TRIGGERS

O que são?

É o conjunto de ações previamente programadas que são disparados no banco quando um determinado evento de alteração de registros ocorre.

Ou ainda, é uma resposta disparada por um evento dentro do banco de dados.

Vale destacar que ao manipularmos registros podemos usar 4 comandos DMLs, select, update, delete e insert, onde todos, exceto o select, realizam mudanças no conteúdo da tabela.

Podemos implementar uma rotina (conjunto de ações) complementar aos eventos de alteração dos registros, quer sejam por razões de segurança (Auditoria), quer sejam por razões de consistência de registros (criar regras de entradas).

Embora todos os triggers sejam iguais em sua natureza, teremos algumas particularidades quanto ao seu escopo (ROW LEVEL ou STATEMENT), quanto ao objeto afetado, quanto ao momento de sua execução (BEFORE ou AFTER), ou ainda quanto a ação que o originou (INSERT, UPDATE e DELETE).

Quanto ao ESCOPO

Row Level - são disparados para cada linha afetada pela transação, de tal modo que se um comando afete 5 linhas de uma dada tabela ele será disparado 5
vezes, uma para cada linha afetada.

Geralamente são usados para:

1 - validarem os valores que estão sendo processados, garantindo a consistência dos registros.

2 - auditoria na forma de geração de logs.

Statement - esse tipo de gatilho só é disparado uma única vez para o comando, independente do número de registros alterados. Dessa maneira, um comando que
altere 5000 linhas de uma tabela somente executará o trigger uma vez.

Quanto ao OBJETO AFETADO

Column Trigger - nesse tipo de trigger a ação somente será executada se a referrida coluna que contém o trigger for alterada. Ou seja, consideremos uma
tabela TAB_A com os campos CAMPO1, CAMPO2 e CAMPO3 e essa tabela tenha um trigger do tipo COLUMN para o campo CAMPO1. Nessa situação a trigger somente será

disparada quando o campo CAMPO1 for alterado, o mesmo NÃO acontecerá para os demais campos.

Table Trigger - nessa modalidade o trigger é disparado para qualquer alteração de qualquer coluna da tabela, ou seja o trigger está vinculado a tabela e não
a uma coluna específica.

Quanto ao MOMENTO
Before - indicam que o trigger será disparado antes do início da ação e são usados para a valiação dos dados de entrada, preenchimento de valores
obrigatórios e impedir ações não previstas ou não permitidas.

After - nesse caso o trigger será alterado após a realização da ação modificadora, ou seja, nesse caso o banco já foi alterado então normalmente são usados
para auditoria dos eventos ocorridos em uma ou mais tabelas.

TRIGGERS

Um pouco de prática....

1 - vamos criar a tabela tb_nota_concurso como mostrado abaixo:

```
create table tb_nota_concurso
( cod_candidato number(4),
 nota_candidato number(5,2));
```

2 - agora vamos criar a trigger para garantir que os valores de entrada estejam entre 0 e 100.

```
CREATE OR REPLACE TRIGGER tg_verifica_nota
BEFORE insert or update
ON tb_nota_concurso
REFERENCING NEW AS NEW
FOR EACH ROW
BEGIN
IF :new.nota_candidato not BETWEEN 0 and 100
THEN raise_application_error(-20001, 'A nota deve estar entre 0 e 100');
END IF;
END;
/
```

3 - vamos inserir o registro abaixo:

```
INSERT INTO tb_nota_concurso values (1, 67.98);
```

4 - o que aconteceu? verifique se o registro foi inserido.

5 - agora vamos tentar inserir o registro:

```
INSERT INTO tb_nota_concurso values (2, 110);
```

Como o valor é maior que o limite especificado foi dado a mensagem de erro e o registro não foi inserido. Verifique.

6 - façamos outro comando, vamos alterar a nota do registro de código 1 para -2.

```
update tb_nota_concurso set nota_candidato = -2 where cod_candidato = 1;
```

O mesmo erro acontece.

