**Buổi 3 Ôn lại tạo bảng**

Trasua(masp,tensp,mota,gia)

Khachhang(tenhk,sdt,diachi)

Dondat(sttdt,sodt,ngaydat,tongtien)

Chitietdat(sttdt,masp,soluong,thanhtien)

Điều kiện: gia >10000, tongtien >0, soluong >0, thanhtien>0

Create table TRASUA(

Masp varchar (10) primary key,

Tensp varchar2 (30) not null,

Mota vachar2(40) not null,

Gia float check(gia>10000)

);

Create table KHACHHANG(

Tenkh varchar2 (30) not null,

Sdt varchar (15) primary key,

Diachi varchar (30) not null

);

Create table DONDAT(

Sttdt varchar (10) primary key,

Sodt varchar (15) constraint pk\_Sodt refernces KHACHHANG(sodt),

Ngaydat date default sysdate,

Tongtien float check(tongtien > 0)

);

ALTER SESSION SET NLS\_DATE\_FORMAT='DD-MM-YYYY';

CREATE TABLE DONDAT (

Sttdt VARCHAR(10) PRIMARY KEY,

Sodt VARCHAR(15) REFERENCES KHACHHANG(sdt),

Ngaydat DATE DEFAULT CURRENT\_DATE,

Tongtien FLOAT CHECK (Tongtien > 0)

);

Create table chitietdat(

Sttdt varchar (10) not null,

Masp varchar (10) not null,

Soluong int check(soluong>0),

Thanhtien float check(thanhtien>0),

Primary key (Masp, Sttdt),

Constraint pk\_sttdt foreign key (sttdt) references DONDAT(sttdt),

Constraint pk\_masp foreign key (masp) references TRASUA(masp)

);

CREATE TABLE Chitietdat (

Sttdt VARCHAR(10) NOT NULL,

Masp VARCHAR(10) NOT NULL,

Soluong INT CHECK (Soluong > 0),

Thanhtien FLOAT CHECK (Thanhtien > 0),

PRIMARY KEY (Masp, Sttdt),

CONSTRAINT fk\_sttdt FOREIGN KEY (Sttdt) REFERENCES DONDAT(Sttdt),

CONSTRAINT fk\_masp FOREIGN KEY (Masp) REFERENCES TRASUA(Masp)

);

INSERT INTO TRASUA VALUES (1, ‘Tra sua tran chau duong den’, ‘Tra sua ngon’, 25000),

INSERT INTO TRASUA VALUES (2, ‘Tra sua thai’, ‘Thom ngon’, 30000),

INSERT INTO TRASUA VALUES (3, ‘Tra sua hoa hong’, ‘Thom lung’, 28000)

INSERT INTO KHACHHANG VALUES (‘Nguyen Van A’ , ‘0123456789’, ‘Can Tho’),

INSERT INTO KHACHHANG VALUES (‘NguyenThi B’ , ‘0123498745’, ‘TP.HCM’),

INSERT INTO KHACHHANG VALUES (‘Nguyen Van C’ , ‘0987445689’, ‘Ha Noi’)

INSERT INTO DONDAT VALUES (‘D01, ‘0123456789’, ‘02-03-2024’, 75000),

INSERT INTO DONDAT VALUES (‘D02, ‘0123498745’, ‘01-03-2024’, 84000),

INSERT INTO DONDAT VALUES (‘D03, ‘0987445689’, ‘29-02-2024’, 30000)

INSERT INTO CHITIETDAT VALUES (‘D01, 1, 2, 50000),

INSERT INTO CHITIETDAT VALUES (‘D02, 2, 1, 30000),

INSERT INTO CHITIETDAT VALUES (‘D03,2, 2, 56000)

\*\*MỘT SỐ LỆNH CƠ BẢN KHÁC

**- -Xóa bảng**:

**DROP TABLE TenBang;**

**DROP TABLE Trasua;**

**- Thêm 1 cột vào bảng:**

**ALTER TABLE TenBang**

**ADD TenCot KieuDuLieu;**

**ALTER TABLE Trasua**

**ADD ngaycapnhat DATE;**

**- Xóa cột trong bảng:**

**ALTER TABLE TenBang**

**DROP COLUMN TenCot;**

**ALTER TABLE Trasua**

**DROP COLUMN ngaycapnhat;**

- Thêm ràng buộc khóa chính:

**ALTER TABLE TenBang**

**ADD CONSTRAINT ten\_rangbuoc PRIMARY KEY (TenCot);**

- Xóa ràng buộc khóa chính:

**ALTER TABLE TenBang**

**DROP PRIMARY KEY;**

- Thêm ràng buộc khóa ngoại:

**ALTER TABLE TenBang**

**ADD CONSTRAINT ten\_rangbuoc FOREIGN KEY (TenCot) REFERENCES TenBangKhac(TenCot);**

- Xóa ràng buộc khóa ngoại:

**ALTER TABLE TenBang**

**DROP FOREIGN KEY ten\_rangbuoc;**

- Thêm hoặc xóa ràng buộc khác như ràng buộc kiểm tra (CHECK CONSTRAINT), ràng buộc duy nhất (UNIQUE CONSTRAINT),...:

**ALTER TABLE TenBang**

**ADD CONSTRAINT ten\_rangbuoc CHECK (DieuKien);**

**Hoặc**

**ALTER TABLE TenBang**

**DROP CONSTRAINT ten\_rangbuoc;**

ADD CONSTRAINT check\_age CHECK (age >= 18) COMMENT

check\_age **là tên ràng buộc**

**BUỔI 4 TẠO NGƯỜI DÙNG**

--cau 5

--tạo người dùng Test1, mk giống tên

--tạo user

CREATE USER Test1 IDENTIFIED BY Test1;

--cấp quyền Create Session cho Test1 (quyền cho phép user kết nối vào CSDL)

GRANT CREATE SESSION TO Test1;

--thay đổi mk cho user này thử xem

AFTER USER Test1 IDENTIFIED BYpTest1;

--cấp quyền hệ thống cho user này để anh ta có thể tạo bảng, tạo user, tạo session và được phép cấp lại các quyền này cho người khác.

GRANT CREATE TABLE, CREATE USER, CREATE SESION TO Test1 WITH ADMIN OPTION;

--Cấp quota cho người dùng này;

AFTER USER Test1 default tablespace users quota 2M on users;

--đang nhập với user vừa tạo để thử tạo một bảng, nhập và truy xuất dữ liệu.

CREATE TABLE STUDENT(

SID varchar2(7) primary key,

SName varchar (30)

);

--them dl cho bảng

INSERT INTO STUDENT(‘B1234567’,’NGUYEN VAN A’);

INSERT INTO STUDENT(‘B1288567’,’NGUYEN VAN B’);

Select \* From STUDENT;

--CAU 6 QUYỀN HỆ THỐNG

--để kt quyền CREATE SESION TO Test1 WITH ADMIN OPTION; của test1, test1 tạo ra tets2

CREATE USER Test2 IDENTIFIED Test2

DEFAULT TABLESPACE USERS QUOTA 2M ON USERS;

--gán quyền create sisson cho test2

GRANT CREATE SESSION TO Test2;

--cấp quyền hệ thống cho user Test2 để nó tạo bảng và cấp quyền lại cho người khác

GRANT CREATE TABLE TO Test WITH ADMIN OPTION;

--đang nhập bằng tài khoản sys để thu hồi quyền CREATE TABLE của người dùng Test1

REVOKE CREATE TABLE FROM Test1;

--tạo user Test3 các quyền giống với test2

CREATE USER Test3 IDENTIFIED Test3

DEFAULT TABLESPACE USERS QUOTA 2M ON USERS;

GRANT CREATE SESSION TO Test3;

GRANT CREATE TABLE TO Test WITH ADMIN OPTION;

--CÂU 7 QUYỀN ĐỐI TƯỢNG

--đăng nhập bằng người dùng sys tạo ra 2 người dùng

CREATE USER user1 IDENTIFIED user1

DEFAULT TABLESPACE USERS QUOTA 1M IN USERS;

CREATE USER user2 IDENTIFIED user2

DEFAULT TABLESPACE USERS QUOTA 1M IN USERS;

