Universidade Federal de Mato Grosso-UFMT
Sistemas de Informação
Laboratório de Banco de Dados
Prof. Clóvis

Triggers

(Gatilhos)

TRIGGERS – Definição

- Programa PL/SQL associado a tabelas do banco de dados;
- São executadas automaticamente em operações de INSERT, UPDATE ou DELETE
- Procedimentos e funções podem ser chamadas explicitamente;

TRIGGERS – Finalidades

- Auditoria de dados;
- Criação de Log de forma transparente;
- Impor regras de negócios complexas;
- Gerar valores para colunas automaticamente;
- Mater replicação de tabelas;

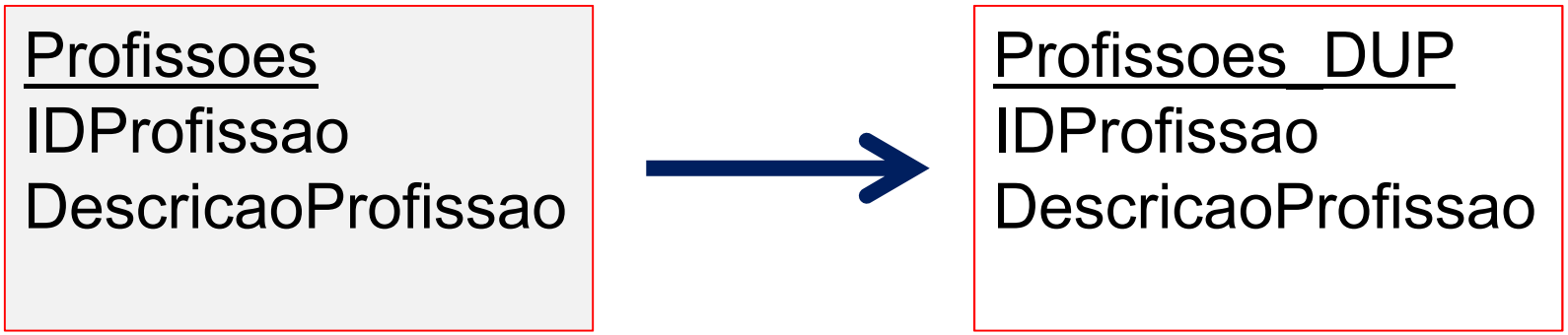
TRIGGERS – Sintaxe

```
CREATE [OR REPLACE] TRIGGER nome_trigger
{BEFORE|AFTER} evento ON nome_tabela
[FOR EACH ROW]
[WHEN condição]
DECLARE
    Declarações de variáveis
BEGIN
    Comandos
EXCEPTION
    excessões
END;
```

TRIGGERS – :OLD e :NEW

- Colunas OLD e NEW determinam valores das colunas antes e após atualizações;
- **:new** usado em operações de INSERT e UPDATE
- **:old** usado em operações de DELETE e UPDATE
- Triggers podem disparar outras triggers em cascata quando há interdependência entre tabelas (CASCADING). Não é recomendável criar numero elevado de interdependências.

