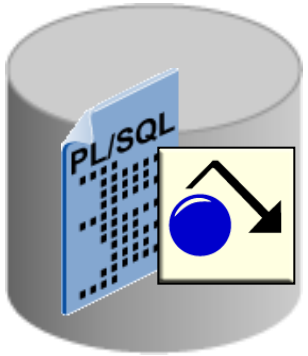
Triggers

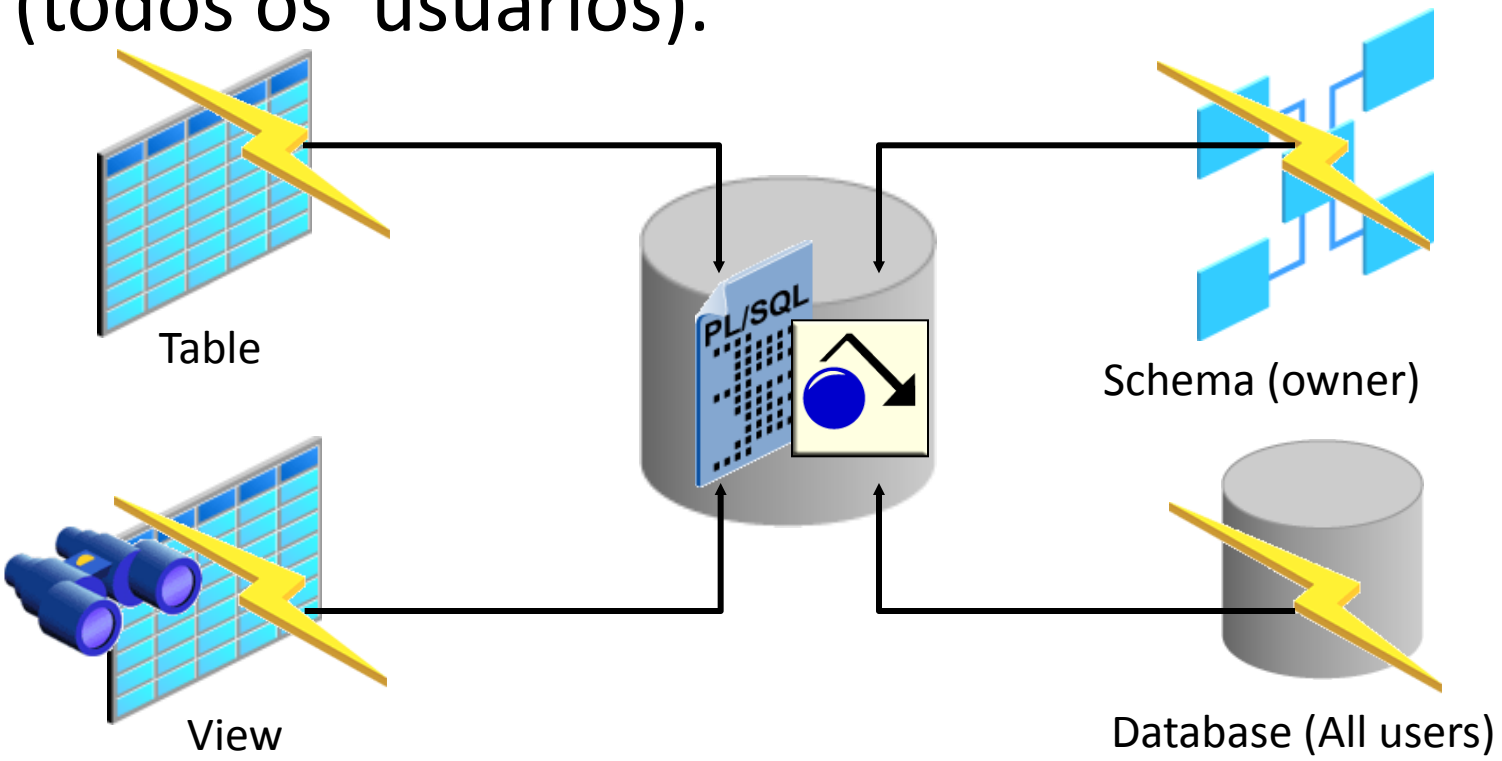
O que são Triggers?

- Um trigger é um bloco PL/SQL que é armazenado no BD e executado (disparado) em resposta a um evento especificado.
- O SGBD Oracle executa automaticamente um trigger quando uma condição especificada ocorre.



Definindo Triggers

- Um trigger pode ser definido sobre a tabela, view, esquema (schema owner), ou database (todos os usuários).