--Tạo role có tên TToracle

CREARE ROLE TToracle;

--cấp quyền kết nối CSDL, TẠO BẢNG,tạo thủ tục, tạo trigger cho role TTOracle

GRANT CREATE SESION, CREATE TABLE. CREATE PROCEDURE, CREATE TRIGGER TO TTOracle;

--cấp quyền cho các SV vào role này(đua các user vào role)

GRANT TTOcracle TO user1, user2;

--thu hồi role

REVOKE TTOracle FROM user1;

**--test1**

CREATE TABLE STUDENT(

SID char (7) primary key,

SName varchar (30)

);

INSERT INTO STUDENT(‘B1234567’,’NGUYEN VAN A’);

INSERT INTO STUDENT(‘B1288567’,’NGUYEN VAN B’);

Select \* From STUDENT;

CREATE TABLE Teachers (

TID char(7) PRIMARY KEY,

TName Varchar2(30)

);--Khong tao duoc bang do bi thu hoi quyen

--Cap quyen Select, Update cho Test2

GRANT SELECT, UPDATE ON Students TO Test2 WITH GRANT OPTION;

--Thu hoi quyen Select va Update cua Test2

REVOKE SELECT, UPDATE ON Students FROM Test2;

**--test2 tạo bảng procedure**

CREATE TABLE Products (

PID char(3) PRIMARY KEY,

PName Varchar2(30)

);

INSERT INTO Products VALUES('P01', 'Kem Danh Rang P/S');

INSERT INTO Products VALUES('P02', 'Khan Giay Puppy');

SELECT \* FROM PRODUCTS;

CREATE TABLE Teachers (

TID char(7) PRIMARY KEY,

TName Varchar2(30)

);

--cau7

SELECT \* FROM Test1.Students;

DELECT FROM Test1.Students;

--Khong thuc hien duoc do khong co quyen nay

GRANT SELECT, UPDATE ON Test1.Students TO Test3;

GRANT DELETE ON Test1.Students TO Test3;

**--Test3**

UPDATE Test1.Students SET sname='bla bla bla' WHERE SID='1940001';

SELECT \* FROM Test1.students;

--user1

CREATE TABLE user1table (

U1ID char(10) PRIMARY KEY,

U1Name Varchar(30)

);

--user2

CREATE TABLE user2table (

U2ID char(10) PRIMARY KEY,

U2Name Varchar(30)

);

**Buổi 5**

--Cau 1 khai báo biến, gán giá trị và in chúng ra màn hình

SET SERVEROUTPUT ON;

DECLARE

total\_sales NUMBER(15,2);

emp\_id VARCHAR2(9);

COMPANY\_NUMBER NUMBER DEFAULT 10;

BEGIN

total\_sales := 350000;

emp\_id := 3;

dbms\_output.put\_line('Employee ' || emp\_id || ', sold total product value: ' || total\_sales);

END;

1. **SET SERVEROUTPUT ON;**: Dòng này được sử dụng để bật chức năng hiển thị kết quả ra màn hình trong quá trình thực thi PL/SQL.
2. **DECLARE**: Bắt đầu một phần khai báo biến và khối mã PL/SQL.
3. **total\_sales NUMBER(15,2**);: Khai báo biến total\_sales là một số có tổng cộng 15 chữ số và 2 chữ số sau dấu thập phân, dùng để lưu trữ tổng doanh số bán hàng.
4. **emp\_id VARCHAR2(9);**: Khai báo biến emp\_id là một chuỗi có độ dài tối đa 9 ký tự, dùng để lưu trữ mã nhân viên.
5. **COMPANY\_NUMBER NUMBER DEFAULT 10;**: Khai báo biến COMPANY\_NUMBER là một số nguyên với giá trị mặc định là 10.
6. **BEGIN**: Bắt đầu khối mã chính.
7. **total\_sales := 350000;**: Gán giá trị 350000 cho biến total\_sales, đại diện cho tổng doanh số bán hàng của nhân viên.
8. **emp\_id := 3;**: Gán giá trị 3 cho biến emp\_id, đại diện cho mã nhân viên.
9. **dbms\_output.put\_line('Employee ' || emp\_id || ', sold total product value: ' || total\_sales);**: In ra màn hình thông tin về doanh số bán hàng của nhân viên, bao gồm mã nhân viên và tổng giá trị sản phẩm đã bán.
10. Về mặt cụ thể, đoạn mã này sẽ hiển thị thông tin "Nhân viên [mã nhân viên], đã bán tổng giá trị sản phẩm: [tổng doanh số bán hàng]" ra màn hình, trong trường hợp này là "Nhân viên 3, đã bán tổng giá trị sản phẩm: 350000". Tuy nhiên, có một vấn đề nhỏ ở đây là biến emp\_id được khai báo là VARCHAR2 nhưng lại được gán giá trị số nguyên, điều này có thể gây ra lỗi hoặc kết quả không mong muốn. Để sửa lỗi này, bạn nên khai báo emp\_id là một số nguyên thích hợp (ví dụ: NUMBER).

--Cau 2

SET SERVEROUTPUT ON;

DECLARE

ten VARCHAR2(10);

BEGIN

ten := &Nhap\_vao\_ten\_cua\_ban;

dbms\_output.put\_line('chao ban ' || ten);

END;

--Cau 3

SET SERVEROUTPUT ON;

DECLARE

vEname hr.employees.last\_name%TYPE;

vSalary hr.employees.salary%TYPE;

BEGIN

SELECT last\_name, salary INTO vEname, vSalary

FROM hr.employees

WHERE employee\_id = 100;

dbms\_output.put\_line('Name: ' || vEname || ', Salary: ' || vSalary);

END;

1. **SET SERVEROUTPUT ON;**: Dòng này bật chức năng hiển thị kết quả ra màn hình trong quá trình thực thi PL/SQL.
2. **DECLARE**: Khối khai báo biến và khối mã PL/SQL bắt đầu từ đây.
3. **ten VARCHAR2(10);**: Biến ten được khai báo là một chuỗi có độ dài tối đa là 10 ký tự, được sử dụng để lưu trữ tên của người dùng.
4. **BEGIN**: Bắt đầu khối mã chính.
5. **ten := &Nhap\_vao\_ten\_cua\_ban;**: Ở đây, dấu & được sử dụng để yêu cầu nhập giá trị từ bàn phím. Người dùng sẽ nhập tên của họ, và giá trị này sẽ được gán cho biến **ten**.
6. **dbms\_output.put\_line('chao ban ' || ten);**: Dòng này sử dụng dbms\_output.put\_line để hiển thị một thông điệp chào đón ra màn hình. Tên của người dùng (lưu trong biến ten) được nối vào thông điệp "chao ban".
7. Khi bạn chạy đoạn mã này, hệ thống sẽ yêu cầu bạn nhập tên của mình và sau đó sẽ hiển thị thông điệp chào đón với tên bạn vừa nhập. Ví dụ, nếu bạn nhập "Alice", thông điệp được hiển thị sẽ là "chao ban Alice".

--Cau 4

SET SERVEROUTPUT ON;

DECLARE

vEname hr.employees.first\_name%TYPE;

BEGIN

SELECT first\_name INTO vEname

FROM hr.employees

WHERE employee\_id = 120;

IF vEname='Matthew' THEN

dbms\_output.put\_line('Hi ' || vEname);

ELSE

dbms\_output.put\_line('Hello ' || vEname);

END IF;

END;

SET SERVEROUTPUT ON;

DECLARE

vArea VARCHAR2(20);

BEGIN

SELECT region\_id INTO vArea

FROM hr.countries

WHERE country\_id = 'CA';

CASE vArea

WHEN 1 THEN vArea:='Europe';

WHEN 2 THEN vArea:='America';

WHEN 3 THEN vArea:='Asia';

ELSE vArea:='Other';

END CASE;

dbms\_output.put\_line('The area is ' || vArea);

END;

SET SERVEROUTPUT ON;

DECLARE

counter NUMBER;

BEGIN

FOR counter IN 1..4

LOOP

dbms\_output.put(counter);

END LOOP;

dbms\_output.new\_line;

FOR counter IN REVERSE 1..4

LOOP

dbms\_output.put(counter);

END LOOP;

dbms\_output.new\_line;

END;

Dòng **vEname hr.employees.first\_name%TYPE;** trong khối khai báo biến của PL/SQL có ý nghĩa như sau:

**hr.employees.first\_name%TYPE**: Đây là một phần của SQLPlus, một công cụ quản lý cơ sở dữ liệu Oracle. Trong một câu lệnh SQLPlus, **%TYPE** là một từ khóa đặc biệt được sử dụng để xác định kiểu dữ liệu của một cột trong một bảng.