TRIGGERS – Duplicação

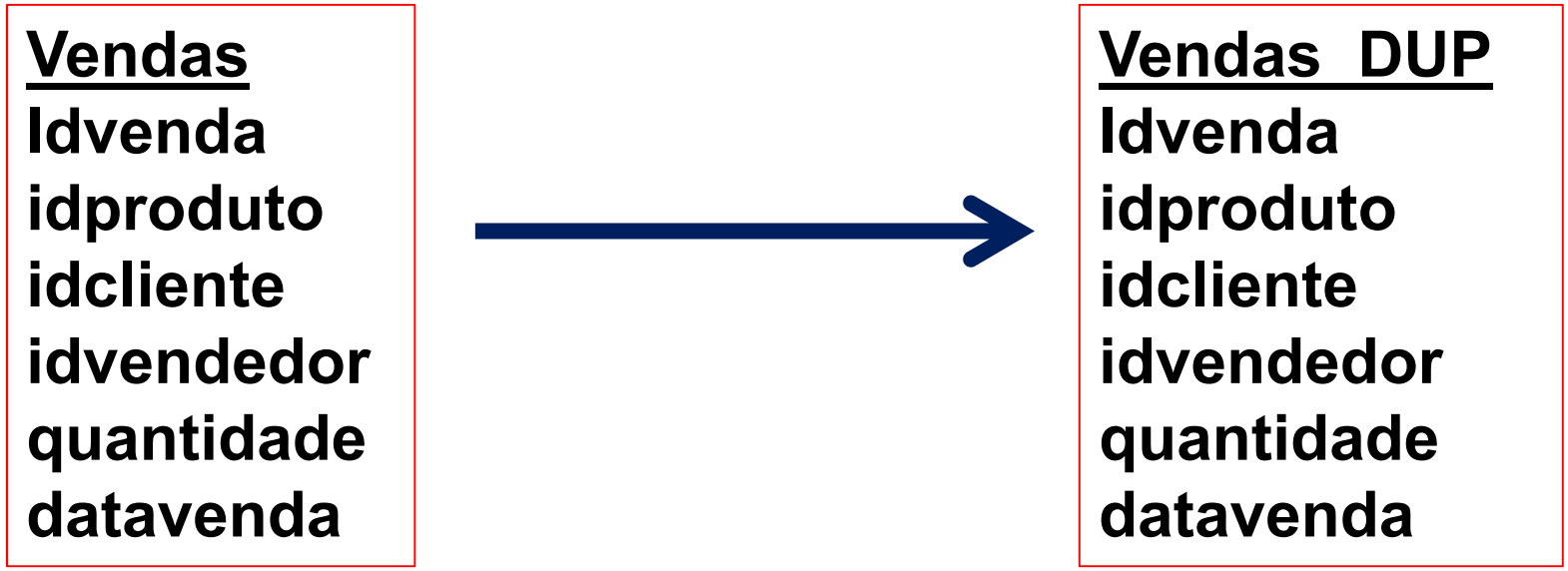


TRIGGERS – Duplicação

CREATE TABLE profissoes_dup(idprofissao NUMBER(4,0) NOT NULL, descricaoprofissao VARCHAR2(40) NULL)

```
CREATE OR REPLACE TRIGGER TI_Profissoes
AFTER INSERT
ON Profissoes REFERENCING NEW AS N
FOR EACH ROW
BEGIN
INSERT INTO
   Profissoes_dup(idprofissao,descricaoprofissao)
  VALUES(:N.idprofissao,:N.descricaoprofissao);
END;
```

TRIGGERS – Duplicação



TRIGGERS – Duplicação

```
CREATE TABLE vendas_dup (     idvenda    NUMBER(12,0) NOT NULL,
                              idproduto  NUMBER(8,0)  NOT NULL,
                              idcliente  NUMBER(8,0)  NOT NULL,
                              idvendedor NUMBER(3,0)  NOT NULL,
                              quantidade NUMBER(8,2)  NULL,
                              datavenda  DATE         NULL
                            )
```

```
CREATE OR REPLACE TRIGGER TI_Vendas
AFTER INSERT
ON Vendas REFERENCING NEW AS N
FOR EACH ROW
BEGIN
INSERT INTO
   vendas_dup(idvenda,idproduto,idcliente,idvendedor,
   quantidade,datavenda)
  VALUES(:N.idvenda,:N.idproduto,:N.idcliente,
   :N.idvendedor,:N.quantidade,:N.datavenda);
END;
```

TRIGGERS – Atualização

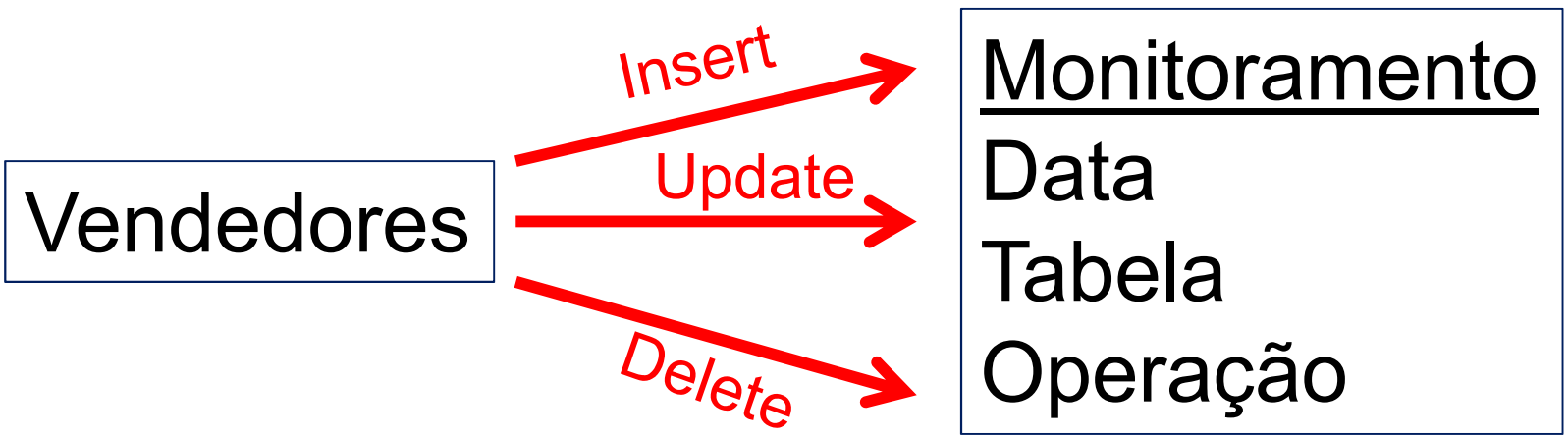
```
CREATE OR REPLACE TRIGGER TU_vendas
AFTER update ON Vendas REFERENCING NEW AS N
FOR EACH ROW
DECLARE total NUMBER(3);
BEGIN
Select count(*) INTO total from Vendas_dup where
 Vendas_dup.idvenda=:n.idvenda;
IF total>0 then
UPDATE vendas_dup SET
   vendas_dup.idvenda=:N.idvenda,vendas_dup.idproduto=:N.idproduto,
   vendas_dup.idcliente=:N.idcliente,
   vendas_dup.idvendedor=:N.idvendedor,
   vendas_dup.quantidade=:N.quantidade,
   vendas_dup.datavenda=:N.datavenda
  WHERE vendas_dup.idvenda=:N.idvenda;
ELSE INSERT INTO vendas_dup(idvenda,idproduto,idcliente,idvendedor,
   quantidade,datavenda) VALUES(:N.idvenda,:N.idproduto,:N.idcliente,
   :N.idvendedor,:N.quantidade,:N.datavenda);
END IF;
END;
```

