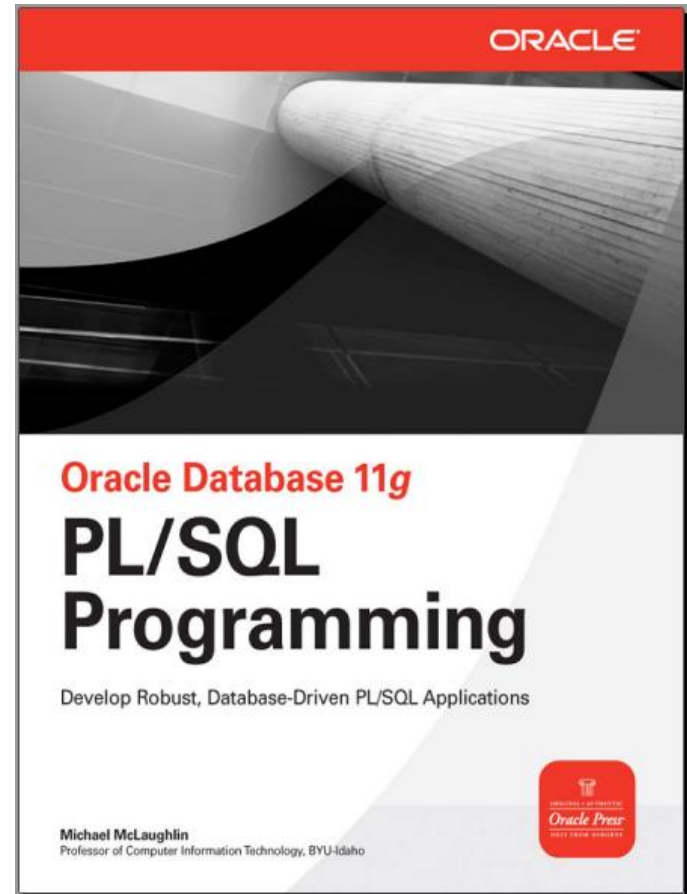
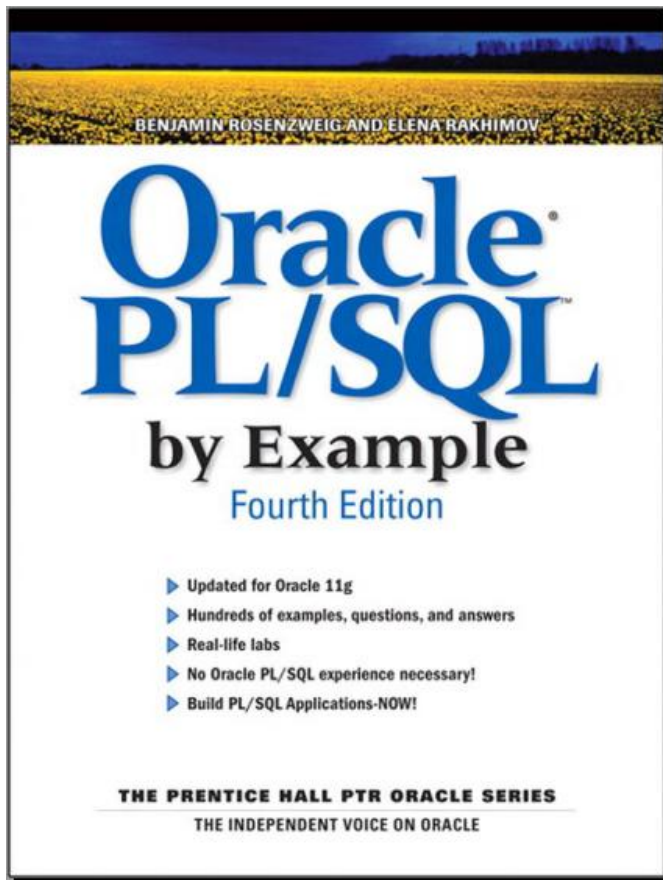




