

BASE DE DADOS



Oracle - PL/SQL
Triggers

Teóricas
Ano Letivo 2018/2019
Rosa Reis

TRIGGERS

Introdução

- ❖ Semelhante aos Procedimentos e às Funções.
- ❖ Associados a Tabelas e a Views
- ❖ Executados automaticamente quando:
 - Há modificação de dados (DML Trigger)
 - ✧ INSERT, UPDATE, coluna UPDATE ou DELETE
 - modificação de esquema (DDL Trigger)
 - eventos do sistema, login / logoff do utilizador (Trigger de sistema)

PL/SQL Block

association

Table, View, etc.

TRIGGERS

Regras

- ❖ Não definir triggers para duplicar ou substituir ações que se pode fazer facilmente de outras formas
 - ❖ Por exemplo, implementar regras simples de integridade de dados usando restrições, não triggers.
- ❖ Usar triggers somente quando necessário. O uso excessivo de triggers pode resultar em interdependências complexas (Cascading Triggers) que dificultam a manutenção de grandes aplicações
- ❖ Evitar a lógica longa do trigger criando procedimentos armazenados que são chamados no corpo do trigger

TRIGGERS

Tipos

- ❖ Classificados de duas formas:
- ❖ Quando eles executam: **antes, depois ou em vez de** - a instrução que aciona o trigger (*instrução de triggering*)
- ❖ Pelo número de vezes que o corpo do trigger é executado
 - ❖ Uma vez para toda a instrução DML (trigger de instrução)
 - ❖ Uma vez por cada linha afetada pela instrução DML (trigger de linha)

TRIGGERS

Sintaxe

CREATE [OR REPLACE] TRIGGER *trigger_name*

AFTER | BEFORE | INSTEAD OF *a_trigger_event* → *instrução de triggering*

ON *table_name (or view_name)*

[FOR EACH ROW [WHEN *trigger_condition*]] → *restrição*

DECLARE (opcional)

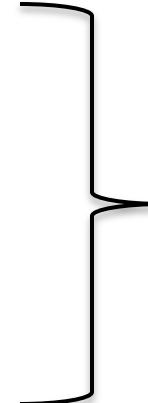
BEGIN (obrigatório)

Executado unicamente quando a condição de Trigger é TRUE

EXCEPTION (opcional)

Exception Section

END; (obrigatório)



ação ou corpo do trigger

NOTA: Um *trigger_event* pode ser qualquer combinação de um INSERT, DELETE e/ou UPDATE numa tabela ou view

Considerações antes de se criar um trigger

Seções do CREATE TRIGGER que devemos considerar antes de se criar um trigger:

```
CREATE [OR REPLACE] TRIGGER trigger_name
    timing
    event1 [OR event2 OR event3] ON object_name
    trigger_body
```

- **Timing:** quando o trigger dispara em relação ao evento desencadeador.
 - Os valores são BEFORE , AFTER ou INSTEAD OF.
- **Event:** Qual a operação DML que faz com que o trigger seja acionado.
 - Os valores são INSERT, UPDATE [OF column] e DELETE.
- **Object_name:** a tabela ou exibição associada ao trigger.
- **Trigger_body:** as ações executadas pelo gatilho são definidas em um bloco anônimo.

TRIGGERS

Trigger Timings and Events Exemplos

- ❖ O Trigger é executado imediatamente antes que o salário de um funcionário seja atualizado:

```
CREATE OR REPLACE TRIGGER sal_upd_trigg
BEFORE UPDATE OF salary ON employees
BEGIN ... END ;
```

- ❖ O Trigger é executado imediatamente depois que um funcionário é excluído:

```
CREATE OR REPLACE TRIGGER emp_del_trigg
AFTER DELETE ON employees
BEGIN ... END ;
```

TRIGGERS

Exemplos - Trigger Timings and Events

- ❖ Podemos restringir um UPDATE a uma especifica coluna ou varias

```
CREATE OR REPLACE TRIGGER sal_upd_trigg
BEFORE UPDATE OF salary, commission_pct ON employees
BEGIN ... END;
```

- ❖ Pode ter mais do que um triggering event

```
CREATE OR REPLACE TRIGGER emp_del_trigg
AFTER INSERT OR DELETE OR UPDATE ON employees
BEGIN ... END;
```

TRIGGERS

Predicados Condicionais

- ❖ Na acção do trigger utilizam-se predicados condicionais (INSERTING, DELETING e UPDATING) para verificar qual dos tipos de operação “disparou” o trigger.

```
CREATE OR REPLACE TRIGGER conta_aluno
BEFORE INSERT OR DELETE OR UPDATE ON aluno
DECLARE
var1 char(1);
BEGIN
IF INSERTING THEN
    DBMS_OUTPUT.PUT_Line( 'Codigo para o inserir')
END IF;
ESEIF UPDATING THEN..... ENDIF;
.......
```