TRIGGERS – Exclusão

```
CREATE OR REPLACE TRIGGER TD_Vendas
AFTER delete ON Vendas REFERENCING OLD
  AS N
FOR EACH ROW
BEGIN
Delete from Vendas_dup where
 Vendas_dup.idvenda=:n.idvenda;
END;
```

TRIGGERS – LOG

INSERT OR DELETE OR UPDATE



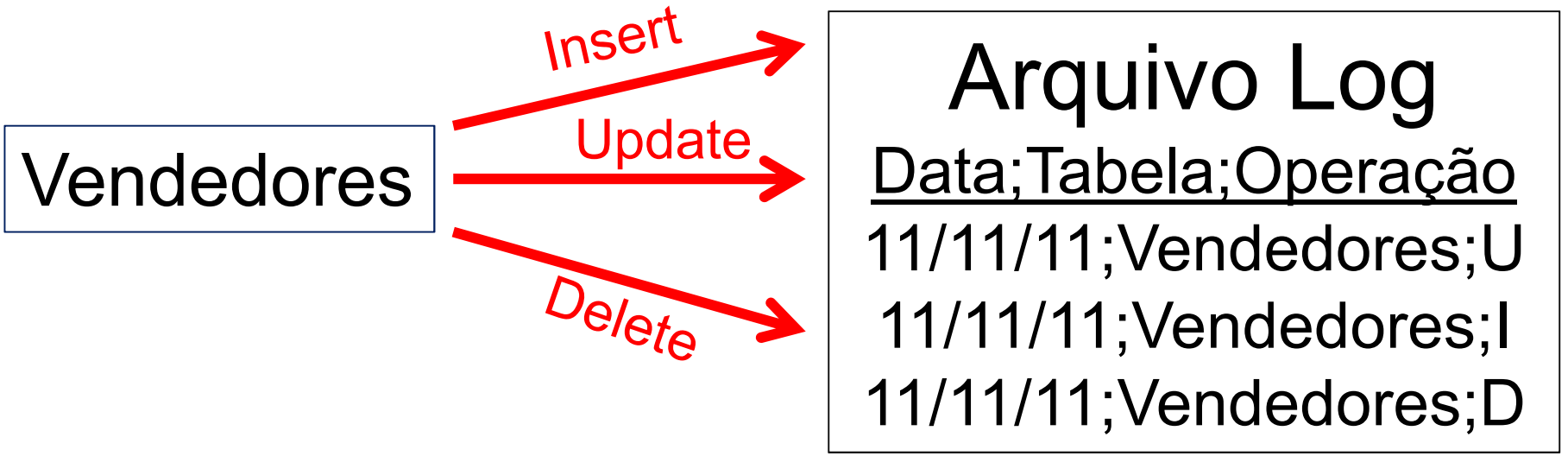
TRIGGERS – LOG

```
CREATE TABLE monitoramento
        (data DATE, tabela VARCHAR(30),
        operacao VARCHAR2(1))
```

TRIGGERS – LOG

```
CREATE OR REPLACE TRIGGER tr_Monitor
BEFORE INSERT OR DELETE OR UPDATE
ON Vendedores FOR EACH ROW
BEGIN
IF INSERTING THEN
  INSERT INTO monitoramento(data,tabela,operacao)
                              VALUES (SYSDATE,'VENDEDORES','I');
  ELSE IF UPDATING THEN
    INSERT INTO monitoramento(data,tabela,operacao)
                              VALUES (SYSDATE,'VENDEDORES','U');
    ELSE IF DELETING THEN
      INSERT INTO monitoramento(data,tabela,operacao)
                              VALUES (SYSDATE,'VENDEDORES','D');
    END IF;
  END IF;
END IF;
END;
```

