Triggers (Gatilhos)

PROFESSOR: VINICIUS CAMPOS

profvinicampos@gmail.com

Introdução

- Uma trigger é um tipo especial de procedimento armazenado, que é executado sempre que há uma tentativa de modificar os dados de uma tabela que é protegida por ele;
- Associados a uma tabela;
- Chamados Automaticamente;
- Não podem ser chamados diretamente;

Introdução

- Ao contrário dos procedimentos armazenados do sistema, os disparadores não podem ser chamados diretamente e não passam nem aceitam parâmetros;

- Armazenados dentro do banco de dados, onde podemos definir um "bloco" PL/SQL para que seja executado automaticamente pelo banco, assim toda vez que uma instrução SQL (evento DML) for aplicada para uma tabela específica ele irá executar um determinado evento automaticamente.

Para que serve uma Trigger?

- Uma *Trigger* dentro do seu banco será de uma utilidade enorme, imaginem que temos um banco de dados com 1.500 registros, onde os funcionários da empresa ganham por hora, e você tem que atualizar o banco de horas de cada funcionário por dia, alguns comandos poderiam fazer isso por nós, mas seria um pouco complicado e desgastante, então, por este motivo, criamos nossas *Triggers*, onde definimos o que deve ser mudado na tabela em um único arquivo e depois acionamos um único comando DML (Update) para ele, assim, ele irá atualizar os 1.500 registros de uma única só vez.

Benefícios das Triggers

- Prover verificações básicas de segurança sobre a manipulação dos dados;
- Prover auditoria sobre a manipulação dos dados;
- Melhorar a integridade dos dados;
- Assegurar que as operações de dados relacionais sejam executadas juntamente de forma implícita.

Comando Create Trigger

Sintaxe:

Create or replace trigger nome_da_trigger

before/after evento ON nome_da_tabela

Bloco PL/SQL.

Onde:

Evento: indica qual é a manipulação do dado que irá disparar a trigger (insert/update/delete)

Exemplo

```
1  CREATE OR REPLACE TRIGGER TGBONUS
2  BEFORE INSERT OR DELETE OR UPDATE ON BONUS
3  FOR EACH ROW
4  DECLARE
5     --DECLARACAO DE VARIAVEIS
6  BEGIN
7  IF NOT DELETING THEN
8     IF :NEW.SAL IS NULL THEN
9        RAISE_APPLICATION_ERROR(-20000,'Salario deve ser informado');
10    END IF;
11  END IF;
12 END;
```

Tabela 1. Elementos triggers

Componentes	Descrição	Valores
Tempo	Quando o trigger dispara em relação ao evento de acionamento (DML)	- BEFORE - AFTER
Evento de acionamento	Quais operações de manipulação de tabela (DML) disparam a trigger	- INSERT - UPDATE - DELETE
abrangência da trigger	Quantas vezes o corpo da trigger será executado	- de linha (for each row) - de instrução(*)
Corpo da trigger	Que ações serão executadas	Bloco PL/SQL

(*) Opção default

Trigger de Linha (For Each Row)

- Esta trigger dispara quando um evento de manipulação de dados afetando as linhas em uma tabela. Para se criar essa trigger, basta apenas acrescentar **FOR EACH ROW** no comando create trigger;

- Uma trigger de linha é disparada uma vez para cada linha afetada pela instrução DML.

Trigger de Linha (For Each Statement)

- Executado uma vez por comando SQL;
- Se não for especificado nenhum dos dois, o padrão é FOR EACH STATEMENT.

Exemplo

```
1   CREATE OR REPLACE TRIGGER TGBONUS
2   BEFORE INSERT OR DELETE OR UPDATE ON BONUS
3   FOR EACH ROW
4   DECLARE
5      --DECLARACAO DE VARIAVEIS
6   BEGIN
7   IF NOT DELETING THEN
8      IF :NEW.SAL IS NULL THEN
9         RAISE_APPLICATION_ERROR(-20000,'Salario deve ser informado');
10     END IF;
11   END IF;
12  END;
```

Momento (Antes/Depois)

Before – (Antes)

- Os Triggers do tipo BEFORE como podemos deduzir, disparam antes que a ação ocorra. Isso leva a entender que antes que uma ação de banco de dados ocorra a Trigger será disparado, o que pode fazer com que a ação nem venha a ocorrer. Um Trigger pode impedir que uma ação venha a ocorrer, portanto podemos usar uma Trigger deste tipo em situações como, por exemplo:
- Validação de dados;
- Carregamento de dados obrigatórios (datas, usuários, etc..);
- Impedimento de ações em horários não previstos;

Exemplo Before

```
CREATE OR REPLACE TRIGGER trg_pedidos_valor
BEFORE INSERT OR DELETE OR UPDATE
ON pedidos

FOR EACH ROW
BEGIN
IF :NEW.PEDI_VL_BRUT_CALC < 10000 THEN

raise_application_error(-20001,'O valor do pedido deve ser inferior a 10.000,00');
END IF;
END;
```

After (Depois)

After – (Depois)

- Os Triggers do tipo AFTER ocorrem depois que a ação tenha ocorrido, ou seja elas são disparadas depois, com isso NÃO podemos com esses tipos de Triggers fazer o que fazemos com Triggers do tipo BEFORE. Aqui a ação já ocorreu então o que podemos fazer com Triggers deste tipo é a auditoria.

