**Triggers (Gatilhos)**

Os exemplos utilizados ao longo do texto, salvo menção em contrário, foram adaptados de GROFF, J. R., WEINBERG, P. N. e OPPEL, A. J. **SQL: The Complete Reference**, relacionado nas referências bibliográficas.Os *scripts* para criação e população das tabelas encontram-se no material de apoio.

* **O QUE SÃO *Triggers* (Gatilhos)**

*Triggers* são programas armazenados que são disparados (daí o nome) quando determinados eventos ocorrem. Cinco tipos de eventos podem disparar um *trigger*:

●        Execução de comandos DML (INSERT, UPDATE e DELETE) em tabelas. São chamados de gatilhos DML;

●        Execução de comandos DDL (CREATE, ALTER, DROP). São chamados de gatilhos DML;

●        Certos estados do SGBDR: inicialização (*start up*), desligamento (*shut down*), estabelecimento de conexão (*log in*), encerramento de conexão (*log out*) e ocorrência de erros;

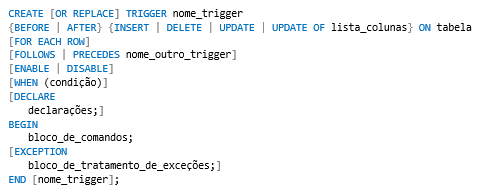
●        Execução de comandos DML (INSERT, UPDATE e DELETE) em *views*;

●        Suspensão de execução de comandos, devido a esgotamento dos recursos autorizados para o usuário (espaço em disco, por exemplo).

                No contexto de aplicações, *triggers* são muito úteis para garantir que regras de negócio estejam sendo obedecidas pelas aplicações.

* ***TRIGGERS*DML**

*Triggers*DML são, de longe, os mais usados por desenvolvedores. Eles estão associados a tabelas e são disparados sempre que um comando DML é executado na respectiva tabela. *Triggers* podem ser disparados antes ou depois da execução do comando. Também podem ser disparados uma vez para cada comando (*triggers* no nível de comando ou *statement-level triggers*)ou a cada registro inserido/alterado/excluído (*triggers* no nível de registro ou *row-level triggers*). A definição de um *trigger* tem a seguinte forma geral:



●        O comando CREATE TRIGGER cria um novo *trigger* com nome *nome\_trigger*. A cláusula opcional OR REPLACE substitui o *trigger*, caso ele exista;

●        As cláusulas BEFORE e AFTER definem se o *trigger* é disparado antes ou depois o comando DML;

●        As cláusulas INSERT, DELETE e UPDATE definem que comando DML causará o disparo. UPDATE ON deve ser utilizado no lugar de UPDATE caso se deseje que o disparo ocorra quando determinadas colunas forem alteradas (separadas por vírgulas). É possível definir mais de uma cláusula utilizando o operador OR;

●        A cláusula FOR EACH ROW define um *trigger* no nível de registro, disparando-o para cada linha alterada. Deve-se omitir esta cláusula para *triggers* no nível de comando.

●        Vários *triggers* podem ser definidos para um mesmo comando DML. FOLLOWS e PRECEDES definem a ordem de disparo;

●        A cláusulas ENABLE e DISABLE determinam se o *trigger* estará inicialmente habilitado (ativo) ou desabilitado. Apenas *triggers* habilitados são disparados. Se não informado, o estado inicial é habilitado (ENABLE);

●        WHEN define em que condições o *trigger*deve ser disparado, evitando que o bloco de comandos seja executado desnecessariamente. A condição deve vir entre parênteses;

●        As cláusulas DECLARE, BEGIN e EXCEPTION têm a mesma função que em *procedures* e funções. Opcionalmente, pode-se incluir o nome do *trigger* após o END.

                O estado de triggersDML pode ser alterado utilizando-se o comando ALTER TRIGGER:

https://paperx-dex-assets.s3.sa-east-1.amazonaws.com/images/1679498935857-mcso5Xbvi4.png

O comando DROP TRIGGER nome\_trigger exclui um trigger existente.