NGÔN NGỮ PL/SQL

Giảng viên: Cao Thị Nhâm

Tài liệu tra cứu



Nội dung chính

- ❖ PL/SQL là gì?
- ❖ Cấu trúc khối lệnh của PL/SQL
- ❖ Biến số, hằng số
- ❖ Phép gán, các phép toán
- ❖ Cấu trúc điều khiển: điều kiện rẽ nhánh, lặp
- ❖ Con trỏ
- ❖ Hàm và thủ tục
- ❖ Trigger
- ❖ Package



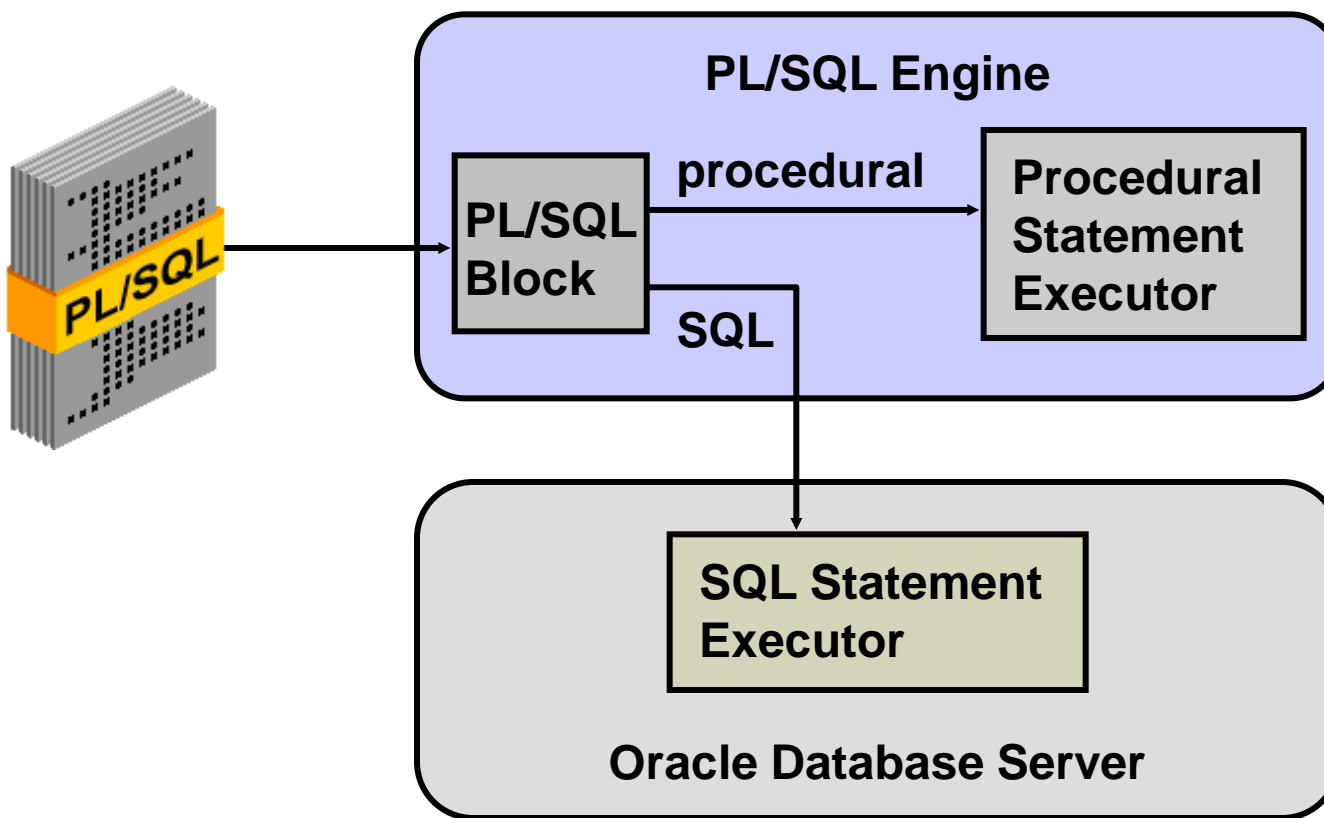
PL/SQL là gì?