Exemplo After

```
1  CREATE OR REPLACE TRIGGER trg_salario_aud
2  AFTER UPDATE
3  ON empregados
4
5  FOR EACH ROW
6  BEGIN
7  INSERT INTO log_salario
8  (codigo, salario_anterior, salario_atual,
9  data_alteracao, usuario )
10 VALUES (:NEW.codigo, :OLD.salario_anterior,
11 :NEW.salario_atual, SYSDATE, USER);
12 END;
```

Modificadores OLD e NEW

- utilizamos estes dois pseudo registros, eles servem para fazer as comparações das colunas velhas (**:old**) com as novas (**:new**), são muito utilizadas para fazer Update nas colunas;

- Podemos nos casos da Triggers de linha , fazer referência a valores contidos nas colunas e com isso podemos querer saber os valores antes da alteração e depois dos valores efetivamente alterados. Isso vale na ação de UPDATE, nos casos de INSERT e DELETE os valores de OLD (INSERT) e NEW (DELETE) são nulos. Estes modificadores podem ser usados APENAS em TRIGGERS. Não podemos usá-los em procedures, functions ou packages;

Tabela Old/New

Tabela 2 – Qualificadores

DML	:old	:new
INSERT	NULO	Valores Novos
DELETE	Valores antigos	NULO
UPDATE	Valores antigos	Valores Novos

Modificadores OLD e NEW

- Os valores são referenciados da seguinte forma :OLD.nomecoluna e :NEW.nomecoluna.

- Não importa se o Trigger for BEFORE ou AFTER os modificadores OLD e NEW não são afetados.

Exemplo

```
1  CREATE OR REPLACE TRIGGER aumento
2  BEFORE UPDATE OF sal ON emp
3  FOR EACH ROW
4  BEGIN
5  IF (:NEW.sal - :OLD.sal) < :OLD.sal * 0.025 THEN
6   RAISE_APPLICATION_ERROR (-20512, 'Favor corrigir indice');
7  END IF;
8  END;
9
10
```

Exemplo

```
1  CREATE OR REPLACE TRIGGER TGPRONFECR -- Cancelamento
2  BEFORE INSERT OR DELETE OR UPDATE ON TBPRONFECR
3  FOR EACH ROW
4  DECLARE
5     I_SITUACAO        NUMBER(2);
6  BEGIN
7     IF UPDATING OR INSERTING THEN
8        IF :NEW.CSTAT = 101 or :NEW.CSTAT = 218 or :NEW.CSTAT = 205 THEN
9        I_SITUACAO := 20;
10    ELSE
11       I_SITUACAO := 91;
12    END IF;
13       UPDATE TBPRONFEA Z
14       SET XMOTIVO_RECIBO_CANCEL    = :NEW.XMOTIVO,
15           CHNFE_CANCEL             = :NEW.CHNFE,
16           DHRECBTO__CANCEL         = :NEW.DHRECBTO,
17           NPROT_CANCEL             = :NEW.NPROT,
18           SITUACAO                 = i_situacao
19       WHERE Z.COD_EMPRESA = :NEW.COD_EMPRESA AND
20             Z.COD_FILIAL  = :NEW.COD_FILIAL AND
21             Z.ORIGEM      = :NEW.ORIGEM AND
22             Z.CODIGO      = :NEW.CODIGO AND
23             Z.SERIE       = :NEW.SERIE AND
24             Z.NUM_NOTA    = :NEW.NUM_NOTA;
25    END IF;
26 END;
```

Exemplo

```
1  CREATE OR REPLACE TRIGGER TGBONUS
2  BEFORE INSERT OR DELETE OR UPDATE ON BONUS
3  FOR EACH ROW
4  DECLARE
5     --DECLARACO DE VARIAVEIS
6  BEGIN
7  IF NOT DELETING THEN
8     IF :NEW.SAL IS NULL THEN
9        RAISE_APPLICATION_ERROR(-20000,'Salario deve ser informado');
10    END IF;
11 END IF;
12
13 IF NOT DELETING THEN
14    IF :NEW.SAL = :OLD.SAL THEN
15       RAISE_APPLICATION_ERROR(-20000,'Salario Salario novo igual o antigo');
16    END IF;
17 END IF;
18 END;
```

Cláusula WHEN

- Podemos restringir a ação da trigger segundo uma condição, onde a mesma será disparada apenas para as linhas que satisfaçam a condição prevista;

- Caso a Trigger tenha alguma condição para ser executada, podemos incluir uma cláusula chamada WHEN. Nesta colocamos as condições que a Trigger irá disparar. Caso precisemos tratar o valor de alguma coluna, usamos os modificadores OLD e NEW, mas nessa cláusula não colocaremos os : na frente, pois nesse caso ocorrerá erro.

