Sistemas de Informação

GSI021 Banco de Dados 2

Programando com SQL Triggers

Prof. Humberto Razente

humberto.razente@ufu.br

Bloco B - sala 1B144

Trigger

- Gatilho ou Disparador

- É uma sub-rotina, semelhante a uma *stored procedure*, que tem como característica operacional ser executada automaticamente quando uma determinada ação (INSERT/UPDATE/DELETE) for realizada no banco de dados

Trigger

- Uma das maneiras mais práticas de implementar rotinas para garantir a integridade de dados ou de operações

- Principal diferença entre Trigger e Stored Procedure
  - Trigger é executado automaticamente
  - Stored Procedure precisa ser explicitamente invocada

Trigger

- Podem ser utilizados para gerenciar informações do banco de dados
  - Automatizar a geração de dados
  - Fazer a auditoria das modificações
  - Implantar as restrições complexas de integridade
  - Personalizar as autorizações complexas de segurança

Trigger

- Contém um bloco PL/SQL para executar determinadas tarefas
- Está associado a uma tabela ou uma visão

Trigger

- Restrições de uso:
  - Não executar os comandos COMMIT, ROLLBACK ou SAVEPOINT
  - O comando SELECT pode ser usado apenas com a cláusula INTO
  - Não pode ler ou modificar o conteúdo de uma tabela mutante
    - Tabela na qual o conteúdo está sendo alterado por um comando INSERT, DELETE ou UPDATE que ainda não foi concluído
  - No Oracle, o tamanho do trigger não pode exceder 32 Kbytes
    - procedures podem ser chamadas pelo trigger

Componentes de um Trigger

1. Comando SQL que aciona o trigger

- O disparo do trigger pode ser ocasionado pelo comando SQL ou por um evento do usuário
  - Em uma tabela, pelos comandos INSERT, UPDATE ou DELETE
  - Em um objeto de esquema, por meio dos comandos CREATE, ALTER ou DROP
  - No carregamento/shutdown do BD por uma mensagem de erro
  - No logon/logoff de um usuário

Componentes de um Trigger

2. Limitador de ação do trigger

- Representado pela cláusula WHEN
  - Especifica qual condição deve ser verdadeira para que o trigger seja disparado

Componentes de um Trigger

3. Ação executada pelo trigger
- É o bloco de comandos que é executado pelo trigger
  - bloco PL/SQL

Trigger no Oracle

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE | AFTER}
{INSERT | UPDATE | UPDATE OF column1
                   [, column2 [, column(n+1)]]
                 | DELETE}
ON table_name
[FOR EACH ROW]
[WHEN (logical_expression)]
[DECLARE]
  declaration_statements;
BEGIN
  execution_statements;
END [trigger_name];
```

Trigger

- Alteração
  - Recriar com o comando CREATE OR REPLACE
- Exclusão
  - DROP TRIGGER trigger_name ON table_name
- Disable
  - ALTER TRIGGER trigger_name DISABLE
- Enable
  - ALTER TRIGGER trigger_name ENABLE

Obtendo informações de um Trigger

- Dicionário de dados do Oracle
  - USER_TRIGGERS e ALL_TRIGGERS

Trigger - Exemplo

- Suponha que seja necessário auditar a alteração de preço quando o novo preço é inferior a 25% do preço antigo
- É necessário especificar uma condição do trigger para comparar o preço novo com o preço antigo

Trigger - Exemplo

```
CREATE TABLE products (
  product_id INTEGER  PRIMARY KEY,
  product_type_id INTEGER
    CONSTRAINT products_fk_product_types
    REFERENCES product_types(product_type_id),
  name VARCHAR2(30) NOT NULL,
  description VARCHAR2(50),
  price NUMBER(5, 2) );
```

Trigger - Exemplo

```
CREATE TABLE product_price_audit (
   product_id INTEGER
     CONSTRAINT price_audit_fk_products
     REFERENCES products(product_id),*
   old_price NUMBER(5, 2),
   new_price NUMBER(5, 2),
   userID varchar(20)
 );
```

* Chave estrangeira que referencia a tabela que contém os produtos

Trigger - Exemplo

```
CREATE TRIGGER before_product_price_update
BEFORE UPDATE OF price ON products
FOR EACH ROW
WHEN (new.price < old.price * 0.75)
BEGIN
   dbms_output.put_line('---------------');
   dbms_output.put_line('produto ' || :old.product_id || ' ' || :old.name);
   dbms_output.put_line(‘preço antigo ' || :old.price);
   dbms_output.put_line(‘preço novo ' || :new.price);
   dbms_output.put_line('A redução de preço é de mais de 25%');

   -- insert row into the product_price_audit table
   INSERT INTO product_price_audit
                     (product_id, old_price, new_price, userID)
          VALUES (:old.product_id, :old.price, :new.price, user);
END before_product_price_update;
```

Trigger - Exemplo

- BEFORE UPDATE OF price
  - Significa que o trigger é disparado antes da atualização de price
- FOR EACH ROW
  - Significa que isso é um **trigger em nível de tupla**, isto é, o código do trigger é executado uma vez para cada tupla atualizada
- A condição do trigger é (new.price < old.price * 0.75)
  - O trigger é disparado somente quando o novo preço é menor do que 75 % do preço antigo
- Os valores novos e antigos do atributo são acessados por meio dos apelidos :old e :new no trigger

Trigger - Exemplo

- Disparando o trigger
  - Para ver a saída do trigger é necessário rodar o comando (Oracle)
    - `SET SERVEROUTPUT ON`
  - Para disparar o trigger do exemplo anterior é necessário reduzir o preço de um produto em mais de 25%