- SQL: Structure Query Language
- SQL chưa đủ mạnh để lập trình
- PL/SQL: Procedural Language extensions for SQL

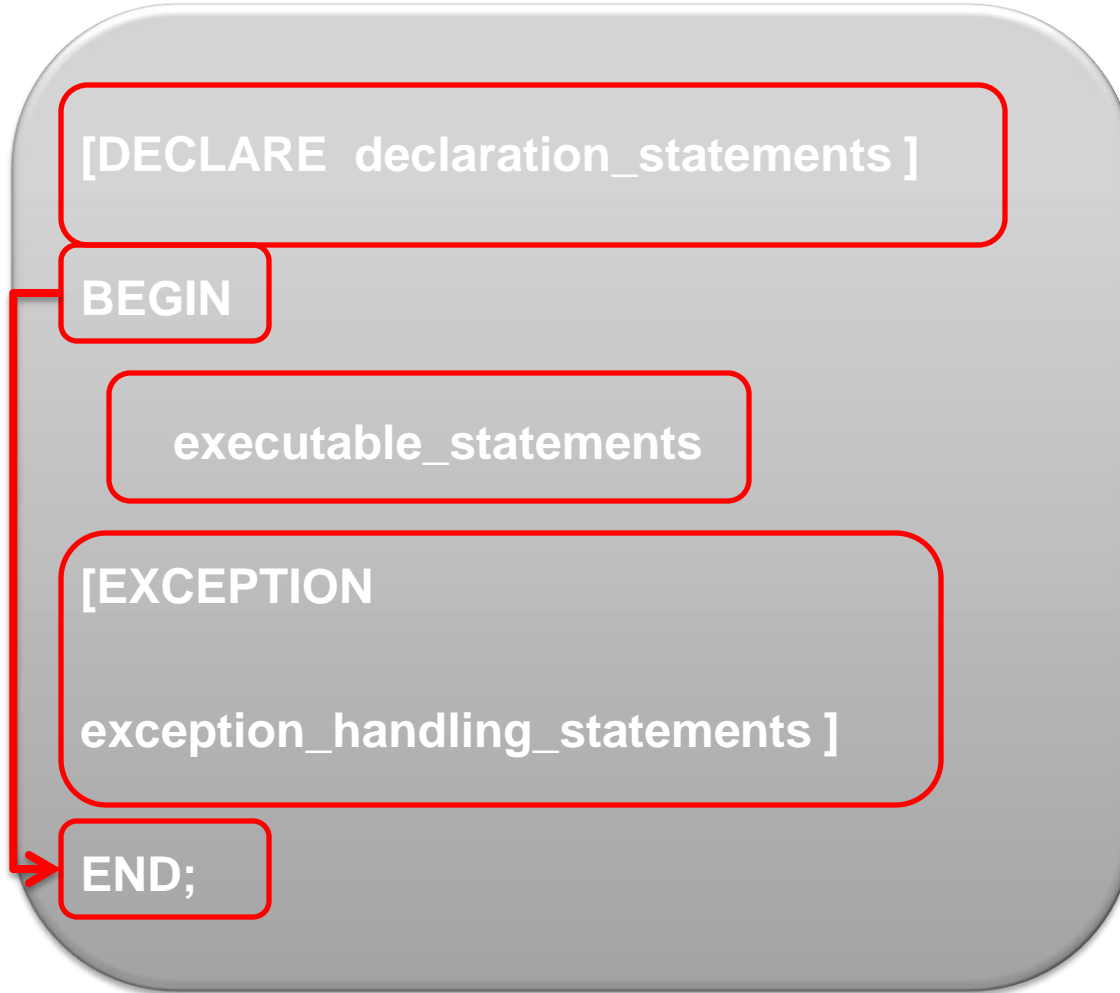


- Điểm mạnh của PL/SQL:
 - Tích hợp cấu trúc hướng thủ tục vào SQL
 - Tăng hiệu năng xử lý
 - Module hóa chương trình
 - Khả chuyển
 - Có cơ chế xử lý ngoại lệ