TRIGGERS

Predicados Condicionais

- ❖ Podemos usar predicados condicionais para testar o UPDATE numa coluna específica:

```
CREATE OR REPLACE TRIGGER secure_emp
  BEFORE UPDATE ON employees
BEGIN
  IF UPDATING('SALARY') THEN
    IF TO_CHAR(SYSDATE, 'DY') IN ('SAT', 'SUN')
      THEN RAISE_APPLICATION_ERROR
        (-20501, 'You may not update SALARY on the weekend');
    END IF;
  ELSIF UPDATING('JOB_ID') THEN
    IF TO_CHAR(SYSDATE, 'DY') = 'SUN'
      THEN RAISE_APPLICATION_ERROR
        (-20502, 'You may not update JOB_ID on Sunday');
    END IF;
  END IF;
END;
```

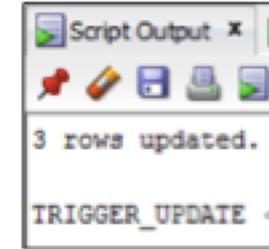
TRIGGERS

Triggers – de Instrução

- ❖ Tem a finalidade de tratar a execução de ações sobre tabelas independentemente de quantas linhas forem afetadas.
- ❖ Através deste tipo de Trigger podemos registrar a execução de comandos INSERT, UPDATE e DELETE contra tabelas que tenham Triggers contemplando essas ações.
- ❖ Caso um comando UPDATE atualize 1000 linhas, um Trigger deste tipo apenas dispara 1 única vez. Este tipo de Trigger não pode referenciar qualquer valor contido numa coluna da tabela. Isso ocorre porque o mesmo dispara uma única vez.

```
UPDATE trabalhadores SET salario = salario * 1.1;

CREATE OR REPLACE TRIGGER trg_demo_1
BEFORE UPDATE OF salario ON trabalhadores
BEGIN
    DBMS_OUTPUT.PUT_LINE('TRIGGER_UPDATE');
END;
/
```



1 disparo sobre
toda a tabela

TRIGGERS

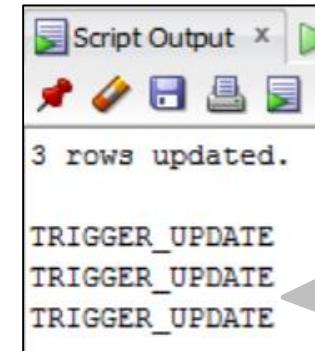
Triggers – de Linha

- ❖ para cada linha afectada pela instrução de triggering, tal como um Update statement que actualiza ‘n’ linhas. (triggers de linha)
- ❖ Temos acesso aos dados de cada linha afetada;
- ❖ Podemos especificar a cláusula **WHEN** (condição)
 - ❖ Especifica condição de disparo do trigger;
- ❖ não pode ler ou modificar a tabela à qual o trigger está associado - **TABELA MUTANTE**

Com cláusula

```
UPDATE trabalhadores SET salario = salario * 1.1;

CREATE OR REPLACE TRIGGER trg_demo_2
BEFORE UPDATE OF salario ON trabalhadores
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('TRIGGER_UPDATE');
END;
/
```



1 disparo sobre
cada linha

TRIGGERS

Triggers Instrução e Triggers de Linha

- ❖ As seguintes instruções não distinguem triggers de instrução de triggers de linha

```
Insert into Departamento value (50,'Educação');
```

```
Update Departamento
Set local =' Lisboa'
Where nrdep=50;
```

```
Delete from Departamento
Where nrdep=50;
```

| Departamento | | |
|--------------|-------------|---------------|
| nrdep | nome | Local |
| 10 | informatica | Porto |
| 20 | vendas | Braga |
| 40 | compras | Vila do Conde |
| 50 | educacao | Lisboa |

TRIGGERS

Triggers Instrução e Triggers de Linha

- ❖ As seguintes instruções distinguem triggers de instrução de triggers de linha

```
Insert into Empregado (nremp, nome)
Select empno,ename
From empregado_backup;
```

```
Update Empregado
Set salario =salario * 1.1
Where nrdep=10;
```

```
Delete From Empregado
where nrdep=10;
```

| Empregado | | | |
|-----------|--------|---------|-------|
| nremp | nome | salario | nrdep |
| 10 | João | 1000€ | 10 |
| 20 | Ana | 1500€ | 10 |
| 40 | Carlos | 3500€ | 10 |
| 50 | Miguel | 850€ | 20 |

TRIGGERS

[For Each Row[when trigger_condition]]

❖ For EACH ROW

- ❖ Esta opção especifica um trigger de linha;
- ❖ Por omissão estamos perante um trigger do tipo instrução;

❖ When (condição)

- ❖ restringe a execução da ação do trigger
- ❖ Especifica a condição de disparo do trigger - Trigger designado de Condicional
- ❖ Tem como objetivo evitar disparos desnecessário.
- ❖ Na condição podemos referenciar Pseudo-Registros (New e Old) e pode invocar só funções SQL predefinidas;
- ❖ Não podemos usar subqueries

```
CREATE [OR REPLACE] TRIGGER nome_trigger
{ BEFORE | AFTER }
{ INSERT | DELETE | UPDATE | UPDATE OF lista_colunas } ON tabela/vista
[ FOR EACH ROW [ WHEN (condição) ] ]
```