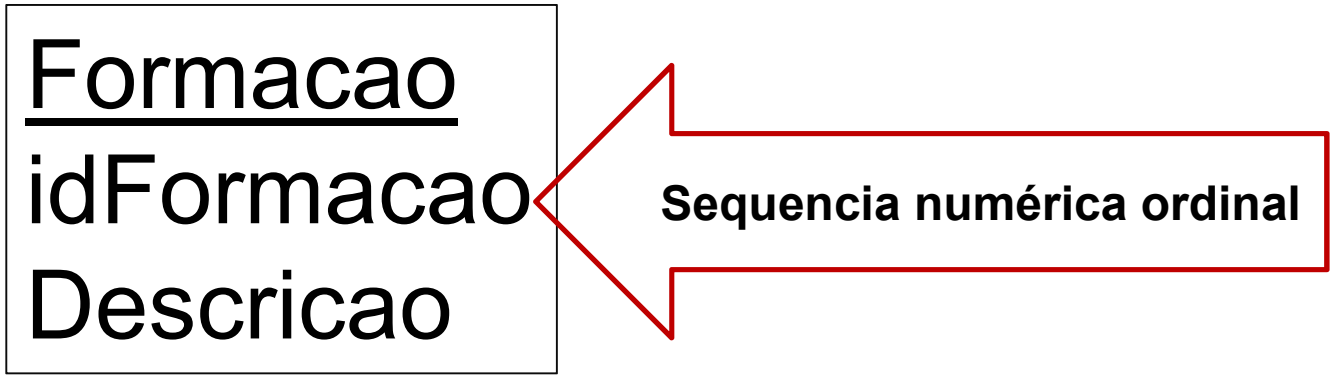
TRIGGERS – LOG - Arquivo



TRIGGERS – LOG - Arquivo

```
CREATE OR REPLACE TRIGGER TG_Monitor_Arquivo
BEFORE INSERT OR DELETE OR UPDATE
ON Vendedores FOR EACH ROW
DECLARE arq utl_file.file_type;
BEGIN
IF verificar_arquivo('DESTINO','Vendedores.csv')=0 THEN
  arq:=utl_file.fopen('DESTINO','Vendedores.csv','W');
ELSE arq:=utl_file.fopen('DESTINO','Vendedores.csv','A');
END IF;
IF INSERTING THEN
  utl_file.PUT_LINE(arq,SYSDATE||';'||
                    To_Char(SYSDATE,'HH:MI')||';VENDEDORES;INSERT;'||:NEW.nomevendedor);
  ELSE IF UPDATING THEN
    utl_file.PUT_LINE(arq,SYSDATE||';'||
                      To_Char(SYSDATE,'HH:MI')||';VENDEDORES;UPDATE;'||:NEW.nomevendedor);
    ELSE IF DELETING THEN
      utl_file.PUT_LINE(arq,SYSDATE||';'||
                        To_Char(SYSDATE,'HH:MI')||';VENDEDORES;DELETE;'||:OLD.nomevendedor);
    END IF;
  END IF;
END IF;
utl_file.fclose(arq);
END;
```

Autoincremento com Trigger - Detalhamento



Autoincremento com Trigger - Detalhamento

Estrutura da tabela para o teste

```
CREATE TABLE Formacao(idFormacao NUMBER(3) PRIMARY KEY,
Descricao VARCHAR2(30))
```

Código Fonte (Trigger)

```
CREATE OR REPLACE TRIGGER TI_Inserir_Formacao
BEFORE INSERT ON Formacao
REFERENCING NEW AS NEW FOR EACH ROW
DECLARE
ultimo NUMBER(5);
BEGIN
    SELECT Nvl(Max(F.idFormacao),0) INTO Ultimo FROM Formacao F;
    :New.idFormacao:=Ultimo+1;
END;
```

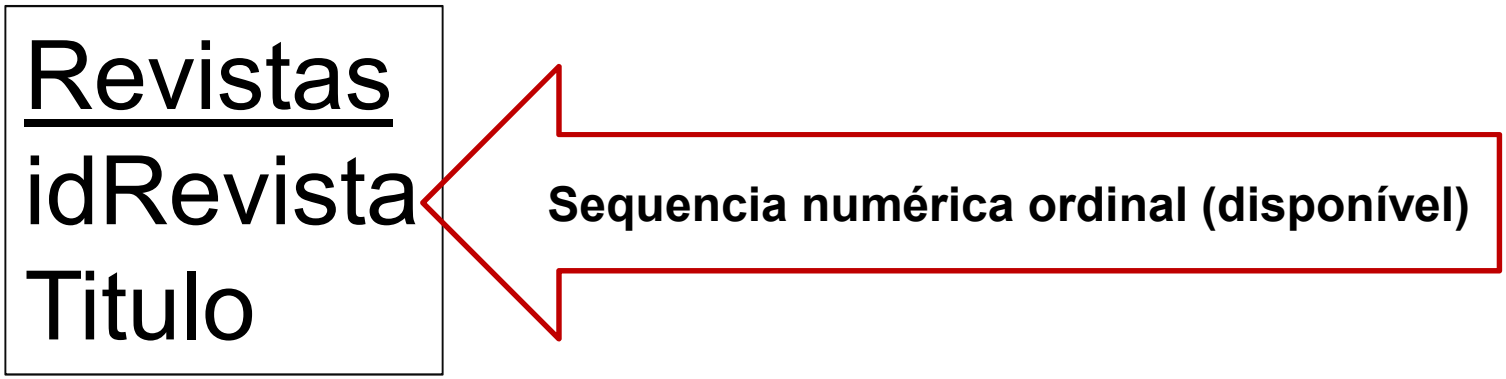
Autoincremento com Trigger - Detalhamento

Estrutura da tabela para o teste

```
CREATE TABLE revistas(idrevista NUMBER(5),
        titulo VARCHAR2(80), PRIMARY KEY(idrevista))
```

