

– SQL –

Gatilhos (*Triggers*)

Laboratório de Bases de Dados
Profa. Dra. Cristina Dutra de Aguiar Ciferri

Gatilho (*trigger*)

- Bloco PL/SQL que é disparado de forma automática e implícita sempre que ocorrer um evento associado a uma tabela
 - INSERT
 - UPDATE
 - DELETE
- Não pode ser chamado explicitamente

Utilidades

- Manutenção de tabelas de auditoria
- Manutenção de tabelas duplicatas
- Implementação de níveis de segurança mais complexos
- Geração de valores de colunas referentes a atributos derivados
- Validação de restrições de integridade mais complexas que as suportadas diretamente pelo SGBD

Estrutura

```
CREATE OR REPLACE TRIGGER nome_gatilho  
  BEFORE | AFTER  
  DELETE OR INSERT OR UPDATE OF coluna1, coluna2, ...  
  ON nome_da_tabela  
  REFERENCING OLD AS nome NEW AS nome  
  FOR EACH ROW  
  WHEN condição  
DECLARE  
  área de declaração  
BEGIN  
  área de comandos  
END ;
```

Cláusulas

- Tempo
 - BEFORE: antes do evento
 - AFTER: depois do evento
- Eventos de disparo
 - INSERT
 - UPDATE →
 - DELETE
- WHEN
 - restringe as tuplas que disparam o gatilho

o gatilho somente será disparado se alguma coluna especificada após a cláusula UPDATE OF for alterada

Cláusulas

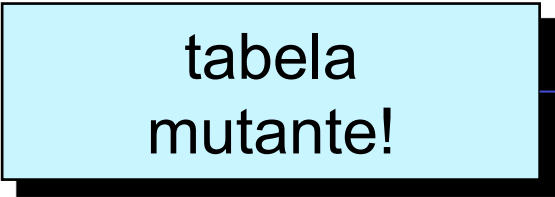
- **COMANDO**
 - aciona o gatilho antes ou depois de um comando, independentemente do número de tuplas afetadas
 - não requer a cláusula FOR EACH ROW
 - não permite o uso dos prefixos :OLD e :NEW
- **LINHA**
 - aciona o gatilho uma vez para cada linha afetada pelo comando ao qual o gatilho está associado
 - requer a cláusula FOR EACH ROW
 - permite o uso dos prefixos :OLD e :NEW
 - permite o uso das cláusulas REFERENCING e WHEN
 - não pode ler ou modificar a tabela à qual o gatilho está associado

Cláusulas

- COMANDO

- aciona o gatilho antes ou depois de um comando, independentemente do número de tuplas afetadas
- não requer a cláusula FOR EACH ROW
- não permite o uso dos prefixos :OLD e :NEW

tabela
mutante!



- LINHA

- aciona o gatilho uma vez para cada linha afetada pelo comando ao qual o gatilho está associado
- requer a cláusula FOR EACH ROW
- permite o uso dos prefixos :OLD e :NEW
- permite o uso das cláusulas REFERENCING e WHEN
- não pode ler ou modificar a tabela à qual o gatilho está associado

Cláusulas

- Referências aos valores dos atributos
 - :NEW.nome_atributo
 - indica um novo valor para um campo que está sendo alterado por um comando INSERT ou UPDATE
 - :OLD.nome_atributo
 - indica o valor anterior de um campo que está sendo alterado por um comando DELETE ou UPDATE
- REFERENCING OLD AS antigo NEW AS novo
substitui OLD por antigo e NEW POR novo

Uso de Gatilhos

- Restrições
 - o número máximo de gatilhos que podem ser especificados por tabela é 12
 - não é possível criar dois gatilhos diferentes com as mesmas características para uma mesma tabela
 - não é possível usar COMMIT ou ROLLBACK, inclusive em subprogramas chamados pelo gatilho
 - chaves primárias, únicas ou estrangeiras não podem ser alteradas por gatilhos

Predicados

- Retornam TRUE se o gatilho foi disparado por
 - INSERT: predicado `inserting`
 - UPDATE: predicado `updating`
 - DELETE: predicado `deleting`

- Exemplo

```
IF inserting THEN comandos_inserção;  
|   ELSIF deleting THEN comandos_remoção;  
|   ELSE comandos_atualização;  
END IF;
```