TRIGGERS

7 - vamos criar outra tabela tb_dedo_duro

```
create table tb_dedo_duro
( data date,
 usuario varchar2(20),
 cod_candidato number(4),
 nota_candidato number(5,2));
```

8 - vamos criar outro trigger para a tabela tb_nota_concurso.

```
CREATE OR REPLACE TRIGGER tg_registra_alteracao
AFTER update
OF nota_candidato ON tb_nota_concurso
REFERENCING OLD AS OLD
FOR EACH ROW
BEGIN
INSERT INTO tb_dedo_duro values (sysdate, user, :old.cod_candidato,
:old.nota_candidato);
END;
/
```

9 - vamos alterar a nota do registro de código 1 para 99.

```
update tb_nota_concurso set nota_candidato = 99 where cod_candidato = 1;
```

10 - vejamos se a alteração foi registrada na tabela tb_dedo_duro.

11 - vamos inserir um novo registro.

```
INSERT INTO tb_nota_concurso values (2, 11);
```

12 - verifique se a alteração foi registrada na tabela tb_dedo_duro.

```
select * from tb_dedo_duro;
```

13 - Por que não?

14 - vamos alterar o código do registro 2 para 3.

```
update tb_nota_concurso set cod_candidato = 3 where cod_candidato = 2;
```

15 - agora foi registrado?

```
select * from tb_dedo_duro;
```

16 - Por que não?

Operadores OLD e NEW

Esses operadores funcionam como variáveis que recebem os valores contidos no campo da tabela em momentos distintos onde:

:OLD.CAMPO - contém o valor anterior a alteração, ou seja o valor velho.
:NEW.CAMPO - contém o valor posterior a alteração , ou seja o valor novo.

Na prática anterior tivemos

```
CREATE OR REPLACE TRIGGER tg_verifica_nota
...
IF :new.nota_candidato not BETWEEN 0 and 100
```

o comando acima testa se o NOVO valor :new.nota_candidato está entre 0 e 100.

Já a prática

```
CREATE OR REPLACE TRIGGER tg_registra_alteracao
...
INSERT INTO tb_dedo_duro values (sysdate, user, :old.cod_candidato,
:old.nota_candidato);
```

pega o valor anterior/VELHO que existia na tabela antes da alteração e grava na outra tabela.

Deve-se reparar que nos nossos exemplos foram usados as cláusulas

```
REFERENCING NEW AS NEW
REFERENCING OLD AS OLD
```

uma de cada vez, isso porquê trabalhamos com uma de cada vez. Mas é possível, e até mais comum, usar em um mesmo trigger valores velhos e novos ao mesmo tempo, logo a cláusula fiacaria assim

```
REFERENCING NEW AS NEW OLD AS OLD
```

Cláusula WHEN

essa opção limita a condição de disparo da trigger, ou seja determina em que condições ela será disparada.

Exemplo:

```
CREATE OR REPLACE TRIGGER tg_registra_alteracao
AFTER update
OF nota_candidato ON tb_nota_concurso
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
WHEN (NEW.nota_candidato > OLD.nota_candidato)
BEGIN
INSERT INTO tb_dedo_duro values (sysdate, user, :old.cod_candidato,
:old.nota_candidato);
END;
/
```

Alteramos a trigger tg_registra_alteracao para ela disparar somente quando a nova nota for maior que a nota anterior.

Mais prática ...

1 - altere a nota do candidato de codigo 1 para 80.

```
update tb_nota_concurso set nota_candidato = 80 where cod_candidato = 1;
```

2 - foi registrado na tabela tb_dedo_duro?

```
select * from tb_dedo_duro;
```

3 - agora altere a mesma nota para 60.

```
update tb_nota_concurso set nota_candidato = 60 where cod_candidato = 1;
```

4 - foi registrado na tabela tb_dedo_duro?

```
select * from tb_dedo_duro;
```

5 - por que não?