Inserções de dados com códigos intercalados

```
INSERT INTO revistas(idrevista,titulo) VALUES (1,'PC Magazine');
INSERT INTO revistas(idrevista,titulo) VALUES (4,'Wired');
INSERT INTO revistas(idrevista,titulo) VALUES (7,'Veja');
```



Autoincremento com Trigger - Detalhamento

```
CREATE OR REPLACE TRIGGER TI_Inserir_revista BEFORE INSERT ON Revistas
REFERENCING NEW AS NEW FOR EACH ROW
DECLARE
primeiro NUMBER(5); ultimo NUMBER(5); aux NUMBER(5); L NUMBER(5); disponivel number(5);
BEGIN
Disponivel:=0;
SELECT Min(r1.idrevista) INTO Primeiro FROM revistas r1;
IF primeiro IS NULL THEN
   Disponivel:=1;
 ELSE
 BEGIN
    SELECT Max(r2.idrevista) INTO Ultimo FROM revistas r2;
    FOR L IN Primeiro..Ultimo
    LOOP
      SELECT Count(*) INTO aux FROM revistas r3 WHERE r3.idrevista=L;
       IF (aux=0) THEN
        Disponivel:=L;
        EXIT;
       END IF;
    END LOOP;
    IF Disponivel=0 THEN Disponivel:=Ultimo+1; END IF;
 END;
END IF;
:New.idrevista:=disponivel;
END
```

Tópicos para Avaliação

Trigger – Contextos do Sistema

1. Procedimentos
2. Funções
3. Cursores
4. Pacotes
5. Triggers

Trigger – Contextos do Sistema

sys_context (NAMESPACE, PARÂMETRO)

sys_context ('USERENV',Parâmetro')

Trigger – Contextos do Sistema

ACTION
AUDITED_CURSORID
AUTHENTICATED_IDENTITY
AUTHENTICATION_DATA
AUTHENTICATION_METHOD
BG_JOB_ID
CLIENT_IDENTIFIER
CLIENT_INFO
CURRENT_BIND
CURRENT_SCHEMA
CURRENT_SCHEMAID
DB_DOMAIN
DB_NAME
DB_UNIQUE_NAME
ENTRYID
ENTERPRISE_IDENTITY
HOST
IDENTIFICATION_TYPE
INSTANCE
IP_ADDRESS

LANGUAGE
NETWORK_PROTOCOL
NLS_CURRENCY
NLS_DATE_FORMAT
NLS_TERRITORY
OS_USER
SERVER_HOST
SERVICE_NAME
SESSION_USER
SESSION_USERID
SID
STATEMENTID
TERMINAL

Trigger – Contextos do Sistema

```
SELECT sys_context('USERENV', 'TERMINAL')
COMPUTADOR, sys_context('USERENV', 'IP_ADDRESS')
SERVIDOR FROM DUAL;
```

```
SELECT sys_context('USERENV', 'SESSION_USER')
USUARIO,sys_context('USERENV', 'IP_ADDRESS') IP,
sys_context('USERENV', 'TERMINAL') COMPUTADOR,
sys_context('USERENV', 'HOST') SERVIDOR, SYSDATE
DATA
FROM DUAL;
```

Trigger - Auditoria

```
CREATE TABLE TABELA_LOG(usuario VARCHAR2(20),ip
VARCHAR2(15),  computador VARCHAR2(30), servidor VARCHAR2(60),
data DATE);
```

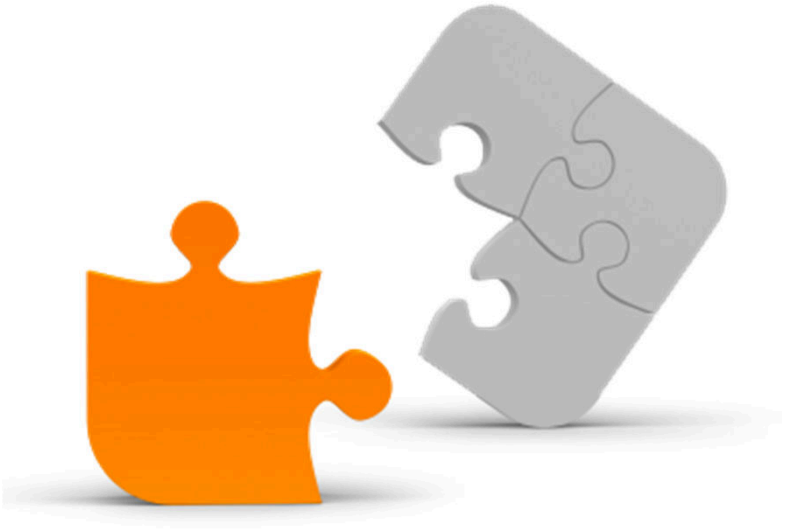
```
create or replace trigger trigger_auditoria
AFTER LOGON ON DATABASE
BEGIN
 insert into tabela_log (usuario,ip,computador,servidor,data) VALUES
 (sys_context('USERENV', 'SESSION_USER') ,sys_context('USERENV',
'IP_ADDRESS') ,
    sys_context('USERENV', 'TERMINAL') , sys_context('USERENV',
'HOST'), SYSDATE);
END;
```