Cách thực thi các lệnh PL/SQL



Cấu trúc khối lệnh



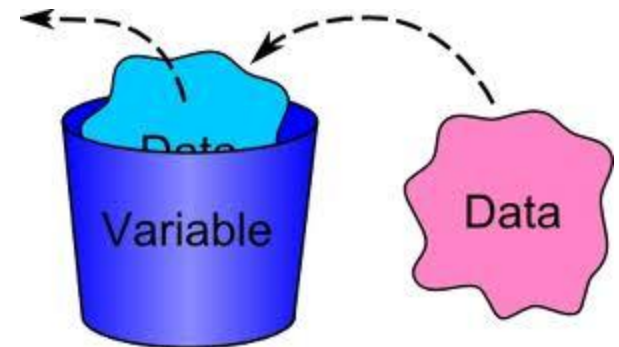
Biến (variable)



**Biến
là gì?**

❖ Đặc điểm của biến:

- Lưu trữ dữ liệu tạm thời
- Cho phép sửa dữ liệu
- Cho phép tái sử dụng



Quy tắc đặt tên biến

- ❖ Bắt đầu bằng chữ cái
- ❖ Có thể chứa cả số và chữ cái
- ❖ Có thể chứa kí tự đặc biệt: dấu \$, gạch dưới, ... (hạn chế dùng \$)
- ❖ Tối đa 30 kí tự
- ❖ Không trùng với từ khóa mà Oracle sử dụng, ví dụ: varchar, table...



Khai báo và khởi tạo giá trị cho biến

- ❖ Đặt trong phần DECLARE
- ❖ Khai báo biến

```
Tên_biến kiểu_dữ_liệu [NOT NULL] [:= expr];
```

- ❖ Ví dụ

```
DECLARE  
  emp_hiredat     DATE;  
  emp_deptno      NUMBER(2) NOT NULL := 10;  
  location        VARCHAR2(13) := 'Atlanta';  
  c_comm          CONSTANT NUMBER := 1400;
```

Gán giá trị cho biến

❖ Có thể gán giá trị theo 2 cách

- Gán trực tiếp:

```
Tên_biến := giá_trị;
```

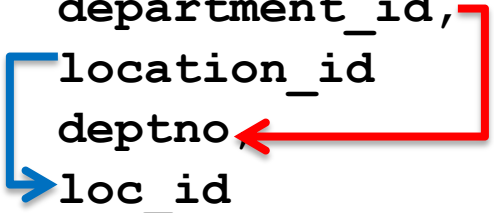
- Gán “gián tiếp”:

```
SELECT tên_cột_1, tên_cột_2, ...  
INTO tên_biến_1, tên_biến_2,...  
FROM tên_bảng  
[WHERE điều_kiện];
```


Gán giá trị cho biến – ví dụ

```
DECLARE
    deptno          NUMBER(4) ;
    loc_id           NUMBER(4) ;
    empno            CHAR(5) ;
BEGIN
    empno := '00010' ;
    SELECT  department_id,
            location_id
    INTO    deptno,
            loc_id
    FROM    departments
    WHERE   department_name
            = 'Sales' ;

    ...
END ;
/
```



Phép toán

- Toán học
 - Logic
 - So sánh
- 
- Giống với SQL**
- Lũy thừa (**)
 - Ví dụ: $4^{**}2 = 16$

Cấu trúc điều khiển

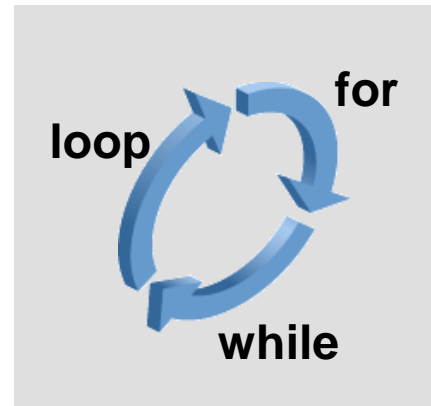
❖ Rẽ nhánh

- IF... THEN...ELSE
- CASE...WHEN



❖ Lặp

- Vòng lặp đơn giản
- Vòng lặp WHILE
- Vòng lặp FOR
- GOTO



Rẽ nhánh

IF...THEN...ELSE	CASE...WHEN
<pre>IF <i>condition</i> THEN <i>statements</i>; [ELSIF <i>condition</i> THEN <i>statements</i>;] [ELSE <i>statements</i>;] END IF;</pre>	<pre>CASE selector WHEN expression1 THEN result1 WHEN expression2 THEN result2 ... WHEN expressionN THEN resultN [ELSE resultN+1] END;</pre>

