

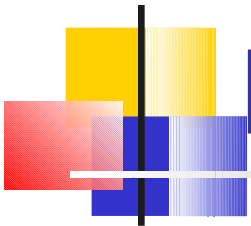
# Laboratório de Bases de Dados



Prof. José Fernando Rodrigues Júnior

## **Aula 8 – Triggers**

Material: Profa. Elaine Parros Machado de Sousa



# Procedures X Triggers

---

Procedure/Function	Trigger
bloco identificado PL/SQL	bloco identificado PL/SQL
pode ser usado em pacotes ou <u>triggers</u>	objeto independente
recebe parâmetros	não recebe parâmetros
executado explicitamente	executado (disparado) implicitamente – eventos desencadeadores



# Triggers

---

- Para que usar?
  - restrições de consistência e validade que não possam ser implementadas com *constraints*
  - mecanismos de validação que envolvam pesquisas em múltiplas tabelas
  - criar conteúdo de uma coluna derivado de outras
  - atualizar tabelas em função da atualização de uma determinada tabela
  - criar *logs* – segurança (auditoria)
  - .....



# Triggers em Oracle

---

- Tipos

- Para tabelas

- Triggers de DML (INSERT, UPDATE, DELETE)
    - Triggers de Sistema (DDL, e logs)

- Para visões

- Triggers de DML Instead-OF (INSERT, UPDATE, DELETE)



# Triggers de DML

---

- **Tabela desencadeadora**
- **Instrução de disparo**
  - INSERT
  - UPDATE
  - DELETE
- ***Timing***
  - BEFORE
  - AFTER
- **Nível**
  - linha
  - instrução

# Exemplo

Turma = {Sigla, Numero, NAlunos}

Matrícula = {Sigla, Numero, Aluno, Ano, Nota}

```
CREATE OR REPLACE TRIGGER NroDeAlunos
AFTER DELETE ON Matricula
FOR EACH ROW /* nível de linha */

BEGIN
    UPDATE Turma
    SET NAlunos = NAlunos - 1
    WHERE Sigla = :old.Sigla AND
           Numero = :old.Numero;

EXCEPTION
    ....
END NroDeAlunos;
```

**identificador :old**  
**refere-se à tabela**  
**Matricula**

```
/*... em um bloco anônimo, por exemplo... */
DELETE FROM matricula WHERE aluno = 222;
```

**O que acontece na base de dados????**



# Triggers de DML

---

- **Identificadores de correlação** – variáveis de vínculo PL/SQL (p/ triggers com nível de linha)
  - sempre vinculados à tabela desencadeadora do trigger
  - pseudoregistros do tipo *tabela\_desencadeadora*%ROWTYPE

<div>instrução</div> <div>identificador</div>	:old	:new
INSERT	NULL	valores que serão inseridos
UPDATE	valores antes da atualização	novos valores para a atualização
DELETE	valores antes da remoção	NULL

# O que um trigger executa?

```
CREATE OR REPLACE TRIGGER Teste
AFTER INSERT OR UPDATE OR DELETE ON LBD01_VINCULO_USP
FOR EACH ROW

BEGIN
    insert into output
    values (:old.nrousp || '-' || :old.tipovinc || '-' || :old.nome || '-' || :old.dataingressos || '-' || :old.datanascimento || '-' || :old.ativo);

    insert into output
    values (:new.nrousp || '-' || :new.tipovinc || '-' || :new.nome || '-' || :new.dataingressos || '-' || :new.datanascimento || '-' || :new.ativo);
END;

*Output
CREATE TABLE output(msg varchar2(200));
```





# O que um trigger executa?

---

- **Suponha os seguintes comandos SQL**
  - **INSERT**
    - `INSERT INTO LBD01_VINCULO_USP(NROUSP, TIPOVINC, NOME, DATAINGRESSO) VALUES(21, 2, 'João', '05/09/2002')`
    - `INSERT INTO LBD01_VINCULO_USP(NROUSP, TIPOVINC, NOME, DATAINGRESSO) VALUES(22, 1, 'Gilberto', '09/05/2000', '05/02/1980', 'y')`
    - `INSERT INTO LBD01_VINCULO_USP(NROUSP, TIPOVINC, NOME, DATAINGRESSO) VALUES(23, 3, 'Alfredo', '03/04/2002', '05/04/1982', 'y')`
  - **UPDATE**
    - `UPDATE LBD01_VINCULO_USP SET NOME = UPPER(NOME) WHERE NROUSP > 20;`
  - **DELETE**
    - `DELETE LBD01_VINCULO_USP SET NOME = UPPER(NOME) WHERE NROUSP > 20;`
- **Quais são os valores de :old e :new para cada uma destas as operações?**



# O que um trigger executa?

---

- Primeiramente, o corpo do trigger será executado 3 vezes para cada tipo de operação
- Os valores para cada execução considerando cada operação serão:

INSERT	:old						:new					
	NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo	NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo
	NULL	NULL	NULL	NULL	NULL	NULL	21	2	João	05/09/2002	NULL	y
	NULL	NULL	NULL	NULL	NULL	NULL	22	1	Gilberto	09/05/2000	NULL	y
	NULL	NULL	NULL	NULL	NULL	NULL	23	3	Alfredo	03/04/2002	NULL	y

- DataNascimento não foi inserido, e Ativo tem valor DEFAULT.