**hr.employees.first\_name**: Đây là cột first\_name trong bảng employees của schema hr.

**vEname:** Đây là tên của biến mà chúng ta muốn khai báo.

Vì **%TYPE** được sử dụng, kiểu dữ liệu của biến vEname sẽ giống với kiểu dữ liệu của cột **first\_name** trong bảng employees. Điều này giúp đảm bảo rằng biến **vEname** sẽ luôn chứa dữ liệu phù hợp khi trích xuất từ cột **first\_name**. Nó cũng là một cách tốt để tránh việc phải cập nhật kiểu dữ liệu của biến nếu kiểu dữ liệu của cột **first\_name** thay đổi trong tương lai.

Tóm lại, câu lệnh **SELECT** này được sử dụng để lấy tên của nhân viên có **employee\_id** là 120 từ bảng employees trong **schema hr**, và sau đó gán tên đó vào biến **vEname**.

Tóm lại, đoạn mã này kiểm tra giá trị của biến vEname. Nếu vEname bằng 'Matthew', nó sẽ hiển thị thông điệp "Hi" cùng với tên 'Matthew', ngược lại nó sẽ hiển thị thông điệp "Hello" cùng với giá trị của vEname.

SET SERVEROUTPUT ON;: Dòng này bật chức năng hiển thị kết quả ra màn hình trong quá trình thực thi PL/SQL.

DECLARE: Bắt đầu phần khai báo biến và khối mã PL/SQL.

vArea VARCHAR2(20);: Khai báo biến vArea là một chuỗi có độ dài tối đa là 20 ký tự, dùng để lưu trữ khu vực (region).

1. BEGIN: Bắt đầu khối mã chính.

SELECT region\_id INTO vArea FROM hr.countries WHERE country\_id = 'CA';: Sử dụng câu lệnh SELECT để lấy giá trị của cột region\_id từ bảng countries trong schema hr, nơi country\_id là 'CA' (Canada). Kết quả được gán cho biến vArea.

CASE vArea ... END CASE;: Sử dụng câu lệnh CASE để xác định khu vực tương ứng với giá trị của biến vArea. Nếu vArea có giá trị là 1, sẽ gán vArea bằng 'Europe', nếu là 2 sẽ gán 'America', nếu là 3 sẽ gán 'Asia', và nếu không phù hợp với bất kỳ giá trị nào trong số này thì sẽ gán 'Other'.

dbms\_output.put\_line('The area is ' || vArea);: In ra màn hình thông điệp "The area is " kèm theo giá trị của biến vArea, thông qua hàm dbms\_output.put\_line.

END;: Kết thúc khối mã PL/SQL.

Tóm lại, đoạn mã này lấy thông tin về khu vực của quốc gia có mã 'CA' và sau đó hiển thị khu vực đó ra màn hình.

1. SET SERVEROUTPUT ON;: Dòng này bật chức năng hiển thị kết quả ra màn hình trong quá trình thực thi PL/SQL.
2. DECLARE: Bắt đầu phần khai báo biến và khối mã PL/SQL.
3. counter NUMBER;: Biến counter được khai báo là một số nguyên.
4. BEGIN: Bắt đầu khối mã chính.
5. Vòng lặp đầu tiên:
6. FOR counter IN 1..4 LOOP: Vòng lặp này sẽ lặp qua các giá trị từ 1 đến 4. Biến counter sẽ lấy giá trị từ 1 đến 4 trong mỗi lần lặp.
7. dbms\_output.put(counter);: Trong mỗi lần lặp, số được lưu trong biến counter sẽ được in ra màn hình bằng cách sử dụng hàm dbms\_output.put.
8. dbms\_output.new\_line;: Sau khi hoàn thành vòng lặp, dòng này tạo ra một dòng mới trên màn hình.
9. Vòng lặp thứ hai:
10. FOR counter IN REVERSE 1..4 LOOP: Vòng lặp này sẽ lặp qua các giá trị từ 1 đến 4 theo thứ tự ngược lại (từ 4 đến 1). Điều này được thực hiện bằng cách sử dụng REVERSE.
11. dbms\_output.put(counter);: Trong mỗi lần lặp, số được lưu trong biến counter sẽ được in ra màn hình.
12. dbms\_output.new\_line;: Sau khi hoàn thành vòng lặp, dòng này tạo ra một dòng mới trên màn hình.
13. END;: Kết thúc khối mã PL/SQL.

Buổi 6

--Cau 5:

--a.

--Cach 1: Dung FETCH

SET SERVEROUTPUT ON;

DECLARE

CURSOR c\_Emp IS

select \*

from SCOTT.emp

where extract(year from sysdate) - extract(year from hiredate)> 25

and sal < 2000;

v\_Emp c\_Emp%rowtype;

BEGIN

OPEN c\_Emp;

LOOP

FETCH c\_Emp INTO v\_Emp;

EXIT WHEN c\_Emp%notfound;

IF v\_Emp.comm is null THEN

v\_Emp.comm:=500;

ELSE

v\_Emp.comm:=v\_Emp.comm+500;

END IF;

UPDATE SCOTT.Emp

SET comm = v\_Emp.comm

WHERE empno = v\_Emp.empno;

INSERT INTO SCOTT.BONUS(ename, job, sal, comm)

VALUES(v\_Emp.ename, v\_Emp.job, v\_Emp.sal, v\_Emp.comm);

END LOOP;

COMMIT;

CLOSE c\_Emp;

END;

- Dòng **CURSOR c\_Emp IS** trong mã là phần khai báo một CURSOR, trong đó:

CURSOR: Là từ khóa để bắt đầu khai báo một CURSOR trong PL/SQL.

c\_Emp: Là tên của CURSOR, trong trường hợp này, CURSOR có tên là c\_Emp. Tên này có thể được sử dụng sau đó để tham chiếu đến CURSOR trong các phần khác của mã PL/SQL.

IS: Là từ khóa được sử dụng để chỉ ra rằng đang khai báo một CURSOR và đang bắt đầu phần định nghĩa của CURSOR.

Do đó, dòng CURSOR c\_Emp IS chỉ đơn giản là phần bắt đầu của phần định nghĩa một CURSOR với tên là c\_Emp.

- Trong đoạn mã này, một CURSOR có tên là c\_Emp được khai báo. CURSOR này sẽ lấy dữ liệu từ bảng EMP trong cơ sở dữ liệu SCOTT với các điều kiện sau:

Năm hiện tại trừ đi năm của cột hiredate trong bảng EMP phải lớn hơn 25.

Mức lương (cột sal) của nhân viên trong bảng EMP phải nhỏ hơn 2000.

Những dòng dữ liệu phù hợp với các điều kiện này sẽ được lấy ra và lưu vào CURSOR c\_Emp.

Sau đó, một biến cấu trúc (structure) với tên v\_Emp và kiểu dữ liệu là %rowtype của CURSOR c\_Emp được khai báo. Cú pháp %rowtype cho phép biến có cấu trúc tương tự như một dòng trong CURSOR, nghĩa là nó có cùng cấu trúc dữ liệu với dòng dữ liệu được trả về từ CURSOR.

Tóm lại, CURSOR c\_Emp này được sử dụng để lấy ra tất cả các dòng dữ liệu từ bảng EMP trong cơ sở dữ liệu SCOTT thỏa mãn các điều kiện được chỉ định, và biến v\_Emp được sử dụng để lưu trữ từng dòng dữ liệu này trong quá trình xử lý.