Vòng lặp

- ❖ Vòng lặp: thực hiện lặp đi lặp lại một dòng lệnh hoặc tập hợp các dòng lệnh.
- ❖ Có 3 loại lặp cơ bản:
 - Lặp đơn giản
 - FOR
 - WHILE
- ❖ Ngoài ra có thể sử dụng lệnh GOTO để lặp



Lặp đơn giản

❖ Cú pháp

```
LOOP  
    statement1;  
    . . .  
    EXIT [WHEN condition];  
END LOOP;
```


Vòng lặp đơn giản – ví dụ

```
DECLARE
  countryid      CHAR(5) := '00001';
  loc_id         NUMBER(4);
  counter        NUMBER(2) := 1;
  new_city       VARCHAR2(20) := 'Montreal';
BEGIN
  SELECT MAX(location_id) INTO loc_id FROM locations
  WHERE country_id = countryid;
  LOOP
    INSERT INTO locations
      VALUES((loc_id + counter), new_city, countryid);
    counter := counter + 1;
    EXIT WHEN counter > 3;
  END LOOP;
END;
```

Vòng lặp FOR

❖ Cú pháp

```
FOR counter IN [REVERSE]  
    lower_bound..upper_bound LOOP  
    statement1;  
    statement2;  
    . . .  
END LOOP;
```

Vòng lặp FOR – ví dụ

```
DECLARE
    countryid      CHAR(5);
    loc_id         NUMBER(4);
    new_city       VARCHAR2(20) := 'Montreal';
BEGIN
    SELECT MAX(location_id) INTO loc_id
    FROM locations
    WHERE country_id = countryid;

    FOR i IN 1..3 LOOP
        INSERT INTO locations
            VALUES((loc_id + i), new_city, countryid );
    END LOOP;
END;
```

Vòng lặp WHILE

❖ Cú pháp

```
WHILE condition LOOP  
    statement1;  
    statement2;  
    . . .  
END LOOP;
```

Vòng lặp WHILE – ví dụ

```
DECLARE
    countryid      CHAR(5);
    loc_id         NUMBER(4);
    new_city       VARCHAR2(20) := 'Montreal';
BEGIN
    SELECT MAX(location_id) INTO loc_id
    FROM locations
    WHERE country_id = countryid;

    WHILE counter <= 3 LOOP
        INSERT INTO locations
            VALUES((loc_id + counter), new_city, countryid);
        counter := counter + 1;
    END LOOP;
END;
```

Con trỏ

- ❖ Là kiểu biến có cấu trúc, cho phép xử lý dữ liệu gồm nhiều dòng
- ❖ Có 2 loại con trỏ
 - Không tường minh
 - Tường minh

Con trỏ không tường minh

- ❖ Là con trỏ PL/SQL tự động sinh ra khi gặp câu lệnh SELECT hoặc DML
- ❖ User chỉ có thể lấy thông tin của con trỏ
 - **SQL%ISOPEN**: Trả về FALSE
 - **SQL%FOUND**: Trả về NULL/TRUE/ FALSE
 - **SQL%NOTFOUND**: Trả về NULL/TRUE/ FALSE
 - **SQL%ROWCOUNT**: Trả về NULL, số lượng bản ghi tác động bởi DML hoặc SELECT

Con trỏ tường minh

- ❖ Con trỏ do người dùng tự định nghĩa
- ❖ Các bước sử dụng con trỏ:

- Bước 1: Khai báo

CURSOR tên_con_trỏ(danh sách biến) **IS** câu_truy_vấn;

- Bước 2: Mở con trỏ

OPEN tên_con_trỏ | tên_con_trỏ(danh sách biến);

- Bước 3: Lấy dữ liệu xử lý

FETCH tên_con_trỏ **INTO** danh_sách_biến;

- Bước 4: Đóng con trỏ

CLOSE Tên cursor;

Con trỏ (cursor)...