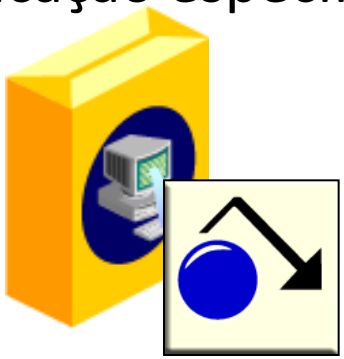


Tipos de Eventos de Trigger

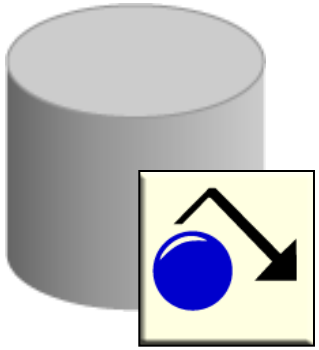
- Você pode escrever triggers que disparam toda vez que uma das seguintes operações ocorre no BD:
  - Um comando de manipulação do BD (DML) (`DELETE`, `INSERT`, or `UPDATE`).
  - Uma comando de definição de BD (DDL) (`CREATE`, `ALTER`, or `DROP`).
  - Uma operação de BD tal como `SERVERERROR`, `LOGON`, `LOGOFF`, `STARTUP`, or `SHUTDOWN`.

Triggers de Aplicação e de BD

- Triggers de BD (tratados aqui):
  - Dispara toda vez que um evento DML, um DLL, ou do sistema ocorre sobre um esquema ou BD
- Triggers de Aplicação:
  - Dispara toda vez que um evento ocorre dentro de uma aplicação específica



Trigger de Aplicação



Trigger de BD

Tipos de Eventos de Trigger e Corpo

- Um tipo de evento de trigger determina qual comando DML causa que o trigger execute. Os eventos possíveis são:
  - `INSERT`
  - `UPDATE [OF column]`
  - `DELETE`
- Um corpo de trigger determina qual ação é executada e é um bloco PL/SQL ou uma chamada a um procedimento.

Criando Triggers DML Usando o Comando CREATE TRIGGER

```
CREATE [OR REPLACE] TRIGGER trigger_name
timing -- when to fire the trigger
event1 [OR event2 OR event3]
ON object_name
[REFERENCING OLD AS old | NEW AS new]
FOR EACH ROW -- default is statement level trigger
WHEN (condition)]]
DECLARE]
BEGIN
... trigger_body -- executable statements
[EXCEPTION . . .]
END [trigger_name];
```

```
timing =  BEFORE | AFTER | INSTEAD OF
```

```
event = INSERT | DELETE | UPDATE | UPDATE OF column_list
```

Especificando o tempo do disparo Trigger (Timing)

- Você pode especificar o tempo de execução da ação do trigger como antes ou depois o evento associado:
  - `BEFORE`: Executa o corpo do trigger antes do disparo do evento LMD sobre uma tabela.
  - `AFTER`: Executa o corpo do trigger depois do disparo do evento LMD sobre uma tabela..
  - `INSTEAD OF`: Executa o corpo do trigger no lugar do comando de disparo. Isto é usado para views que não são de outro modo modificáveis

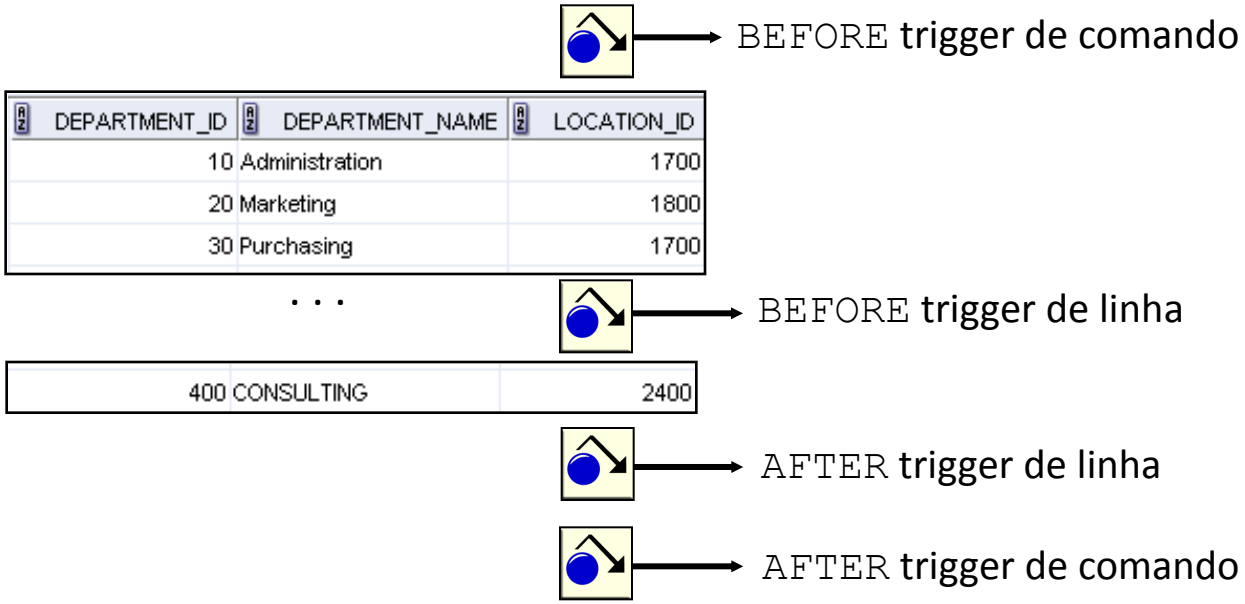
Triggers do nível de Comando versus Triggers do nível de linha