Trigger – Controle de Excessões

Exercicios3

ALTER table revistas ADD(anopublicacao VARCHAR2(4))

anopublicacao<4

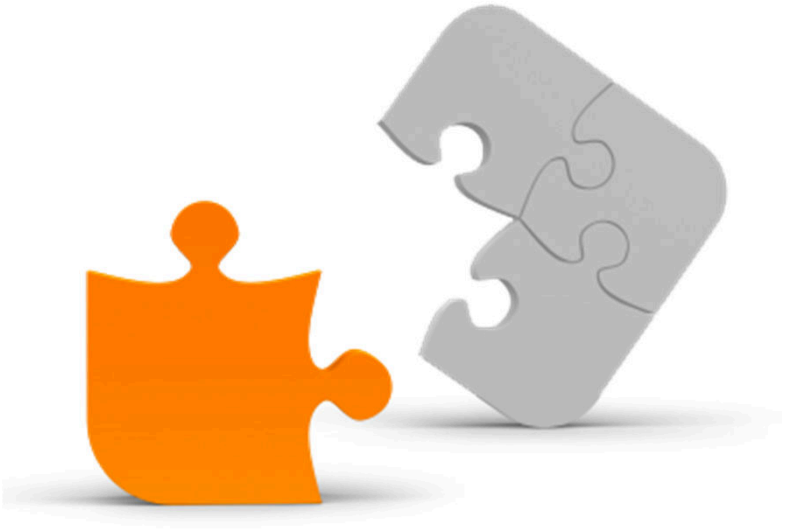


Trigger – Controle de Excessões

Exercicios3

ALTER TABLE clientes ADD(cpf VARCHAR2(11))

Valores devem ser numéricos



Trigger – Controle de Excessões

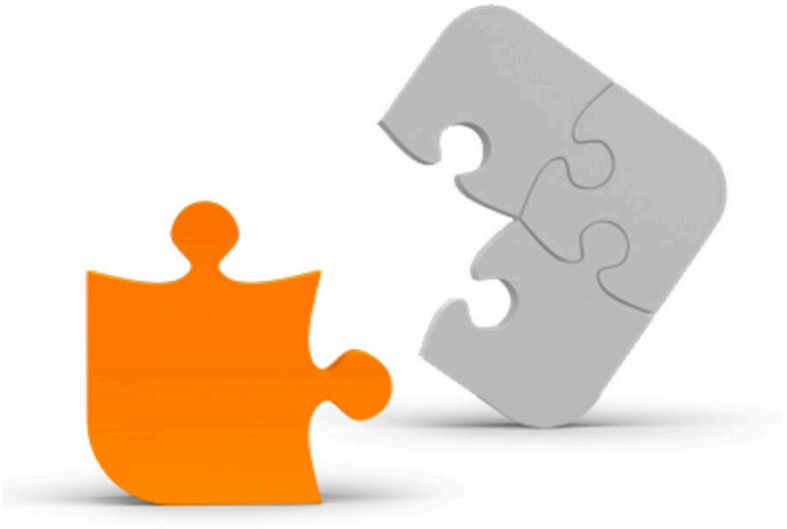
```
CREATE OR REPLACE TRIGGER TI_ANO_Revista
BEFORE INSERT OR UPDATE ON revistas REFERENCING
NEW AS NEW FOR EACH ROW
BEGIN
IF Length(:NEW.anopublicacao)<4 THEN
  RAISE_APPLICATION_ERROR(-20000,'A coluna ano deve
ter 4 digitos');
END IF;
END;
```

Trigger – Controle de Excessões

Exercicios3

ALTER TABLE clientes ADD(cpf VARCHAR2(11))