- Dòng OPEN c\_Emp;: Mở CURSOR c\_Emp để bắt đầu truy xuất dữ liệu từ bảng EMP theo các điều kiện đã được chỉ định trong khai báo của CURSOR.

-Vòng lặp LOOP: Bắt đầu một vòng lặp vô hạn, mà sau mỗi vòng lặp sẽ thực hiện một số hành động trên dòng dữ liệu hiện tại được lấy ra từ CURSOR c\_Emp.

-Dòng FETCH c\_Emp INTO v\_Emp;: Lấy dữ liệu từ CURSOR c\_Emp vào biến v\_Emp. Mỗi lần lặp, dòng này sẽ lấy một dòng dữ liệu mới từ CURSOR và gán vào biến v\_Emp.

-Dòng EXIT WHEN c\_Emp%notfound;: Kiểm tra điều kiện c\_Emp%notfound, nếu không còn dòng dữ liệu nào phù hợp với CURSOR c\_Emp thì thoát khỏi vòng lặp.

-Các dòng lệnh bên trong vòng lặp:

-Kiểm tra xem cột comm của dòng dữ liệu hiện tại có giá trị NULL hay không. Nếu có, thì gán giá trị 500 cho cột comm của dòng dữ liệu này, ngược lại, tăng giá trị của cột comm lên 500.

-Cập nhật giá trị của cột comm trong bảng EMP và thực hiện thêm dòng dữ liệu vào bảng BONUS với các giá trị từ dòng dữ liệu hiện tại.

-Dòng COMMIT;: Lưu các thay đổi vào cơ sở dữ liệu. COMMIT sẽ xác nhận rằng tất cả các thay đổi được thực hiện trong phiên làm việc hiện tại được lưu lại.

-Dòng CLOSE c\_Emp;: Đóng CURSOR c\_Emp để giải phóng tài nguyên sau khi đã sử dụng xong.

Tóm lại, đoạn mã này lấy dữ liệu từ CURSOR c\_Emp, thực hiện các thay đổi và thêm dữ liệu vào các bảng, sau đó lưu các thay đổi và đóng CURSOR.

UPDATE SCOTT.Emp

SET comm = v\_Emp.comm

WHERE empno = v\_Emp.empno;

* SCOTT.Emp: Đây là tên bảng cần được cập nhật, trong trường hợp này, đó là bảng Emp trong schema SCOTT.
* SET comm = v\_Emp.comm: Là phần của lệnh UPDATE xác định cột cần cập nhật và giá trị mới mà nó sẽ được cập nhật. Ở đây, cột comm sẽ được cập nhật bằng giá trị của cột comm của dòng dữ liệu hiện tại trong biến v\_Emp.
* WHERE empno = v\_Emp.empno: Điều kiện để xác định dòng dữ liệu cụ thể cần được cập nhật. Trong trường hợp này, nó so sánh cột empno trong bảng Emp với giá trị empno của dòng dữ liệu hiện tại trong biến v\_Emp. Khi tìm thấy dòng có empno phù hợp, thì cột comm trong dòng đó sẽ được cập nhật với giá trị tương ứng từ biến v\_Emp.
* Tóm lại, dòng này thực hiện cập nhật giá trị của cột comm trong bảng Emp để đồng bộ với giá trị của cột comm trong biến v\_Emp.

--Cach 2: Dung FOR

SET SERVEROUTPUT ON;

DECLARE

CURSOR c\_Emp IS

select \*

from SCOTT.emp

where extract(year from sysdate) - extract(year from hiredate)> 25

and sal < 2000;

BEGIN

FOR emp\_rec IN c\_Emp

LOOP

IF emp\_rec.comm is null THEN

emp\_rec.comm:=500;

ELSE

emp\_rec.comm:=emp\_rec.comm+500;

END IF;

UPDATE SCOTT.Emp

SET comm = emp\_rec.comm

WHERE empno = emp\_rec.empno;

INSERT INTO SCOTT.BONUS(ename, job, sal, comm)

VALUES(emp\_rec.ename, emp\_rec.job, emp\_rec.sal, emp\_rec.comm);

END LOOP;

END;

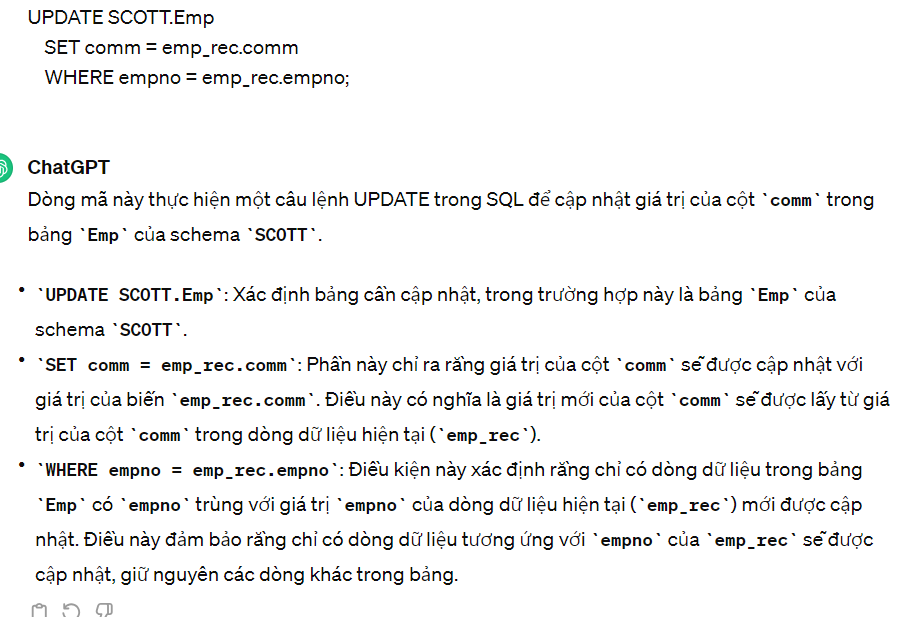
* FOR emp\_rec: Đây là cách khai báo một biến vòng lặp trong PL/SQL. Trong trường hợp này, biến vòng lặp được đặt tên là emp\_rec.
* IN c\_Emp: Đây là phần mô tả về nguồn dữ liệu cho vòng lặp. Trong trường hợp này, nó là một CURSOR được đặt tên là c\_Emp.
* Khi vòng lặp FOR IN được thực hiện, nó lặp qua mỗi dòng dữ liệu được trả về bởi CURSOR c\_Emp, và mỗi lần lặp, biến vòng lặp emp\_rec sẽ chứa các giá trị của dòng dữ liệu hiện tại. Điều này cho phép bạn thực hiện các thao tác xử lý trên mỗi dòng dữ liệu một cách dễ dàng, không cần phải sử dụng FETCH như trong các vòng lặp thông thường sử dụng CURSOR.

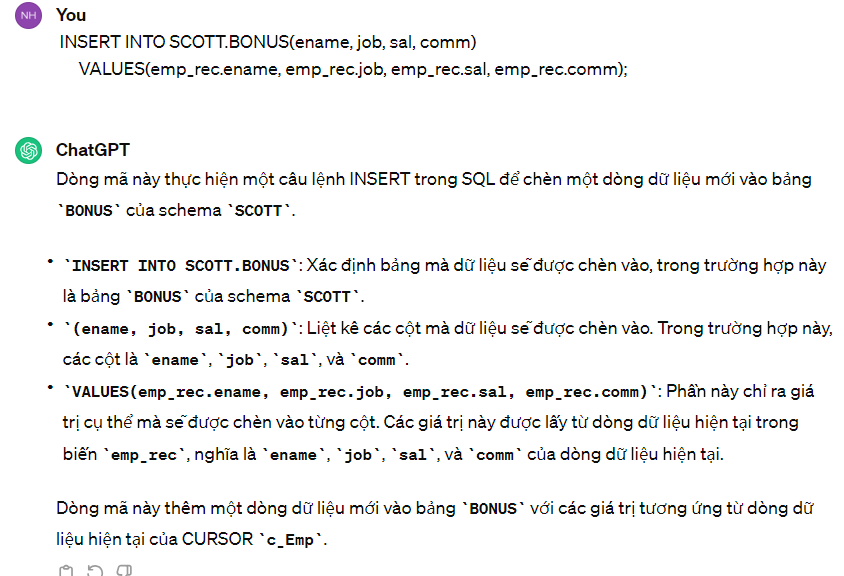
- Đoạn mã này kiểm tra giá trị của cột comm trong dòng dữ liệu hiện tại, được lưu trữ trong biến emp\_rec, và thực hiện các hành động tương ứng:

- Nếu giá trị của cột comm là null (emp\_rec.comm is null), nghĩa là không có giá trị nào đã được gán cho cột comm, thì nó sẽ gán giá trị 500 cho cột comm. Điều này làm cho giá trị của cột comm trở thành 500.