Triggers do nível de comando	Triggers do nível de linha
É o default quando criamos o trigger	Usa a cláusula `FOR EACH ROW` quando se cria um trigger.
Dispara uma vez para o evento do trigger	Dispara uma vez para cada linha afetada pelo evento pelo evento de disparo
Dispara uma vez mesmo que se nenhuma linha foi afetada	Não dispara se o evento de disparo não afeta nenhuma linha

Sequência de disparo do Trigger: Manipulação de uma só linha

- Use a seguinte sequência de disparo para um trigger sobre uma tabela quando uma só linha é manipulada:

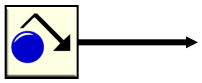
```
INSERT INTO departments
   (deptno,dname, loc)
VALUES (400, 'CONSULTING', 2400);
```



Sequência de disparo do Trigger: : Manipulação Multilinha

Use a seguinte sequência para um trigger sobre uma tabela quando várias linhas são manipuladas:

```
UPDATE emp
  SET sal = sal * 1.1
  WHERE deptno = 30;
```



BEFORE trigger de comando

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
114	Raphaely	30
115	Khoo	30
116	Baida	30
117	Tobias	30
118	Himuro	30
119	Colmenares	30

BEFORE trigger de linha

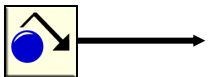
AFTER trigger de linha

. . .

BEFORE trigger de linha

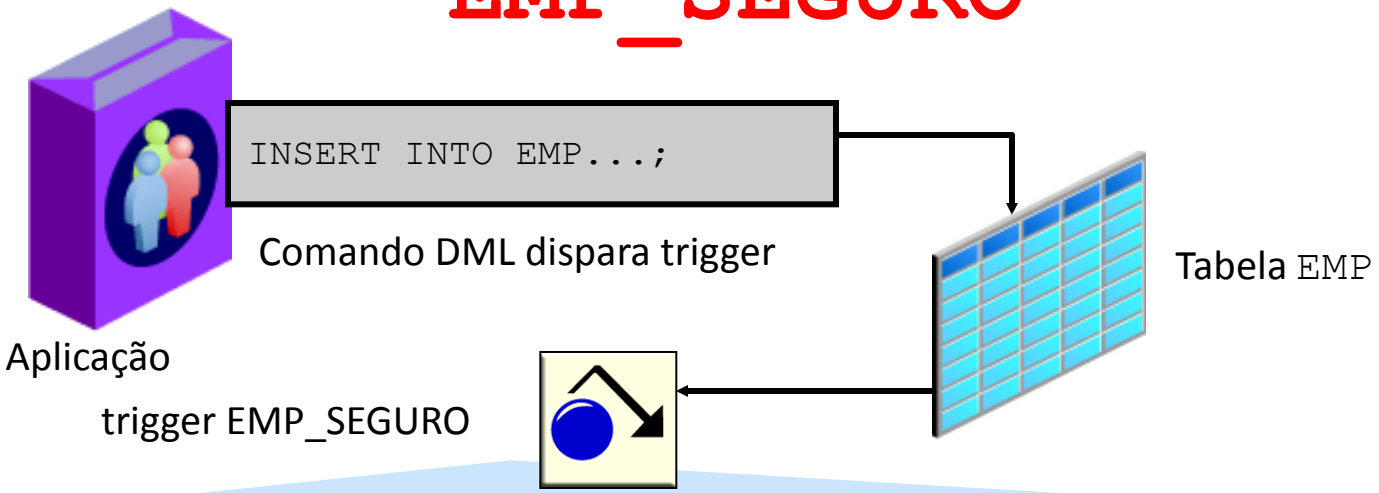
AFTER trigger de linha

. . .