Valores devem ser numéricos

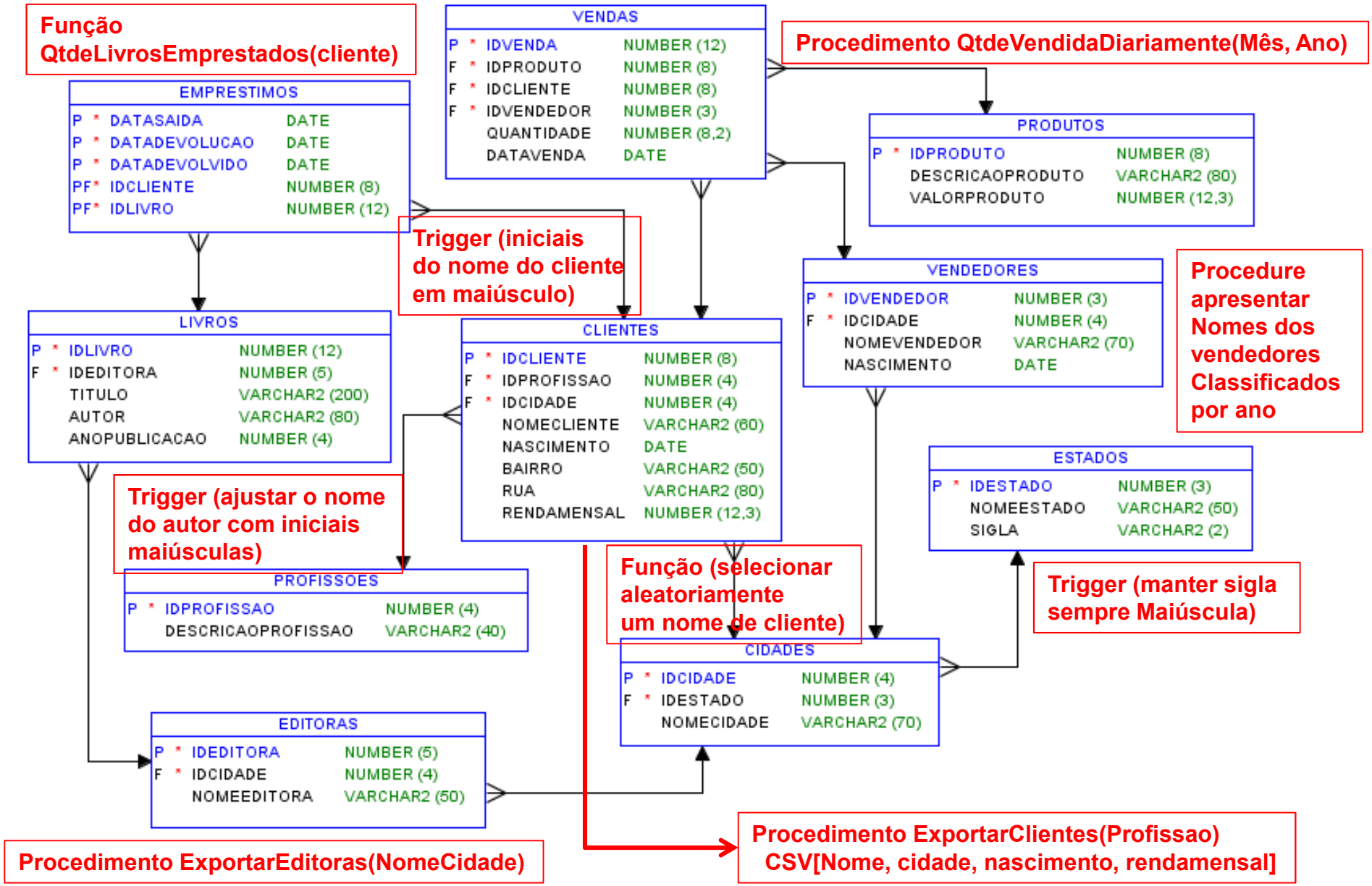


Trigger – Controle de Excessões

```
CREATE OR REPLACE FUNCTION verificanumero (p_string IN VARCHAR2)
   RETURN integer IS
   valor NUMBER;
BEGIN
   valor := TO_NUMBER(p_string);
   RETURN 1;
EXCEPTION
WHEN VALUE_ERROR THEN
   RETURN 0;
END;
```

```
CREATE OR REPLACE TRIGGER TI_CPF_CLIENTES
BEFORE INSERT OR UPDATE ON clientes REFERENCING NEW AS NEW FOR
EACH ROW
BEGIN
IF verificanumero(:NEW.cpf)=0 THEN
  RAISE_APPLICATION_ERROR(-20000,'O CPF deveser numérico');
END IF;
END;
```

Exercícios


Função QtdeLivrosEmprestados(cliente)
VENDAS
P * IDVENDA NUMBER (12)
F * IDPRODUTO NUMBER (8)
F * IDCLIENTE NUMBER (8)
F * IDVENDEDOR NUMBER (3)
QUANTIDADE NUMBER (8,2)
DATAVENDA DATE
Procedimento QtdeVendidaDiariamente(Mês, Ano)
EMPRESTIMOS
P * DATASAIDA DATE
P * DATADEVOLUCAO DATE
P * DATADEVOLVIDO DATE
PF* IDCLIENTE NUMBER (8)
PF* IDLIVRO NUMBER (12)
PRODUTOS
P * IDPRODUTO NUMBER (8)
DESCRICAOPRODUTO VARCHAR2 (80)
VALORPRODUTO NUMBER (12,3)
Trigger (iniciais do nome do cliente em maiúsculo)
VENDEDORES
P * IDVENDEDOR NUMBER (3)
F * IDCIDADE NUMBER (4)
NOMEVENDEDOR VARCHAR2 (70)
NASCIMENTO DATE
Procedure apresentar Nomes dos vendedores Classificados por ano
LIVROS
P * IDLIVRO NUMBER (12)
F * IDEDITORA NUMBER (5)
TITULO VARCHAR2 (200)
AUTOR VARCHAR2 (80)
ANOPUBLICACAO NUMBER (4)
CLIENTES
P * IDCLIENTE NUMBER (8)
F * IDPROFISSAO NUMBER (4)
F * IDCIDADE NUMBER (4)
NOMECLIENTE VARCHAR2 (60)
NASCIMENTO DATE
BAIRRO VARCHAR2 (50)
RUA VARCHAR2 (80)
RENDAMENSAL NUMBER (12,3)
ESTADOS
P * IDESTADO NUMBER (3)
NOMEESTADO VARCHAR2 (50)
SIGLA VARCHAR2 (2)
Trigger (ajustar o nome do autor com iniciais maiúsculas)
Função (selecionar aleatoriamente um nome de cliente)
Trigger (manter sigla sempre Maiúscula)
PROFISSOES
P * IDPROFISSAO NUMBER (4)
DESCRICAOPROFISSAO VARCHAR2 (40)
CIDADES
P * IDCIDADE NUMBER (4)
F * IDESTADO NUMBER (3)
NOMECIDADE VARCHAR2 (70)
EDITORAS
P * IDEDITORA NUMBER (5)
F * IDCIDADE NUMBER (4)
NOMEEDITORA VARCHAR2 (50)
Procedimento ExportarEditoras(NomeCidade)
Procedimento ExportarClientes(Profissao) CSV[Nome, cidade, nascimento, rendamensal]