❖ Các thuộc tính của con trỏ:

- **Tên_con_trỏ%ISOPEN**
- **Tên_con_trỏ%NOTFOUND**
- **Tên_con_trỏ%FOUND**
- **Tên_con_trỏ%ROWCOUNT**

❖ Khai báo kiểu dữ liệu “nhanh”:

- **%TYPE**
- **%ROWTYPE**

Con trỏ (cursor)...

```
DECLARE
CURSOR c_Emp IS SELECT empno, ename, job FROM emp
                WHERE dept_id = 10;

v_empno CHAR(5);
v_ename VARCHAR2(20);
v_job VARCHAR2 (20);
BEGIN
OPEN c_Emp;
LOOP
    FETCH c_Emp INTO v_empno, v_ename, v_job;
    EXIT WHEN c_Emp%notfound;
    INSERT INTO Emp_ext (empno, ename, job)
    VALUES (v_empno, v_ename, v_job);
END LOOP;
CLOSE c_Emp;
END;
```

Hàm và thủ tục

- ❖ Một nhóm lệnh thực hiện một chức năng cụ thể được nhóm lại
- ❖ Mục đích:
 - Tăng khả năng xử lý
 - Tăng khả năng sử dụng chung
 - Tăng tính bảo mật và an toàn dữ liệu
- ❖ Lưu trữ trong CSDL dưới dạng p-code



Thủ tục

❖ Cú pháp

```
CREATE [OR REPLACE] PROCEDURE procedure_name
  [(argument1 [mode1] datatype1,
    argument2 [mode2] datatype2,
    . . .)]
IS|AS
procedure_body;
```

❖ Gọi thủ tục

- EXEC *tên_thủ_tục*;
- *Tên_thủ_tục*;

Thủ tục – ví dụ

```
CREATE PROCEDURE add_dept IS  
  dept_id NUMBER(4) ;  
  dept_name VARCHAR2(50) ;  
BEGIN  
  dept_id:=280 ;  
  dept_name:='ST-Curriculum' ;  
  INSERT INTO dept VALUES(dept_id,dept_name) ;  
  DBMS_OUTPUT.PUT_LINE(' Inserted ' ||  
SQL%ROWCOUNT || ' row ');  
END;
```

Hàm

❖ Cú pháp

```
CREATE [OR REPLACE] FUNCTION function_name
  [(argument1 [mode1] datatype1,
    argument2 [mode2] datatype2,
    . . .)]
RETURN datatype
IS|AS
function_body;
```

❖ Gọi hàm

- Tên *_biến* := tên_hàm;
- Dùng trong câu lệnh truy vấn

Hàm – ví dụ

```
CREATE FUNCTION check_sal RETURN Boolean IS
  dept_id NUMBER(4);
  empno    NUMBER(4);
  sal      NUMBER(8);
  avg_sal  NUMBER(8,2);
BEGIN
  empno:=205;
  SELECT salary,department_id INTO sal,dept_id
  FROM employees WHERE employee_id= empno;
  SELECT avg(salary) INTO avg_sal FROM employees
  WHERE department_id=dept_id;
  IF sal > avg_sal THEN
    RETURN TRUE;
  ELSE
    RETURN FALSE;
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN NULL;
END;
/
```

Hủy bỏ và sửa thủ tục/hàm

❖ Hủy

```
DROP PROCEDURE tên_thủ_tục;
```

```
DROP FUNCTION tên_hàm;
```

❖ Sửa

```
ALTER PROCEDURE tên_thủ_tục ...
```

```
ALTER FUNCTION tên_hàm ...
```


So sánh thủ tục và hàm

Thủ tục	Hàm
Thực hiện giống như thực hiện các câu lệnh	Có thể được gọi giống như một phần của lệnh PL/SQL
Không có kiểu giá trị trả về	Có chứa giá trị trả về
Có thể trả về một hoặc nhiều tham số	Trả về một giá trị

Trigger



1. Trigger là gì?

2. Trigger dùng để làm gì?

❖ Là một thủ tục được thực hiện ngầm định ngay khi thực hiện lệnh SQL nhằm đảm bảo các quy tắc logic phức tạp của dữ liệu.