- Nếu giá trị của cột comm không phải là null (ELSE), nghĩa là đã có một giá trị được gán cho cột comm, thì nó sẽ tăng giá trị hiện tại của cột comm lên 500 (emp\_rec.comm + 500). Sau đó, giá trị mới này được gán lại cho cột comm.

- Dòng mã này cung cấp một cách tiếp cận linh hoạt để cập nhật giá trị của cột comm dựa trên giá trị hiện tại của nó, giảm thiểu việc cập nhật cứng nhắc bằng các giá trị cứng.





--b.

CREATE TABLE EMP\_RETIRE (

EMPNO NUMBER(4) NOT NULL,

ENAME VARCHAR2(10),

JOB VARCHAR2(9),

MGR NUMBER(4),

HIREDATE DATE,

SAL NUMBER(7,2),

COMM NUMBER(7,2),

DEPTNO NUMBER(2)

);

--Cach 1: Dung FOR

SET SERVEROUTPUT ON;

DECLARE

CURSOR cEmpRetire IS

select \* from scott.emp

where extract(year from sysdate) - extract(year from hiredate) >= 44 FOR UPDATE;

BEGIN

FOR emp\_rec IN cEmpRetire

LOOP

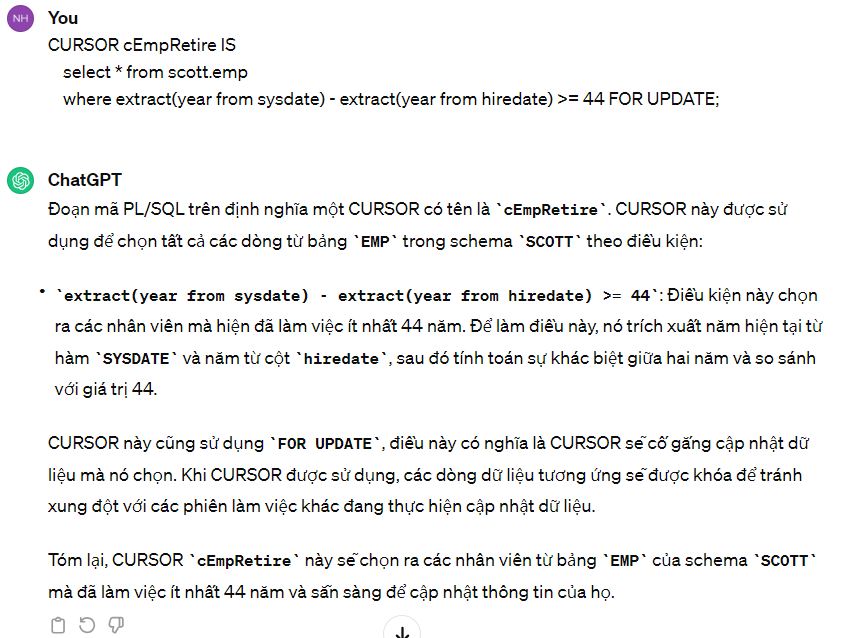
INSERT INTO Emp\_Retire VALUES (emp\_rec.Empno, emp\_rec.Ename, emp\_rec.Job,

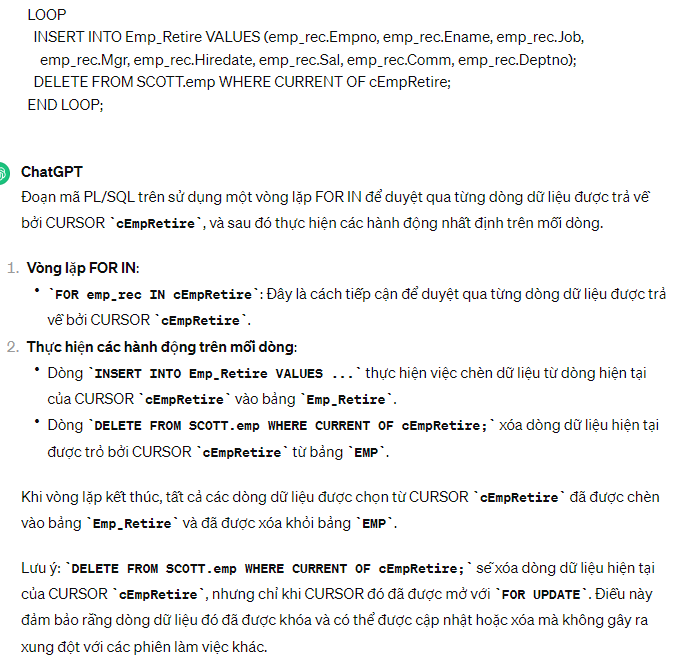
emp\_rec.Mgr, emp\_rec.Hiredate, emp\_rec.Sal, emp\_rec.Comm, emp\_rec.Deptno);

DELETE FROM SCOTT.emp WHERE CURRENT OF cEmpRetire;

END LOOP;

END;





--Cach 2: Dung FETCH

SET SERVEROUTPUT ON;

DECLARE

CURSOR cEmpRetire IS

select \* from scott.emp

where extract(year from sysdate) - extract(year from hiredate) >= 44 FOR UPDATE;

vEmpRetire cEmpRetire%rowtype;

BEGIN

OPEN cEmpRetire;

LOOP

FETCH cEmpRetire INTO vEmpRetire;

EXIT WHEN cEmpRetire%notfound;

INSERT INTO Emp\_Retire VALUES (vEmpRetire.Empno, vEmpRetire.Ename, vEmpRetire.Job,

vEmpRetire.Mgr, vEmpRetire.Hiredate, vEmpRetire.Sal, vEmpRetire.Comm, vEmpRetire.Deptno);

DELETE FROM SCOTT.emp WHERE CURRENT OF cEmpRetire;

END LOOP;

COMMIT;

END;

--c.

--Cach 1: Dung FOR

SET SERVEROUTPUT ON;

DECLARE

manager SCOTT.EMP.MGR%TYPE;

CURSOR C\_Emp(mgr\_no NUMBER) IS

SELECT SAL

FROM SCOTT.EMP

WHERE MGR = mgr\_no

FOR UPDATE of SAL;

BEGIN

SELECT empno INTO manager

FROM SCOTT.EMP

WHERE ename like 'KING';

FOR emp\_rec IN C\_Emp(manager)

LOOP

UPDATE SCOTT.EMP

SET sal = emp\_rec.sal \* 1.05

WHERE CURRENT OF C\_Emp;

END LOOP;

COMMIT;

END;

--Cach 2: Dung FETCH

SET SERVEROUTPUT ON;

DECLARE

manager SCOTT.EMP.MGR%TYPE;

CURSOR C\_Emp(mgr\_no NUMBER) IS

SELECT SAL

FROM SCOTT.EMP

WHERE MGR = mgr\_no

FOR UPDATE of SAL;

V\_Emp C\_Emp%rowtype;

BEGIN

SELECT empno INTO manager

FROM SCOTT.EMP

WHERE ename like 'KING';

OPEN C\_Emp(manager);

LOOP

FETCH C\_Emp INTO V\_Emp;

EXIT WHEN C\_Emp%notfound;

UPDATE SCOTT.EMP

SET sal = V\_Emp.sal \* 1.05

WHERE CURRENT OF C\_Emp;

END LOOP;

COMMIT;

END;

Buổi 7

--Cau 1:

--Tao thu tuc de xoa mot nhan vien khoi danh sach voi tham so truyen vao la ma nhan vien