AFTER trigger de comando

Criando um Exemplo de Trigger de Comando DML : EMP_SEGURO




```
CREATE OR REPLACE TRIGGER emp_seguro
BEFORE INSERT ON emp
  BEGIN
   IF (TO_CHAR(SYSDATE,'DAY') IN ('SABADO',‘DOMINGO')) OR
      (TO_CHAR(SYSDATE,'HH24:MI')
       NOT BETWEEN '08:00' AND '18:00') THEN
   RAISE_APPLICATION_ERROR(-20500, ‘Você pode inserir'
     ||' na tabela EMP somente durante '
     ||' horas normais de trabalho.');
   END IF;
  END;
```

Usando predicados condicionais

```
CREATE OR REPLACE TRIGGER emp_seguro BEFORE
INSERT OR UPDATE OR DELETE ON emp
  BEGIN
    IF (TO_CHAR(SYSDATE,'DAY') IN ('SABADO',‘DOMINGO')) OR
       (TO_CHAR(SYSDATE,'HH24')
        NOT BETWEEN '08' AND '18') THEN
      IF DELETING THEN RAISE_APPLICATION_ERROR(
        -20502,‘Você pode remover na tabela EMP'||
        ‘somente durante as horas normais de trabalho.');
      ELSIF INSERTING  THEN RAISE_APPLICATION_ERROR(
        -20500,'Você pode inserir na tabela EMP '||
        ‘somente durante as horas normais de trabalho.');
      ELSIF UPDATING  ('SAL') THEN
        RAISE_APPLICATION_ERROR(-20503, ‘Você pode atualizar '||
           ‘o SAL somente durante as horas normais de trabalho.');
      ELSE RAISE_APPLICATION_ERROR(-20504, ‘Você pode '||
        ' atualizar a tabela EMP somente durante'||
        ' normais de trabalho.');
      END IF;
    END IF;
  END;
```

Criando um trigger de linha DML

```
CREATE OR REPLACE TRIGGER salario_restrito
BEFORE INSERT OR UPDATE OF sal ON empl
FOR EACH ROW
BEGIN
  IF NOT (:NEW.job IN ('MANAGER', 'PRESIDENT'))
     AND :NEW.sal > 15000 THEN
    RAISE_APPLICATION_ERROR (-20202,
      'Empregado não pode ganhar mais de $15,000.');
  END IF;
END;
```

```
UPDATE emp
SET sal = 15500
WHERE ename = 'RUSSELL';
```

```
Error starting at line 1 in command:
UPDATE employees
SET salary = 15500
WHERE last_name = 'Russell'
Error report:
SQL Error: ORA-20202: Employee cannot earn more than $15,000.
ORA-06512: at "ORA62.RESTRICT_SALARY", line 4
ORA-04088: error during execution of trigger 'ORA62.RESTRICT_SALARY'
```

Usando qualificadores OLD e NEW

- Quando um trigger de nível de linha dispara, o motor de execução PL/SQL cria e popula duas estruturas de dados:
  - `OLD`: Armazena os valores originais do registro processado pelo trigger
  - `NEW`: Contem os valores novos
- `NEW` e `OLD` têm a mesma estrutura, como um registro declarado usando `%ROWTYPE` sobre a tabela para a qual o trigger esta ligado.

Operações de Dados	Valor Old	Valor New
`INSERT`	`NULL`	Valor Inserido
`UPDATE`	Valor antes da atualização	Valor após a atualização
`DELETE`	Valor antes da remoção	`NULL`

Usando os Qualificadores OLD e NEW: Exemplo

```
CREATE TABLE audit_emp (
  user_name      VARCHAR2(30),
  time_stamp     date,
  id             NUMBER(6),
  old_last_name VARCHAR2(25),
  new_last_name VARCHAR2(25),
  old_title      VARCHAR2(10),
  new_title      VARCHAR2(10),
  old_salary     NUMBER(8,2),
  new_salary     NUMBER(8,2) )
/
CREATE OR REPLACE TRIGGER audit_emp_valores
AFTER DELETE OR INSERT OR UPDATE ON emp
FOR EACH ROW
BEGIN
  INSERT INTO audit_emp(user_name, time_stamp, id,
    old_last_name, new_last_name, old_title,
    new_title, old_salary, new_salary)
  VALUES (USER, SYSDATE, :OLD.empno,
    :OLD.ename, :NEW.ename, :OLD.job,
    :NEW.job, :OLD.sal, :NEW.sal);
END;
```