# O que um trigger executa?

- Primeiramente, o corpo do trigger será executado 3 vezes para cada tipo de operação
- Os valores para cada execução considerando cada operação serão:

:old

:new

INSERT

NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo
NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL

NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo
21	2	João	05/09/2002	NULL	y
22	1	Gilberto	09/05/2000	NULL	y
23	3	Alfredo	03/04/2002	NULL	y

UPDATE

NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo
21	2	Joao	05/09/2002	NULL	y
22	1	Gilberto	09/05/2000	NULL	y
23	3	Alfredo	03/04/2002	NULL	y

NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo
21	2	JOAO	05/09/2002	NULL	y
22	1	GILBERTO	09/05/2000	NULL	y
23	3	ALFREDO	03/04/2002	NULL	y

# O que um trigger executa?

- Primeiramente, o corpo do trigger será executado 3 vezes para cada tipo de operação
- Os valores para cada execução considerando cada operação serão:

:old

:new

INSERT

NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo
NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL

NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo
21	2	João	05/09/2002	NULL	y
22	1	Gilberto	09/05/2000	NULL	y
23	3	Alfredo	03/04/2002	NULL	y

UPDATE

NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo
21	2	João	05/09/2002	NULL	y
22	1	Gilberto	09/05/2000	NULL	y
23	3	Alfredo	03/04/2002	NULL	y

NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo
21	2	JOAO	05/09/2002	NULL	y
22	1	GILBERTO	09/05/2000	NULL	y
23	3	ALFREDO	03/04/2002	NULL	y

DELETE

NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo
21	2	João	05/09/2002	NULL	y
22	1	Gilberto	09/05/2000	NULL	y
23	3	Alfredo	03/04/2002	NULL	y

NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo
NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL

# O que um trigger executa?

- Primeiramente, o corpo do trigger será executado 3 vezes para cada tipo de operação
- Os valores para cada execução considerando cada operação serão:

:old

:new

INSERT

NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo	NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo
NULL	NULL	NU								NULL	y
NULL	NULL	NU								NULL	y
NULL	NULL	NU								NULL	y

Esta análise é válida tanto para BEFORE quanto para AFTER.

UPDATE

NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo	NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo
21	2	Joao	05/09/2002	NULL	y	21	2	JOAO	05/09/2002	NULL	y
22	1	Gilberto	09/05/2000	NULL	y	22	1	GILBERTO	09/05/2000	NULL	y
23	3	Alfredo	03/04/2002	NULL	y	23	3	ALFREDO	03/04/2002	NULL	y

DELETE

NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo	NroUSP	TipoVinc	Nome	DataIngresso	DataNascimento	Ativo
21	2	João	05/09/2002	NULL	y	NULL	NULL	NULL	NULL	NULL	NULL
22	1	Gilberto	09/05/2000	NULL	y	NULL	NULL	NULL	NULL	NULL	NULL
23	3	Alfredo	03/04/2002	NULL	y	NULL	NULL	NULL	NULL	NULL	NULL



# Triggers de DML

---

- Ordem de disparo e execução

1ª. *triggers* **BEFORE** – nível de instrução

2ª. para cada linha (da tabela) afetada pela instrução

*a.* *triggers* **BEFORE** – nível de linha

*b.* instrução

*c.* *triggers* **AFTER** – nível de linha

3ª. *triggers* **AFTER** – nível de instrução



# Triggers de DML

---

- Mesmo *trigger* pode ser disparado para mais de um tipo de instrução na mesma tabela
- ex:  
`CREATE TRIGGER nome BEFORE INSERT OR UPDATE ON tabela ...`
- *Trigger* é executado com a instrução na mesma transação
- Mesma tabela pode ter mais de um *trigger* associado
  - inclusive mais de um *trigger* do mesmo tipo
    - ex: 2 *triggers* `BEFORE INSERT FOR EACH ROW`
    - disparados em sequência (ordem não definida)

# Exemplo

---

```
CREATE OR REPLACE TRIGGER AcertaNota
BEFORE INSERT OR UPDATE ON Matricula
FOR EACH ROW
/* condição WHEN - somente para trigger em nível de linha */
WHEN (new.nota < 0) /*condição booleana
                    avaliada para cada linha */
BEGIN
    :new.nota := 0;

    EXCEPTION
        . . . . .
END AcertaNota;
```