...

de disparo do trigger

TRIGGERS

[For Each Row[when trigger_condition]]

```
CREATE TRIGGER Calcula_Comissão
AFTER INSERT OR UPDATE OF sal on empregado
FOR EACH ROW WHEN ( new.job= 'vendedor')
BEGIN
.....
End;
```

Insert into Empregado values (12, 'Maria Jose', 3000, 'vendedor')

Insert into Empregado values (12, 'Maria Jose', 3000, 'professor')

Não
executa

TRIGGERS

Triggers – Aceder aos valores de atributos

- ❖ No corpo dum trigger é possível aceder aos valores antigos e novos dos atributos do registo afectado pela instrução de triggering.
- ❖ Existem dois nomes de correlação para cada coluna da tabela a ser modificada:
 - ❖ um para o valor antigo (:OLD)
 - ❖ **:OLD.nome_atributo** - indica o valor anterior de um campo que está a ser alterado por um comando DELETE ou UPDATE
 - ❖ outro para o valor novo (:NEW):
 - ❖ **:NEW.nome_atributo** . Indica um novo valor para um campo que está a ser alterado por um comando INSERT ou UPDATE

| Operação | :OLD | :NEW |
|---------------|------|------|
| INSERT | x | ✓ |
| UPDATE | ✓ | ✓ |
| DELETE | ✓ | x |

TRIGGERS

Pseudo-Registros NEW e OLD

```
CREATE OR REPLACE TRIGGER log_emps
  AFTER UPDATE OF salary ON employees FOR EACH ROW
BEGIN
  INSERT INTO log_emp_table
    (who, when, which_employee, old_salary, new_salary)
  VALUES (USER, SYSDATE, :OLD.employee_id,
          :OLD.salary, :NEW.salary);
END;
```

O :NEW e o :OLD podem ser personalizados na cláusula REFERENCING do Comando CREATE TRIGGER

```
CREATE OR REPLACE TRIGGER trg_demo_3
  BEFORE INSERT OR UPDATE ON barcos
  REFERENCING NEW AS BARCO
  FOR EACH ROW WHEN (BARCO.cor='azul')
  DECLARE
    corSubstituta VARCHAR(10);
  BEGIN
    corSubstituta := 'encarnada';
    :BARCO.cor := corSubstituta;
  END;
  /
```

TRIGGERS

Exemplo

```
create table cartaodebito(cd_id int, data_validade date);
create table identificador(identificador_id int, ativo number(1), cd_id int);
create table passagem(passagem_id int, identificador_id int, data_passagem date);
```

```
create or replace trigger trg_passagem after insert on passagem for each row
declare
    v_data_validade date;
    v_cd_id int;
begin
    select cd_id into v_cd_id from identificador where identificador_id = :new.identificador_id;
    select data_validade into v_data_validade from cartaodebito where cd_id = v_cd_id;
    if v_data_validade < :new.data_passagem then
        update identificador set ativo = 0 where identificador_id = :new.identificador_id;
    end if;
end;
```

```
Declare
  v_passagem_id int;
  v_identificador_id int;
  v_data_passagem date;
Begin
  ...
  insert into pagassem(passagem_id, identificador_id, data_passagem) values (v_passagem_id, v_identificador_id, v_data_passagem);
  ...
end;
```

TRIGGERS

Tabelas Mutantes e Triggers de Linha

- ❖ Uma tabela mutante é uma tabela que está a ser modificada por uma instrução DML.
- ❖ Trigger de linha não pode ter um SELECT de uma tabela mutante, porque ele considera existência de conjunto inconsistente de dados (os dados na tabela seriam alterados enquanto o trigger tenta lê-lo).
- ❖ No entanto, um trigger de linha pode ter um SELECT de uma tabela diferente, se necessário.
- ❖ Esta restrição não se aplica triggers de instrução DML, apenas aos Triggers de linha DML.

TRIGGERS

Tabelas Mutantes e Triggers de Linha -Exemplo

```
CREATE OR REPLACE TRIGGER check_salary
  BEFORE INSERT OR UPDATE OF salary, job_id ON
employees
  FOR EACH ROW
DECLARE
  v_minsalary employees.salary%TYPE;
  v_maxsalary employees.salary%TYPE;
BEGIN
  SELECT MIN(salary), MAX(salary)
    INTO v_minsalary, v_maxsalary
   FROM employees
  WHERE job_id = :NEW.job_id;
  IF :NEW.salary < v_minsalary OR
    :NEW.salary > v_maxsalary THEN
    RAISE_APPLICATION_ERROR(-20505,'Out of range');
  END IF;
END;
```

```
UPDATE employees
  SET salary = 3400
 WHERE last_name = 'Davies';
```

ORA-04091: table USVA_TEST_SQL01_T01_EMPLOYEES is

mutating, trigger/function may not see it

ORA-06512: at "USVA_TEST_SQL01_T01.CHECK_SALARY", line 5

ORA-04088: error during execution of trigger
'USVA_TEST_SQL01_T01.CHECK_SALARY'.

WHERE last_name = 'Davies';