CREATE OR REPLACE PROCEDURE scott.Del\_Emp(p\_EmpNo NUMBER)

IS

BEGIN

delete from scott.emp where empno = p\_EmpNo;

COMMIT;

END;

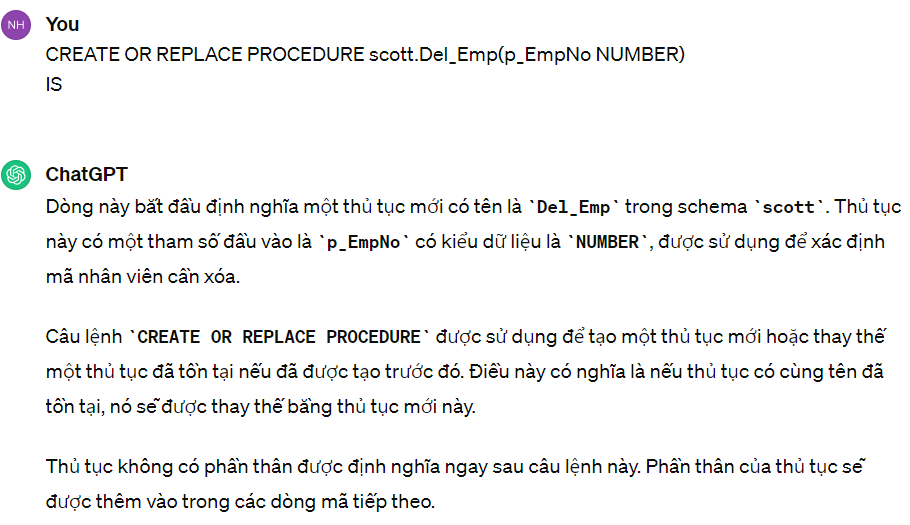
--Tim xem nhan vien co ma 7654 co trong bang khong

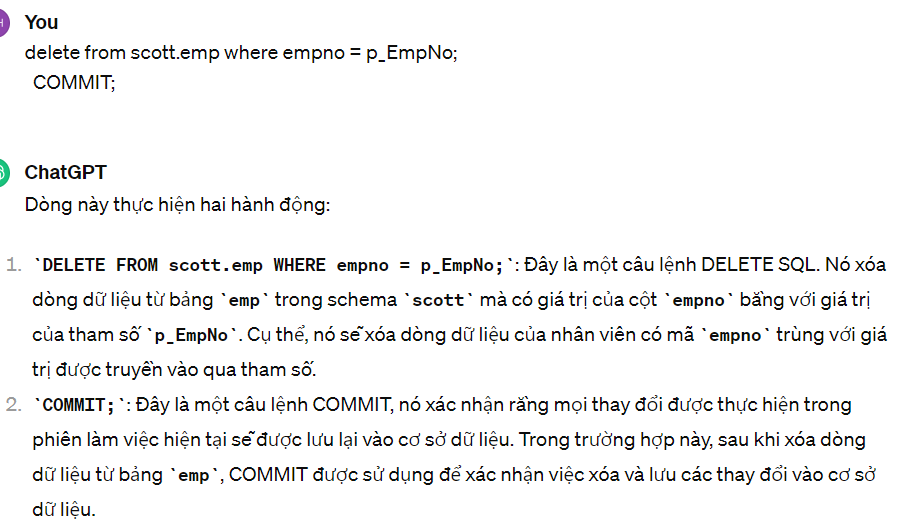
select \* from scott.emp where empno = 7654;

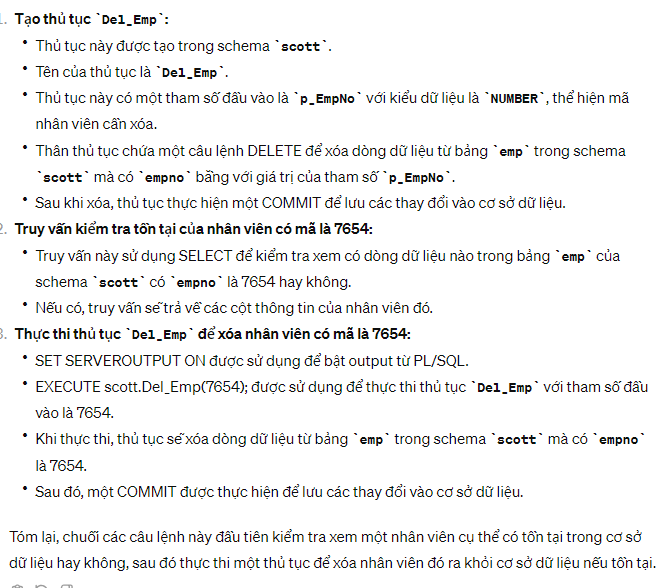
--Thuc thi ham xoa nhan vien co ma 7654 ra khoi bang

SET SERVEROUTPUT ON;

execute scott.Del\_Emp(7654);







--Cau 2:

--Ham lay tong tien luong cua mot phong ban

CREATE OR REPLACE FUNCTION scott.get\_dept\_salary(dno NUMBER) RETURN NUMBER

IS

all\_sal NUMBER;

BEGIN

all\_sal := 0;

FOR emp\_sal IN

(SELECT SAL FROM scott.emp WHERE deptno = dno AND sal is not null)

LOOP

all\_sal := all\_sal + emp\_sal.sal;

END LOOP;

RETURN all\_sal;

END scott.get\_dept\_salary;

SET SERVEROUTPUT ON;

SELECT scott.get\_dept\_salary(30) AS Tsal FROM DUAL;

DECLARE

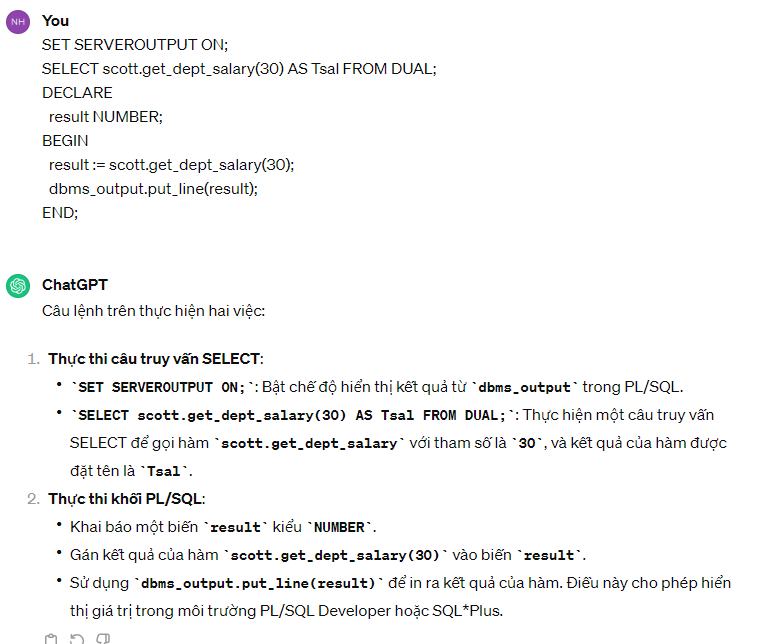
result NUMBER;

BEGIN

result := scott.get\_dept\_salary(30);

dbms\_output.put\_line(result);

END;