```
UPDATE products
SET price = price * 0.7
WHERE product_id IN (5, 10);
```

Trigger - Exemplo

- Disparando o trigger
  - ```
    UPDATE products
    SET price = price * .7
    WHERE product_id IN (5, 10);
    ```
- Saída:

produto 10
preço antigo 15.99
preço novo 11.19
A redução é de mais de 25%
---
produto 5
preço antigo 49.99
preço novo 34.99
A redução é de mais de 25%
    ```

Trigger - Exemplo

- Disparando o trigger
  - Para ver o conteúdo da tabela que guarda as informações de auditoria

```
SELECT *
FROM product_price_audit
ORDER BY product_id;
```

PRODUCT_ID	OLD_PRICE	NEW_PRICE	USERID
5	49.99	34.99	JOHNDOE
10	15.99	11.19	JOHNDOE

Trigger – FOR EACH ROW

- Em triggers FOR EACH ROW
  - manipule os atributos envolvidos por meio de:

new

old

System or Database Event Triggers

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE | AFTER} database_event ON {database | schema}
[DECLARE]
  declaration_statements;
BEGIN
  execution_statements;
END [trigger_name];
```

- Eventos:
  - logon, logoff: login de usuários
  - startup, shutdown: banco de dados

Trigger em tabela mutante

```
CREATE TABLE mutant
( mutant_id      NUMBER
, mutant_name    VARCHAR2(20));

INSERT INTO mutant VALUES (mutant_s1.nextval,'Donatello');
INSERT INTO mutant VALUES (mutant_s1.nextval,'Leonardo');
INSERT INTO mutant VALUES (mutant_s1.nextval,'Michelangelo');
INSERT INTO mutant VALUES (mutant_s1.nextval,'Raphael');

CREATE OR REPLACE TRIGGER mutator
AFTER DELETE ON mutant
FOR EACH ROW
DECLARE
  rows NUMBER;
BEGIN
  SELECT COUNT(*) INTO rows FROM mutant;
  dbms_output.put_line('[rows] has '||rows||']');
END;
```

- Não é permitido consultar tabela mutante: restrição existe para não permitir que o *trigger* veja dados inconsistentes

Trigger em tabela mutante

```
CREATE OR REPLACE TRIGGER mutator
AFTER DELETE ON mutant
FOR EACH ROW
DECLARE
  rows NUMBER;
BEGIN
  SELECT COUNT(*) INTO rows FROM mutant;
  dbms_output.put_line('[rows] has '||rows||']');
END;
```

```
DELETE FROM mutant WHERE mutant_name = 'Michelangelo'

ERROR at line 1:
ORA-04091: table PLSQL.MUTANT is mutating, trigger/function may not see it
ORA-06512: at "PLSQL.MUTATOR", line 4
ORA-04088: error during execution of trigger 'PLSQL.MUTATOR'
```

- Não é permitido consultar tabela mutante: restrição existe para não permitir que o *trigger* veja dados inconsistentes

Cancelamento do evento pelo trigger

- Se uma exceção for disparada em um trigger, o comando que disparou o trigger é desfeito
  - para que ocorra o cancelamento, é preciso chamar a função RAISE_APPLICATION_ERROR
  - se houverem outros comandos executados na transação, eles ainda podem ser confirmados (COMMIT)

```
CREATE OR REPLACE TRIGGER GatilhoProdutoValor
BEFORE UPDATE OF VALOR ON PRODUTO
FOR EACH ROW
DECLARE
   ERRO EXCEPTION;
BEGIN
   INSERT INTO produtolog VALUES (sysdate, :old.codigo, :old.descricao, :old.valor, :new.valor, user);
   IF :old.codigo = 5 THEN
       RAISE ERRO;
   END IF;
   EXCEPTION WHEN ERRO THEN
       RAISE_APPLICATION_ERROR(-20000, 'Nao é permitido alterar esse produto!');
END;
```

Detectando tipo de trigger

- Variáveis INSERTING, UPDATING e DELETING disponíveis para testar tipo de trigger:

```
CREATE OR REPLACE TRIGGER GatilhoProdutosValorInsUpd
AFTER INSERT OR UPDATE ON PRODUTOS
FOR EACH ROW
BEGIN
    IF INSERTING THEN
        DBMS_OUTPUT.PUT_LINE('Trigger INSERT executado codigo ' || :new.codigo);
    ELSIF UPDATING THEN
        DBMS_OUTPUT.PUT_LINE('Trigger UPDATE executado codigo ' || :new.codigo);
    END IF;
END;
```

Leitura Adicional para Casa

- ELMASRI-NAVATHE – **Sistemas de Banco de Dados**
  - **Capítulo 9**

- Manual de PL/SQL do Oracle