Cláusula WHEN - Exemplo

```
CREATE TRIGGER scott.salary_check
    BEFORE
    INSERT OR UPDATE OF sal, job ON scott.emp
    FOR EACH ROW
    WHEN (new.job <> 'PRESIDENT')
       pl/sql_block
```

Cláusula WHEN - Exemplo

```
1  CREATE OR REPLACE TRIGGER TGEMPSAL
2  BEFORE INSERT OR UPDATE ON EMP
3  FOR EACH ROW
4  WHEN (NEW.SAL < 550)
5  BEGIN
6    :NEW.sal := 550;
7  END;
8
```

Cláusula WHEN - Exemplo

```
1  CREATE OR REPLACE TRIGGER audit_emp_values
2  BEFORE delete or insert or update ON emp
3  FOR EACH ROW
4  when (new.job= 'SALESMAN')
5  BEGIN
6    :NEW.COMM:= :OLD.COMM*(:NEW.SAL/:OLD.SAL);
7  END;
```

Chamando um procedimento

```
CREATE TRIGGER scott.salary_check
    BEFORE INSERT OR UPDATE OF sal, job ON scott.emp
    FOR EACH ROW
    WHEN (new.job <> 'PRESIDENT')
    CALL check_sal(:new.job, :new.sal, :new.ename);
```

Triggers Armazenadas

- Gerenciando triggers armazenadas dentro do banco de dados com comandos similares aos comandos para procedures:

Tarefa	Comando
Criar uma nova trigger	Create trigger
Modificar uma trigger existente	Create or replace trigger
Remover uma trigger	Drop trigger.

Triggers Armazenadas

- Diferente das procedures, pode-se desabilitar uma trigger quando for conveniente.

Sintaxe:

- ALTER TRIGGER nome_da_trigger DISABLE/ENABLE.

- ALTER TABLE nome_da_tabela DISABLE/ENABLE ALL TRIGGERS.

USER_TRIGGERS

COLUNA	DESCRIÇÃO
TRIGGER_NAME	O nome da trigger.
TRIGGER_TYPE	O momento em que a trigger será executada. (Before/After).
TRIGGERIMG_EVENT	O comando de manipulação de dado que causa a execução da trigger (INSERT, UPDATE OU DELETE).
TABLE_OWNER	O dono da tabela associado a trigger.
TABLE_NAME	Nome da tabela associado a trigger.
WHEN	Condição de restrição.
STATUS	Disponibilidade da trigger. (Enable/Disable).
TRIGGER_BODY	Texto do bloco PL/SQL.

Restrições – Não permitido

- Não podemos realizar os comandos COMMIT, ROLLBACK e SAVEPOINT em uma Trigger, mesmo que seja uma procedure executada em um Trigger;

- Não podemos fazer select na mesma tabela que sofre a ação de uma Trigger , pois isso pode provocar um erro chamado MUTANT TABLE. Mesmo porque se quisermos saber o valor de uma coluna do registro que está sendo tratado em um Trigger basta colarmos :new.nomecoluna ou :old.nomecoluna para termos respectivamente os valores atuais e anteriores a alteração;

- Triggers tornam as operações mais lentas, isso ocorre principalmente em casos de Triggers de linha. (For Each Row).

Exercícios

1) Criar um procedimento para inserir dados na tabela bônus deixando o campo JOB nulo. No momento da inserção disparar uma trigger que verifica esse campo (JOB), caso esteja nulo apresentar a seguinte mensagem **na trigger: (Campo, job preenchimento obrigatório).**

2) Criar um procedimento para: inserir dados na tabela emp exceto o campo COMM (deixar nulo). No momento da inserção disparar uma trigger que atualize o campo COMM da tabela emp para o mesmo valor do salário informado no insert acima.

3) Criar uma trigger validando o campo SAL da tabela EMP. Essa validação (trigger) não poderá permitir que o salário seja aumentado em 10%.

Bibliografia

Elmasri, Ramez
Sistemas de banco de dados/ Ramez Elmasri e Shamkant B. Navathe; revisor técnico Luis Ricardo de Figueiredo. –São Paulo: Pearson Addison Wesley, 2005.

Rob, Peter
Sistemas de banco de dados : projeto, implementação e gerenciamento / Peter Rob,

http://www.devmedia.com.br/pl-sql-functions-e-procedures/29882