1. Tạo hàm get\_dept\_salary:
2. CREATE OR REPLACE FUNCTION scott.get\_dept\_salary(dno NUMBER) RETURN NUMBER: Đây là câu lệnh để tạo một hàm trong cơ sở dữ liệu Oracle. Hàm này có tên là get\_dept\_salary và thuộc schema scott. Nó nhận một tham số là dno có kiểu dữ liệu là NUMBER, đại diện cho mã phòng ban. Hàm trả về một giá trị kiểu NUMBER, tức là tổng lương của các nhân viên trong phòng ban đó.
3. IS: Phần bắt đầu của phần thân hàm.
4. all\_sal NUMBER;: Đây là khai báo biến all\_sal với kiểu dữ liệu NUMBER, được sử dụng để tính tổng lương của nhân viên trong phòng ban.
5. BEGIN: Bắt đầu của phần thân hàm.
6. all\_sal := 0;: Gán giá trị ban đầu cho biến all\_sal, tức là 0.
7. FOR emp\_sal IN ... LOOP ... END LOOP;: Đây là một vòng lặp FOR IN trong đó mỗi vòng lặp lấy giá trị lương của nhân viên từ kết quả của câu truy vấn SELECT. Cứ mỗi lần lặp, giá trị lương được cộng vào biến all\_sal.
8. RETURN all\_sal;: Trả về giá trị tổng lương đã tính được sau khi kết thúc vòng lặp.
9. END scott.get\_dept\_salary;: Kết thúc của hàm.
10. Thực thi hàm get\_dept\_salary trong câu truy vấn SELECT:
11. SELECT scott.get\_dept\_salary(30) AS Tsal FROM DUAL;: Đây là một câu truy vấn SELECT sử dụng hàm get\_dept\_salary để tính tổng lương của nhân viên trong phòng ban có mã 30. Kết quả của hàm được đặt tên là Tsal.
12. Thực thi hàm get\_dept\_salary trong khối PL/SQL:
13. DECLARE ... BEGIN ... END;: Đây là một khối PL/SQL được sử dụng để gán kết quả của hàm get\_dept\_salary vào một biến và sau đó in ra giá trị đó.
14. result NUMBER;: Khai báo một biến result với kiểu dữ liệu là NUMBER.
15. result := scott.get\_dept\_salary(30);: Gọi hàm get\_dept\_salary để tính tổng lương của nhân viên trong phòng ban có mã 30 và gán kết quả vào biến result.
16. dbms\_output.put\_line(result);: In ra giá trị của biến result sử dụng dbms\_output. Điều này cho phép hiển thị kết quả trong môi trường PL/SQL Developer hoặc SQL\*Plus.
17. 

--Cau 3:

create or replace procedure MucLuong(p\_jobid jobs.job\_id%TYPE, min\_lg OUT jobs.min.salary%TYPE, max\_lg OUT jobs.max.salary%TYPE)

IS

BEGIN

select min\_salary, max\_salary into min\_lg, max\_lg

from hr.jobs

where job\_id = p\_jobid;

exception

when no\_data\_found then

dbms\_output.put\_line ('Ma cong viec ' || p\_jobid || ' khong tim thay');

when others then

dbms\_output.put\_line ('Khong biet loi gi');

END;

SET SERVEROUTPUT ON;

declare

minl hr.jobs.min\_salary%TYPE;

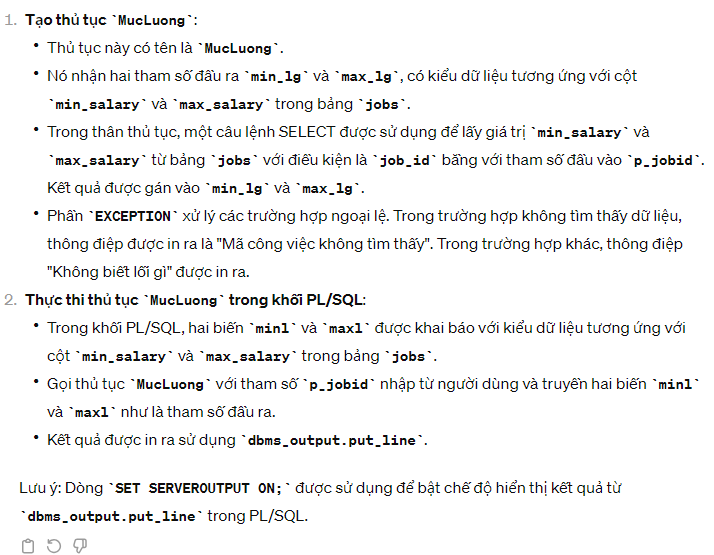
maxl hr.jobs.max\_salary%TYPE;

BEGIN

MucLuong('&Nhap\_ma\_cong\_viec', minl, maxl);

dbms\_output.put\_line('Min = ' || to\_char(minl) || ' Max = ' || to\_char(maxl));

END;



Buổi 8

--Cau 4: Tao trigger de kiem tra rang buoc du lieu

alter table scott.dept add budget number(10);

select deptno, sum(Sal) as total\_sal from scott.emp

group by deptno;

create or replace trigger check\_budget\_emp

after insert or update of sal, deptno on scott.emp

declare

cursor dept\_cur is

select deptno, budget from SCOTT.dept;

DNO scott.dept.deptno%type;

ALLSAL scott.dept.budget%type;

DEPT\_SAL number;

begin

open dept\_cur;

loop

fetch dept\_cur into DNO, ALLSAL;

exit when dept\_cur%notfound;

select sum(sal) into dept\_sal from scott.emp where deptno=DNO;

if DEPT\_SAL > ALLSAL then

raise\_application\_error(-20325, 'Tong luong trong phong ' || to\_char(DNO) || ' da vuot qua ngan sach');

end if;

end loop;

close dept\_cur;

end;

--Test trigger

update scott.dept set budget = 9500 where deptno =30;

update scott.emp set sal= 1800 where empno= 7499;

Dĩ nhiên! Dưới đây là giải thích từng dòng code trong trigger check\_budget\_emp:

Tạo trigger và khai báo các loại sự kiện kích hoạt:

CREATE OR REPLACE TRIGGER check\_budget\_emp: Bắt đầu định nghĩa một trigger mới hoặc thay thế trigger đã tồn tại nếu có.

AFTER INSERT OR UPDATE OF sal, deptno ON scott.emp: Trigger này sẽ được kích hoạt sau khi có sự thay đổi (insert hoặc update) trong cột sal hoặc deptno của bảng scott.emp.

Khai báo các biến cần thiết:

CURSOR dept\_cur IS SELECT deptno, budget FROM SCOTT.dept;: Khai báo một con trỏ (cursor) có tên dept\_cur, sẽ lặp qua các hàng trong bảng scott.dept và lấy các cột deptno và budget.

DNO scott.dept.deptno%TYPE;: Khai báo một biến DNO với kiểu dữ liệu tương ứng với cột deptno trong bảng scott.dept.

ALLSAL scott.dept.budget%TYPE;: Khai báo một biến ALLSAL với kiểu dữ liệu tương ứng với cột budget trong bảng scott.dept.

DEPT\_SAL NUMBER;: Khai báo một biến DEPT\_SAL kiểu NUMBER để lưu tổng lương của nhân viên trong mỗi phòng ban.

Bắt đầu thực thi trigger:

OPEN dept\_cur;: Mở con trỏ dept\_cur để bắt đầu lặp qua các hàng trong bảng scott.dept.

Bắt đầu vòng lặp:

LOOP: Bắt đầu vòng lặp.

Lấy dữ liệu từ con trỏ và kiểm tra điều kiện kết thúc vòng lặp:

FETCH dept\_cur INTO DNO, ALLSAL;: Lấy các giá trị deptno và budget từ hàng hiện tại trong con trỏ và gán cho các biến DNO và ALLSAL.

EXIT WHEN dept\_cur%NOTFOUND;: Kiểm tra xem con trỏ đã duyệt qua tất cả các hàng hay chưa. Nếu đã duyệt qua hết thì thoát khỏi vòng lặp.

Tính tổng lương của nhân viên trong phòng ban hiện tại:

SELECT SUM(sal) INTO dept\_sal FROM scott.emp WHERE deptno = DNO;: Tính tổng lương của các nhân viên trong phòng ban hiện tại (được xác định bởi DNO) và gán vào biến DEPT\_SAL.

So sánh tổng lương với ngân sách và xử lý ngoại lệ:

IF DEPT\_SAL > ALLSAL THEN ... END IF;: So sánh tổng lương (DEPT\_SAL) với ngân sách của phòng ban (ALLSAL). Nếu tổng lương vượt quá ngân sách, trigger sẽ gây ra một lỗi.

RAISE\_APPLICATION\_ERROR(-20325, 'Tong luong trong phong ' || TO\_CHAR(DNO) || ' da vuot qua ngan sach');: Khi tổng lương vượt quá ngân sách, trigger sẽ gây ra một lỗi với mã số -20325 và thông điệp tương ứng.