Usando os Qualificadores OLD e NEW: Exemplo

```
INSERT INTO emp (empno, ename, job, sal, hire_date)
VALUES (999, 'Temp emp', 'SA_REP', 6000, TRUNC(SYSDATE))
/
UPDATE emp
 SET sal = 7000, ename = 'SMITH'
 WHERE empno = 999
/
SELECT *
FROM  audit_emp;
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

	USER_NAME	TIME_STAMP	ID	OLD_LAST_NAME	NEW_LAST_NAME	OLD_TITLE	NEW_TITLE	OLD_SALARY	NEW_SALARY
1	ORA61	04-JUN-09	(null)	(null)	Temp emp	(null)	SA_REP	(null)	6000
2	ORA61	04-JUN-09	999	Temp emp	Smith	SA_REP	SA_REP	6000	7000

Usando a Cláusula WHEN para disparar um Trigger de linha baseado numa Condição

```
CREATE OR REPLACE TRIGGER derive_commissao
BEFORE INSERT OR UPDATE OF sal ON emp
FOR EACH ROW
WHEN (NEW.job = 'SA_REP')
BEGIN
 IF INSERTING THEN
    :NEW.comm := 0;
 ELSIF :OLD.comm IS NULL THEN
    :NEW.comm := 0;
 ELSE
    :NEW.comm := :OLD.comm+0.05;
 END IF;
END;
/
```

Usando a Cláusula WHEN para disparar um Trigger de linha baseado numa Condição

Não é prefixada com ":" , porque está fora do bloco PL/SQL

```
CREATE OR                                    ssao
BEFORE INSE                                  p
FOR EACH RO
WHEN (NEW.job = 'SA_REP')
BEGIN
 IF INSERTING THEN
   :NEW.comm := 0;
 ELSIF :OLD.comm IS NULL THEN
   :NEW.comm := 0;
 ELSE
   :NEW.comm := :OLD.comm+0.05;
 END IF;
END;
/
```

Resumo do Modelo de Execução do Trigger

1. Executar todos os triggers `BEFORE STATEMENT`.
2. Laço *for each row* afetado pelo comando SQL:
   a. Executar todos os triggers `BEFORE ROW` para essa linha.
   b. Executar os comandos DML statement e desenvolver a verificação de restrição de integridade para essa linha.
   c. Executar todos os triggers `AFTER ROW` para essa linha.
3. Executar todos os triggers `AFTER STATEMENT`.

Exemplos -BD simplificado da Empresa

EMPREGADO

NOME	SSN	SALARIO	DNO	SUPERVISOR_SSN

DEPARTAMENTO

DNOME	DNO	TOTAL_SAL	GERENTE_SSN

```
Create or Replace Trigger Manut_Total_Sal
After insert or delete or update of sal or update of deptno on Emp
For Each Row
Begin
    if inserting and (:new.deptno  is not null) then
        update dept  set  total_sal = total_sal + :new.sal
        where deptno = :new.deptno;
    elsif  deleting and (:old.deptno is not null) then
        update dept  set  total_sal = total_sal - :old.sal
        where deptno = :old.deptno;
    elsif  updating (‘SAL’) then
        update dept  set  total_sal = total_sal + :new.sal - :old.sal
        where deptno = :new.deptno;
    elsif  updating(‘DEPTNO’) then
        Case
         When (:new.deptno is not null) and  (:old.deptno is not null) then
              update dept  set  total_sal = total_sal + :new.sal
              where deptno = :new.deptno;
              update dept  set  total_sal = total_sal - :old.sal
              where deptno = :old.deptno;
         When (:new.deptno is not null) and  (:old.deptno is null) then
               update dept  set  total_sal = total_sal + :new.sal
               where deptno = :new.deptno;
         When (:new.deptno is null) and  (:old.deptno is not null) then
                update dept  set  total_sal = total_sal - :old.sal
                where deptno = :old.deptno;
         Else DBMS_OUTPUT.PUT_LINE(‘Sem atalização’);
         End Case;
    End  if;
End;
```

Sintaxe de Oracle

<trigger> ::= **CREATE TRIGGER** <nome gatilho>
(**AFTER** | **BEFORE**) <eventos ativadores> **ON** <nome da tabela>
[**FOR EACH ROW**]
[**WHEN** <condicao>]
<acoes disparadas> ;

<eventos ativadores>::=<evento ativador> {**OR** <evento ativador>}

<acao disparada>::=**INSERT** | **DELETE** | **UPDATE** [**OF** <nome coluna> {, <nome coluna>}]

<acao disparada> ::=<PL/SQL block>