Exercícios

```
CREATE OR REPLACE FUNCTION Iniciais(PNome IN VARCHAR2)
                                        RETURN VARCHAR2 IS
aux VARCHAR2(200);
BEGIN
aux:=Upper(SubStr(PNome,1,1));
FOR i IN 2..Length(PNome)
LOOP
  IF substr(PNome,i-1,1)=' ' THEN
    aux:=aux||Upper(substr(PNome,i,1));
  ELSE aux:=aux||lower(substr(PNome,i,1));
  END IF;
END LOOP;
RETURN aux;
END;
```

Exercícios

```
CREATE OR REPLACE PROCEDURE
QtdeVendidaDiariamente(PMes IN VARCHAR2, PAno IN VARCHAR2) AS
CURSOR VendasDiarias IS SELECT pd.descricaoproduto,
Sum(vd.quantidade) Qtde FROM vendas vd, produtos pd
WHERE vd.idproduto=pd.idproduto
AND To_Char(vd.datavenda,'MM')=PMes
AND To_Char(vd.datavenda,'YYYY')=PAno
GROUP BY pd.descricaoproduto ORDER BY pd.descricaoproduto;
BEGIN
FOR dados IN VendasDiarias
LOOP
  Dbms_Output.put_line(dados.descricaoproduto||' - '||dados.qtde);
END LOOP;
END;
```

Exercícios

```
CREATE OR REPLACE FUNCTION NomeAleatorio RETURN varchar2 IS
temp VARCHAR2(200);
BEGIN
SELECT cl.nomecliente INTO temp FROM clientes cl WHERE idcliente=
Trunc(Dbms_Random.Value((SELECT Min(c1.idcliente) FROM clientes c1),
        (SELECT Max(c1.idcliente) FROM clientes c1)));

RETURN temp;
END;
```

Exercícios

```
CREATE OR REPLACE PROCEDURE ConsultaVendedores AS
CURSOR AnoVendedor IS SELECT DISTINCT
To_Char(ve.nascimento,'YYYY') Ano FROM vendedores ve
ORDER BY Ano;
CURSOR NomesVendedores(PAno IN VARCHAR2) is
SELECT vd.nomevendedor FROM vendedores vd
WHERE To_Char(vd.nascimento,'YYYY')=PAno;
BEGIN
FOR C_Ano IN AnoVendedor
LOOP
  Dbms_Output.put_line(C_Ano.Ano);
  FOR C_Nomes IN NomesVendedores(C_Ano.Ano)
  LOOP
   Dbms_Output.put_line('    ->'||C_Nomes.nomevendedor);
  END LOOP;
END LOOP;
END;
```

Exercícios

```
CREATE OR REPLACE PROCEDURE
ExportarClientes(PProfissao IN VARCHAR2) as
CURSOR ListaClientes IS
SELECT cl.nomecliente,cl.rendamensal,cl.nascimento,ci.nomecidade
FROM clientes cl,cidades ci,profissoes pf
WHERE ci.idcidade=cl.idcidade AND pf.idprofissao=cl.idprofissao
and pf.descricaoprofissao=PProfissao;
l Utl_File.file_type;
BEGIN
  l:=Utl_File.fopen('DESTINO','Clientes.csv','W');
  FOR linha IN ListaClientes
  LOOP
    Utl_File.put_line(l,linha.nomecliente||';'||linha.nomecidade||';'||
linha.nascimento||';'||linha.rendamensal);
  END LOOP;
  Utl_File.fclose(l);
END;
```

Exercícios

```
CREATE OR REPLACE PROCEDURE
ExportarEditoras(PCidade IN VARCHAR2) as
CURSOR ListaEditoras IS SELECT e.nomeeditora,
c.nomecidade FROM editoras e,cidades c
WHERE c.idcidade=e.idcidade AND c.nomecidade LIKE PCidade||'%';

l Utl_File.file_type;
BEGIN
  l:=Utl_File.fopen('DESTINO','Cidades.txt','W');
  FOR linha IN ListaEditoras
  LOOP
    Utl_File.put_line(l,linha.nomeeditora||' - '||linha.nomecidade);
  END LOOP;
  Utl_File.fclose(l);
END;
```

Exercícios

```
CREATE OR REPLACE TRIGGER tid_sigla_uf
before INSERT OR UPDATE ON estados
REFERENCING NEW AS NEW FOR EACH ROW
BEGIN
 :new.sigla:=Upper(:new.sigla);
END;
```