# Exemplo

---

```
CREATE OR REPLACE TRIGGER AcertaNota
BEFORE INSERT OR UPDATE ON Matricula
/*especificando nomes para NEW e OLD ...*/
REFERENCING new AS nova_matricula
FOR EACH ROW
WHEN (nova_matricula.nota < 0)

BEGIN
    :nova_matricula.nota := 0;
END AcertaNota;
```

# Exemplo – criando log

```
CREATE OR REPLACE TRIGGER LogDisciplina
AFTER INSERT OR UPDATE OR DELETE ON Disciplina
FOR EACH ROW

DECLARE
    v_operacao CHAR;

BEGIN
    /*usando predicados booleanos...*/
    IF INSERTING THEN v_operacao := 'I';
    ELSIF UPDATING THEN v_operacao := 'U';
    ELSIF DELETING THEN v_operacao := 'D';
    END IF;

    INSERT INTO logTabelaDisciplina
        VALUES (USER, SYSDATE, v_operacao);

END LogDisciplina;
```



# Triggers de Sistema

---

- *Triggers* disparados por:
  - instruções DDL - *timing*
    - CREATE – before/after
    - ALTER - before/after
    - DROP - before/after
    - DDL - before/after
  - eventos do banco de dados - *timing*
    - STARTUP - after
    - SHUTDOWN - before
    - LOGON - after
    - LOGOFF - before
    - SERVERERROR – after
    - ...



# Triggers de Sistema

---

- Níveis
  - DATABASE
  - SCHEMA
    - do usuário que criou o *trigger* ou de outro usuário
- Atributos de Eventos
  - ex:
    - ora\_server\_error
      - OU sys.server\_error
    - ora\_dict\_obj\_owner
      - OU sys.dictionary\_obj\_owner
    - ora\_dict\_obj\_type
      - OU sys.dictionary\_obj\_type
    - ...

# Exemplo

---

```
-- conectado como labbd
```

```
CREATE OR REPLACE TRIGGER TodosUsuarios  
AFTER LOGON ON DATABASE
```

```
BEGIN
```

```
    INSERT INTO logUser VALUES (USER, 'Trigger TodosUsuarios');  
END;
```

```
-- conectado como labbd
```

```
CREATE OR REPLACE TRIGGER UsuarioLogado  
AFTER LOGON ON SCHEMA
```

```
BEGIN
```

```
    INSERT INTO logUser VALUES (USER, 'Trigger UsuarioLogado');  
END;
```

---



# Triggers Instead-of

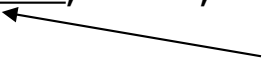
---

- Usados para alterar **visões não atualizáveis** e **visões de junção atualizáveis**
  - permitem fazer as atualizações de maneira adequada para a **semântica da aplicação**
- Executa o corpo do *trigger* **AO INVÉS** da instrução que o acionou

# Exemplo

Professor = {Nome, NFunc, Idade, Titulação}

Disciplina = {Sigla, Nome, NCred, Professor, Livro}



```
CREATE VIEW prof_disciplina AS
  SELECT d.sigla, d.nome, p.nfunc, p.nome as
                                     professor
  FROM disciplina d, professor p
 WHERE d.professor = p.nfunc;
```

-- qual o efeito do comando abaixo na base de dados???

```
DELETE FROM prof_disciplina WHERE nfunc = 111;
```

# Exemplo

Professor = {Nome, NFunc, Idade, Titulação}

Disciplina = {Sigla, Nome, NCred, Professor, Livro}

Nesta view, apenas a tabela disciplina terá preservação de chave, portanto a operação de delete sobre um atributo da tabela Professor causará um erro: tentativa de atualização de tabela sem preservação de chave.

-- qual o efeito do comando abaixo na base de dados???

```
DELETE FROM prof_disciplina WHERE nfunc = 111;
```



# Exemplo

---

Professor = {Nome, NFunc, Idade, Titulação}

Disciplina = {Sigla, Nome, NCred, Professor, Livro}



```
CREATE OR REPLACE TRIGGER RemoveProfDisciplina  
INSTEAD OF DELETE ON prof_disciplina  
FOR EACH ROW /* opcional - sempre é nível de linha */
```

```
BEGIN
```

```
UPDATE disciplina SET professor = null  
WHERE professor = :old.nfunc;
```

```
DELETE FROM professor WHERE nfunc = :old.nfunc;
```

```
END RemoveProfDisciplina;
```



# Outros comandos

---

- `alter/drop trigger`
- Consulta
  - *SQL Reference*
  - *Database Concepts*
  - *Application Developer's Guide – Fundamentals*
    - usando triggers: informações, exemplos, eventos, atributos,...

# PRÁTICA 8