❖ Các loại trigger:

- DDL trigger
- DML trigger
- Compound trigger
- Instead-of trigger
- System/database trigger

Chú ý khi sử dụng trigger

❖ Chú ý khi sử dụng trigger:

- Chỉ sử dụng trigger với các thao tác trọng tâm
- Không sử dụng trigger cho trường hợp có thể sử dụng constraint
- Trigger có thể gây khó khăn cho việc bảo trì và phát triển hệ thống lớn

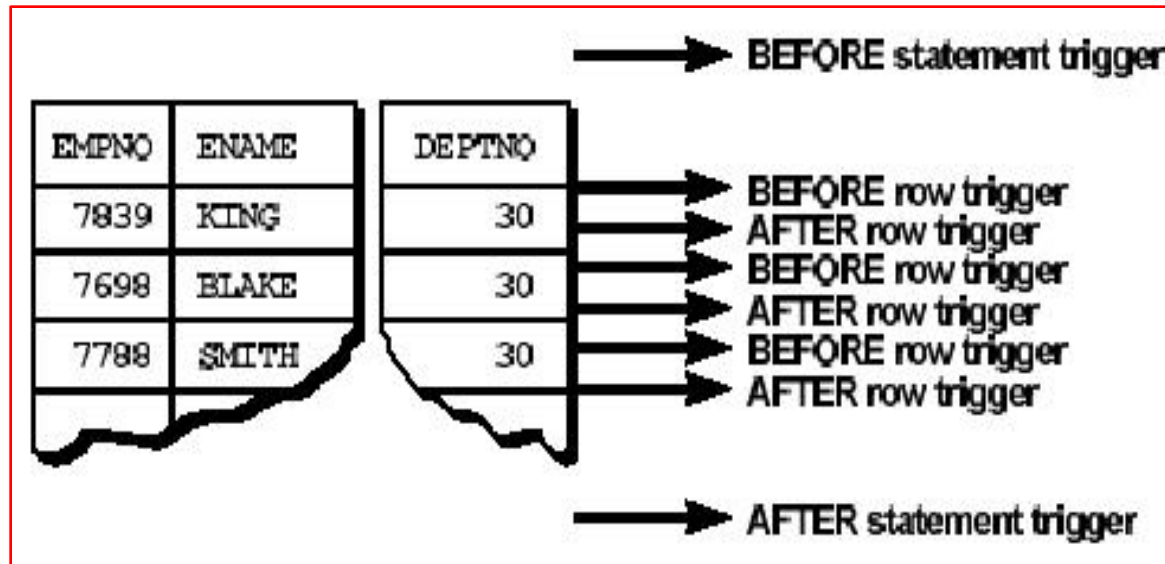


Chỉ sử dụng trigger khi thật cần thiết

Phân loại trigger DML

- ❖ Phân theo thời gian thực hiện
 - BEFORE
 - AFTER
- ❖ Phân loại theo loại câu lệnh kích hoạt
 - INSERT
 - UPDATE
 - DELETE
- ❖ Phân loại theo số lần kích hoạt
 - Mức câu lệnh
 - Mức banr ghi

Phân loại trigger...



Tạo trigger

❖ Mức câu lệnh

```
CREATE [OR REPLACE] TRIGGER  
trigger_name  
timing event1 [OR event2 OR  
event3]  
ON table_name  
BEGIN  
PL/SQL Block;  
END;
```

Tạo trigger...

❖ Mức dòng

```
CREATE [OR REPLACE] TRIGGER
trigger_name timing event1 [OR
event2 OR event3]
ON table_name
FOR EACH ROW
[WHEN condition]
BEGIN
PL/SQL Block;
END;
```

Instead-of trigger

- ❖ Cú pháp viết như trigger DML
- ❖ Chỉ được dùng cho view

Quản lý trigger

❖ Thay đổi trạng thái

```
ALTER TRIGGER trigger_name DISABLE | ENABLE;
```

```
ALTER TABLE table_name DISABLE | ENABLE ALL TRIGGERS;
```

❖ Hủy trigger

```
DROP TRIGGER trigger_name;
```

Package

- ❖ Tham khảo trong tài liệu:
User&PLSQL.pdf (trang 47)



PL/SQL
ORACLE

ORACLE
DATABASE **11^g**

ORACLE