* **OS PSEUDOREGISTROS OLD E NEW**

*Triggers*DML são disparados quando são feitas alterações na tabela a que estão vinculados. PL/SQL oferece duas estruturas, OLD e NEW, que são populadas imediatamente antes do disparo do *trigger*. OLD possui os valores (colunas) do registro afetado antes da execução do comando DML, enquanto NEW possui os valores que o registro afetado terá após a execução do comando DML.

                Dependendo do comando DML que disparou o *trigger*, as seguintes situações podem ocorrer:

●        INSERT: apenas a estrutura NEW possui valores válidos (não havia valores anteriores à inserção de um novo registro);

●        DELETE: apenas a estrutura OLD possui valores válidos (não há novos valores após a exclusão de um registro);

●        UPDATE: ambas as estruturas OLD e NEW possuem valores válidos.

Alguns cuidados com o uso destas estruturas:

●    Deve-se colocar ‘:’ antes de referências a OLD e NEW no bloco de comandos apenas[[1]](https://aulas.descomplica.com.br/graduacao/engenharia-da-computacao/turma/desenvolvimento-de-aplicacoes-de-bancos-de-dados-3ontm/aula/triggers-gatilhos-agyep" \l "_ftn1" \t "_blank);

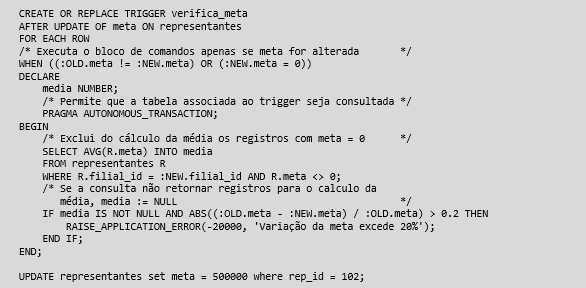
●    OLD e NEW só são criados em *triggers* no nível de linha. Referências a eles em *triggers* no nível de comando causarão erro.

●    Não é possível modificar valores na estrutura OLD, mas isto é possível na estrutura NEW;

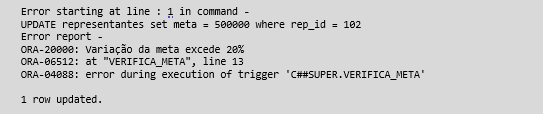
      De forma geral, não é permitido que *triggers* no nível de linha consultem ou alterem a tabela a qual estão vinculados. Esta situação é apontada com o erro de compilação *ORA-04091 table xxx is mutating*. Para que sejam possíveis consultas, deve-se utilizar a diretiva PRAGMA AUTONOMOUS\_TRANSACTION*.* *Triggers* no nível de comando são livres para consultar e alterar as respectivas tabelas associadas.

     No exemplo a seguir o *trigger* *VERIFICA\_META* é criado. Ele é disparado antes de alterações na coluna *meta* da tabela *REPRESENTANTES*. A cláusula WHEN habilita a execução apenas se a coluna tiver seu valor alterado. Este *trigger* limita alterações na meta de um represente a ±20% da média das metas dos representantes vinculados à mesma filial. Caso a meta seja 0 (o representante é novo e ainda não tem meta), permite a alteração para qualquer valor.

[[1]](https://aulas.descomplica.com.br/graduacao/engenharia-da-computacao/turma/desenvolvimento-de-aplicacoes-de-bancos-de-dados-3ontm/aula/triggers-gatilhos-agyep" \l "_ftnref1" \t "_blank)OLD e NEW são estruturas do tipo RECORD, declaradas como *bind variables*(variáveis de ligação, em tradução literal). Para acessar *bind variables* dentro do bloco de comandos, deve-se utilizar o ‘:’ como prefixo, porém isto já não é necessário (na verdade, não é permitido) quando a referência ocorre na cláusula WHEN.