Kết thúc vòng lặp và đóng con trỏ:

CLOSE dept\_cur;: Khi vòng lặp kết thúc, con trỏ sẽ được đóng.

Trên thực tế, trigger này sẽ kiểm tra xem tổng lương của nhân viên trong mỗi phòng ban có vượt quá ngân sách của phòng ban hay không sau mỗi lần có sự thay đổi hoặc cập nhật lương của nhân viên.

--Cau 5: Tao Trigger de theo doi cap nhat du lieu

CREATE TABLE change\_sal\_emp (

UserName VARCHAR2(20),

MODIFY\_TIME DATE,

EMPNO NUMBER(4),

Old\_Sal NUMBER(7),

New\_Sal NUMBER(7)

);

CREATE OR REPLACE TRIGGER store\_change\_sal\_EMP

AFTER UPDATE OF sal ON scott.emp

FOR EACH ROW

BEGIN

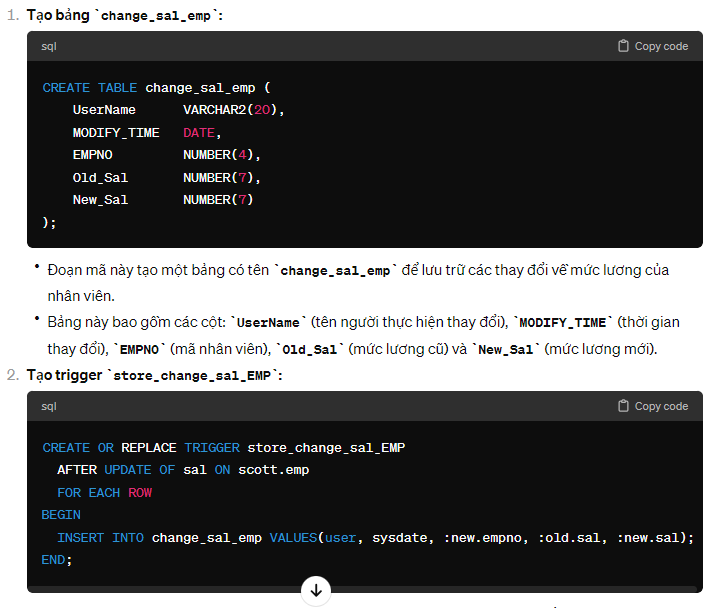
INSERT INTO change\_sal\_emp VALUES(user, sysdate, :new.empno, :old.sal, :new.sal);

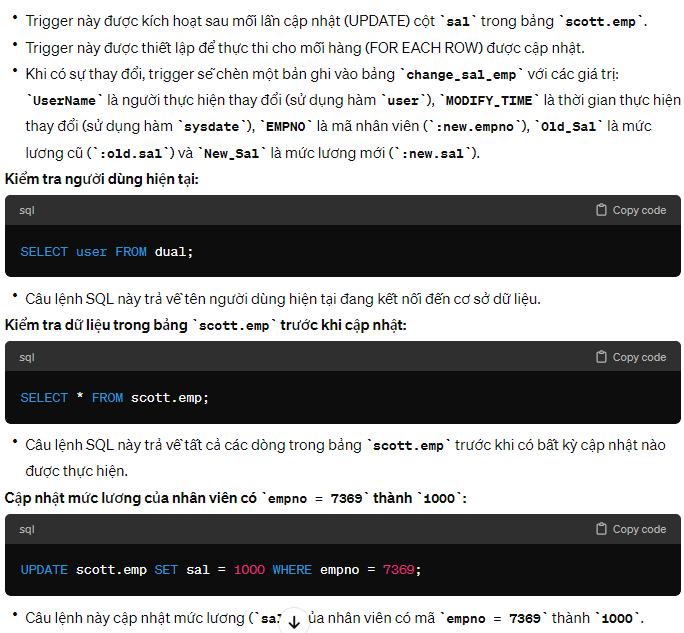
END;

select user from dual;

select \* from scott.emp;

update scott.emp set sal = 1000 where empno = 7369;





Khi một cập nhật được thực hiện trên cột sal của bảng scott.emp, trigger store\_change\_sal\_EMP sẽ được kích hoạt và chèn một bản ghi mới vào bảng change\_sal\_emp để lưu trữ thông tin về thay đổi mức lương của nhân viên.

--Cau 6: Tao Trigger ngan kiem tra viec nhap lieu

alter table thebh add conhl number(7);

alter table thebh add ngaykt date;

create trigger Da\_co\_BH before insert on thebh for each row

declare

sodong int;

begin

select count(\*) into sodong from thebh

where makh=:new.makh and maloai=:new.maloai

and ngaykt<sysdate;

if sodong>0 then

raise\_application\_error(-20111, 'KH nay hien van con duoc BH loai nay');

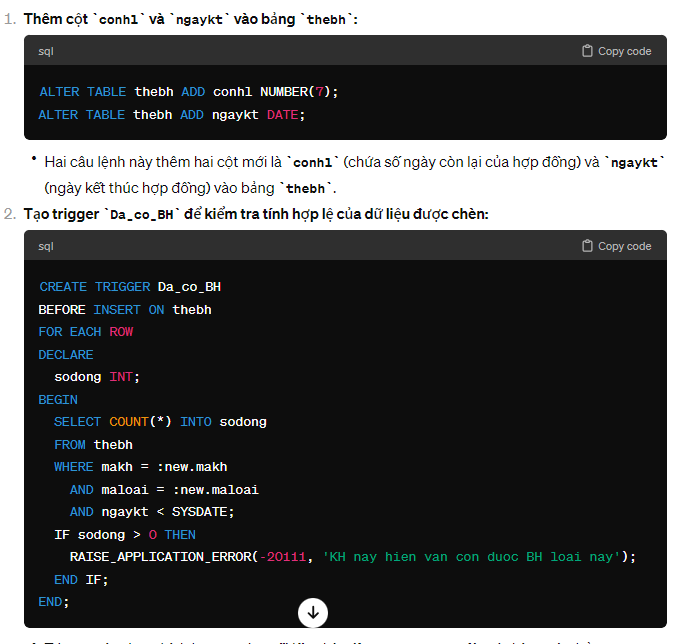
end if;

end;

--Test Trigger

insert into thebh (maloai, ngaybd, makh, thoihan, ngaykt, conhl) values ('YT', sysdate, '006', 6, add\_months(sysdate, 6), 1);

insert into thebh (maloai, ngaybd, makh, thoihan, ngaykt, conhl) values ('YT', sysdate+1, '006', 6, add\_months(sysdate+1, 6), 1);



* + Trigger này được kích hoạt trước mỗi lần chèn (BEFORE INSERT) một hàng vào bảng thebh.
  + Trigger được thiết lập để thực thi cho mỗi hàng (FOR EACH ROW) được chèn.
  + Trong trigger, biến sodong được khai báo để lưu số lượng hàng trong bảng thebh có cùng makh, maloai và ngaykt < SYSDATE.
  + Nếu sodong lớn hơn 0 (tức là đã tồn tại hợp đồng hợp lệ cho khách hàng với loại bảo hiểm này), trigger sẽ gây ra một lỗi (RAISE\_APPLICATION\_ERROR) với mã số -20111 và thông báo 'KH nay hien van con duoc BH loai nay'.
  + Thử nghiệm trigger:
  + Hai câu lệnh INSERT được sử dụng để thêm dữ liệu vào bảng thebh.
  + Cả hai hàng được chèn đều có maloai là 'YT' và makh là '006', và ngaykt được thiết lập để làm cho hợp đồng hết hạn trước thời điểm hiện tại (ngày thực hiện thử nghiệm).

--Cau 7: Tao Trigger tu dong luu du lieu

create table mucphi(

maloai char(2),

mucphi char(15),

ngaybd date,

nguoicn varchar2(30)

);

create or replace trigger CN\_Phi after insert or update on LoaiBH

for each row

declare

begin

insert into mucphi (maloai, mucphi, ngaybd, nguoicn) values (:new.maloai, :new.mucphi, sysdate, user);

end;

update loaibh set mucphi=8 where maloai='TS';