Exemplo (1/3)

```
CREATE OR REPLACE TRIGGER conta_aluno
AFTER INSERT OR DELETE OR UPDATE OF sexo_aluno
ON aluno
REFERENCING OLD AS antigo NEW AS novo
FOR EACH ROW
BEGIN
  IF inserting THEN
    IF :novo.sexo_aluno = 'f'
      THEN UPDATE totaliza SET total = total + 1
        WHERE sexo = 'feminino';
    ELSE UPDATE totaliza SET total = total + 1
        WHERE sexo = 'masculino';
  END IF;
```

Exemplo (2/3)

ELSIF deleting THEN

```
IF :antigo.sexo_aluno = 'f'
```

```
    THEN UPDATE totaliza SET total = total - 1
```

```
        WHERE sexo = 'feminino';
```

```
    ELSE UPDATE totaliza SET total = total - 1
```

```
        WHERE sexo = 'masculino';
```

```
END IF;
```



Exemplo (3/3)

```
ELSE IF :antigo.sexo_aluno = 'f'  
    THEN UPDATE totaliza SET total = total - 1  
        WHERE sexo = 'feminino';  
        UPDATE totaliza SET total = total + 1  
        WHERE sexo = 'masculino';  
    ELSE UPDATE totaliza SET total = total - 1  
        WHERE sexo = 'masculino';  
        UPDATE totaliza SET total = total + 1  
        WHERE sexo = 'feminino';  
    END IF;  
END IF;  
END;
```

Habilitar/Desabilitar

- ALTER TRIGGER
 - habilita/desabilita um ou mais gatilhos

```
ALTER TRIGGER nome_gatilho  
ENABLE | DISABLE;
```

- Exemplo
 - ALTER TRIGGER  conta_aluno ENABLE;
 - ALTER TRIGGER  aluno ENABLE ALL TRIGGERS;

Compilar

- ALTER TRIGGER
 - compila novamente um gatilho

```
ALTER TRIGGER nome_gatilho  
COMPILE;
```

- Exemplo
 - ALTER TRIGGER conta_aluno COMPILE;

Remover

- DROP TRIGGER
 - remove um gatilho do banco de dados

```
DROP TRIGGER nome_gatilho ;
```

- Exemplo
 - DROP TRIGGER conta_aluno;

Tabela USER_TRIGGERS

- Estrutura
 - TRIGGER_NAME : nome
 - TRIGGER_TYPE : tempo e tipo
 - TRIGGERING_EVENT : evento que dispara
 - TABLE_OWNER : proprietário da tabela associada
 - TABLE_NAME : nome da tabela associada
 - WHEN_CLAUSE : condição da restrição
 - STATUS : ativo ou inativo
 - DESCRIPTION : declaração
 - TRIGGER_BODY : bloco PL/SQL

Problema de Tabela Mutante

- Ocorre quando um gatilho refere-se à própria tabela que está sendo alterada
- Exemplo

```
CREATE OR REPLACE TRIGGER testa_empresa  
AFTER INSERT  
ON empresa  
FOR EACH ROW  
BEGIN  
UPDATE empresa SET descricao = 'passei por aqui'  
WHERE emp_codigo = :new.emp_codigo;  
END;
```

Problema de Tabela Mutante

```
INSERT INTO empresa  
VALUES (5,'TESTE', 'TESTE', '(44) 3788-3737', NULL);
```

Error starting at line 43 in command:

```
INSERT INTO empresa VALUES (5,'TESTE', 'TESTE', '(44) 3788-3737', NULL)
```

Error report:

SQL Error: ORA-04091: table A5357762.EMPRESA is mutating, trigger/function may not see it

ORA-06512: at "A5357762.TESTA_EMPRESA", line 2

ORA-04088: error during execution of trigger 'A5357762.TESTA_EMPRESA'

04091. 00000 - "table %s.%s is mutating, trigger/function may not see it"

*Cause: A trigger (or a user defined plsql function that is referenced in this statement) attempted to look at (or modify) a table that was in the middle of being modified by the statement which fired it.

*Action: Rewrite the trigger (or function) so it does not read that table.

Soluções

- Uso combinado de gatilhos e pacote
 - essa solução deve ser usada no trabalho prático 4
- Uso de transação autônoma
 - PRAGMA
AUTONOMOUS_TRANSACTION;
 - COMMIT;
- Uso combinado de gatilhos e visão