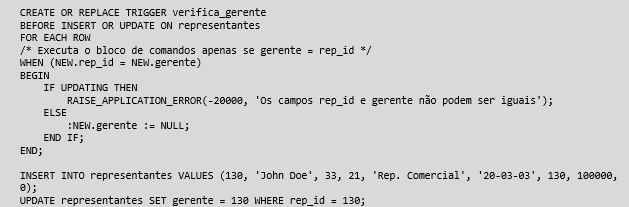


                O resultado apresentado após a execução dos comandos UPDATE é:



* **IDENTIFICANDO O EVENTO QUE DISPAROU O *TRIGGER***

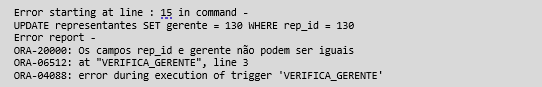
                A linguagem PL/SQL possui um conjunto de funções que permitem identificar o comando DML que disparou o *trigger*. São as funções INSERTING, DELETING e UPDATING. Elas permitem que as ações certas sejam tomadas em função do tipo de alteração que está sendo efetuada na tabela. O *trigger* *VERIFICA\_GERENTE*, mostrado a seguir, verifica se os campos *id\_rep* e *gerente* são iguais em inclusões e alterações de registros na tabela *REPRESENTANTES*. Se for um comando INSERT, a operação é rejeitada com uma mensagem de erro.  Para alterações, o campo *gerente*recebe o valor NULL.



                O registro incluído tem os seguintes valores:

https://paperx-dex-assets.s3.sa-east-1.amazonaws.com/images/1679497714602-i8jFa5cFbO.png

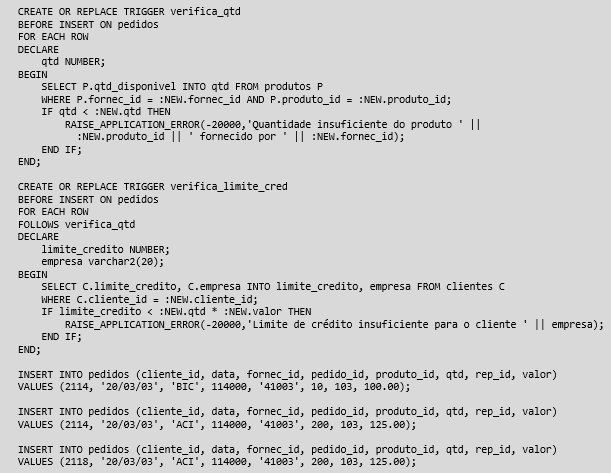
                Após a execução do comando UPDATE, o resultado é:



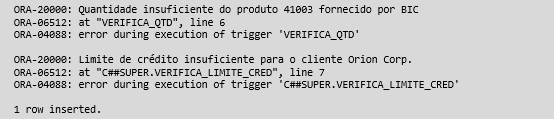
* **DEFININDO A ORDEM DE DISPARO DE *TRIGGERS***

                É possível definir mais de um *trigger* para uma mesma condição de disparo. Para garantir a ordem em que serão disparados, utilizam-se as cláusulas FOLLOWS e PRECEDES. Estas cláusulas fazem com que o *trigger* em questão seja disparado após/antes o *trigger* cujo identificador aparece logo em seguida a elas.

                No exemplo a seguir, são definidos dois *triggers* para a tabela *PEDIDOS.* O primeiro, *VERIFICA\_QTD*, não permite que sejam colocados pedidos com quantidades superiores aos estoques dos respectivos produtos. O segundo, *VERIFICA\_LIMITE\_CRED*, impede que sejam colocados pedidos cujo valor total exceda o limite de crédito do respectivo cliente. Você deseja que *VERIFICA\_QTD* seja disparado antes de *VERIFICA\_LIMITE\_CRED.*



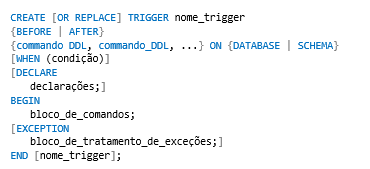
Após a execução dos três comandos de inserção, o resultado é:



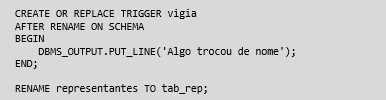
* ***TRIGGERS*DDL**

*Triggers*DDL são muito similares a seus primos DML, porém são disparados a partir da execução de comandos DDL. Outra diferença importante diz respeito ao escopo do *trigger*. Enquanto *triggers*DML estão associados a uma tabela específica, *triggers*DDL são associados ao esquema em que foram criados ou todo o banco de dados. Sua forma geral é:[[1]](https://aulas.descomplica.com.br/graduacao/engenharia-da-computacao/turma/desenvolvimento-de-aplicacoes-de-bancos-de-dados-3ontm/aula/triggers-gatilhos-agyep#_ftn1)

