |



* Процедурата трябва да добавя запис в ERROR\_DEPART, с вдигане на съобщение за грешка липса на съответстващ foreign key (несъществуващ служител)

|  |
| --- |
| BEGIN  ADD\_DEPARTMENT(  310,  'Politics4',  1000,  207);  -- Not existing employee  END; |



* Процедурата трябва да добавя запис в ERROR\_DEPART, с вдигане на съобщение за грешка нарушение на логиката (служителят е вече мениджър на друг отдел)

|  |
| --- |
| BEGIN  ADD\_DEPARTMENT(  320,  'Politics5',  1000,  100);  -- Employee is manager  END; |



Резултатите в таблица ERROR\_DEPART след изпълнението на всички тестове