[[1]](https://aulas.descomplica.com.br/graduacao/engenharia-da-computacao/turma/desenvolvimento-de-aplicacoes-de-bancos-de-dados-3ontm/aula/triggers-gatilhos-agyep#_ftnref1)Qualquer comando DDL, como GRANT, REVOKE e AUDIT, pode ser usado com evento de disparo. Os mais comuns são CREATE, ALTER e DROP. Para definir a condição de disparo para qualquer comando DDL, utilize o identificador DDL.



As demais cláusulas funcionam como nos t*riggers*DML. Exemplo:

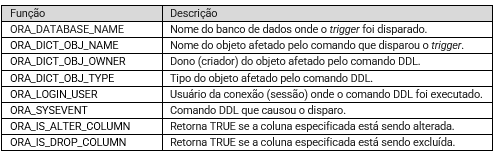


                O resultado, após a execução do comando RENAME, é:

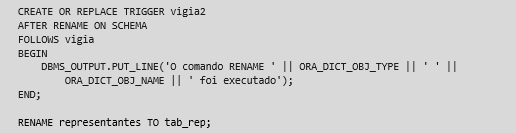
https://paperx-dex-assets.s3.sa-east-1.amazonaws.com/images/1679498486931-QQnIPIH6oS.png

Como *triggers* DDL são vinculados a um esquema particular ou a todo o banco de dados, é importante saber, além do comando DDL causou o disparo (condições de disparo DDL ou com o operador OR), que objeto foi afetado pelo comando. Para tal, há um conjunto de funções úteis, chamadas de funções de atributos de eventos (*event atribute functions*). As mais comuns são mostradas Tabela.

**O conjunto completo de funções pode ser encontrado em**[**https://docs.oracle.com/cd/B10501\_01/appdev.920/a96590/adg14evt.htm**](https://docs.oracle.com/cd/B10501_01/appdev.920/a96590/adg14evt.htm)**.**



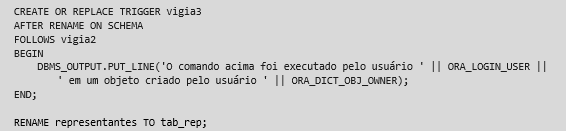
                A seguir, são apresentados diversos exemplos do uso destas funções. As funções ORA\_DICT\_OBJ\_TYPE e ORA\_DICT\_OBJ\_NAME retornam o tipo e o nome do objeto afetado, respectivamente.



                O resultado, após a execução do comando RENAME, é:

https://paperx-dex-assets.s3.sa-east-1.amazonaws.com/images/1679498588722-pvC3MpvnKQ.png

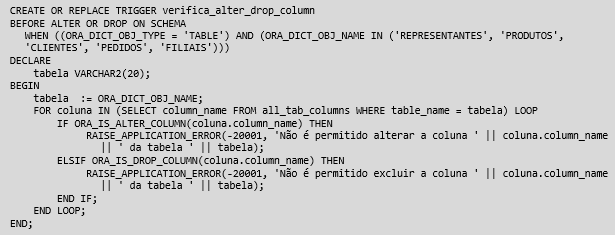
                As funções ORA\_LOGIN\_USER e ORA\_DICT\_OBJ\_OWNER retornam o usuário da conexão (sessão) e o usuário que criou o objeto afetado, respectivamente.



                O resultado, após a execução do comando RENAME, é:

https://paperx-dex-assets.s3.sa-east-1.amazonaws.com/images/1679498623565-YhAW46d2qz.png

​Duas funções demandam maior atenção: ORA\_IS\_ALTER\_COLUMN e ORA\_IS\_DROP\_COLUMN. Elas permitem que seja feito o controle de exclusões e alterações em colunas de uma tabela. No exemplo a seguir, o trigger VERIFICA\_ALTER\_DROP\_COLUMN impede que as colunas das tabelas utilizadas nos exemplos sejam alteradas ou excluídas.​



​O comando FOR percorre todas as colunas da tabela afetada verificando se serão alteradas ou excluídas.