Sumário

[SQL 3](#_Toc78896389)

[Instalar 3](#_Toc78896390)

[Configurar 3](#_Toc78896391)

[Seção 1- Criação das Tabelas e Comandos Iniciais 4](#_Toc78896392)

[Seção 2 -> Tipos de Dados 5](#_Toc78896393)

[Seção 4 – Funções Básicas 5](#_Toc78896394)

[Single-Row 5](#_Toc78896395)

[Datas 5](#_Toc78896396)

[#Diferença de dias é considerado também as horas. Fazer o trunc no resultado. 5](#_Toc78896397)

[--Somando horas em uma data 5](#_Toc78896398)

[--Somar minutos 6](#_Toc78896399)

[--Somar Segundos 6](#_Toc78896400)

[--Proxima data a partir de uma dia da semana 6](#_Toc78896401)

[--Ultimo dia do mes 6](#_Toc78896402)

[--Primeiro dia do mes 6](#_Toc78896403)

[--Primeiro dia do proximo mês 6](#_Toc78896404)

[---Formatação de data 6](#_Toc78896405)

[--Busca o nome do mês 6](#_Toc78896406)

[--Busca o nomed do dia do mês 6](#_Toc78896407)

[--Nome, em extenso, do Ano 6](#_Toc78896408)

[--Concatenar String no To\_char . 6](#_Toc78896409)

[--Convertendo números no TO\_CHAR 6](#_Toc78896410)

[Seção 7 – SubConsultas 7](#_Toc78896411)

[Exemplos 1: 7](#_Toc78896412)

[Exemplo 2: 7](#_Toc78896413)

[Exemplo 3: 7](#_Toc78896414)

[Seção 8: Variáveis de Substituição e SavePoint 7](#_Toc78896415)

[Seção 10 – Constraints: 8](#_Toc78896416)

[--Desabilitando/Habilitando constraint 8](#_Toc78896417)

[--Excluir Constraint 8](#_Toc78896418)

[Seção 11 – Views (Visões) 8](#_Toc78896419)

[Ex: 8](#_Toc78896420)

[--View com parametros de saida 8](#_Toc78896421)

[--View Complexa 8](#_Toc78896422)

[--OPERACAO DML NA VIEW 9](#_Toc78896423)

[--View somente leitura (Nao permite DML) 9](#_Toc78896424)

[Seção 13 - Sequence 9](#_Toc78896425)

[--Valor Atual 9](#_Toc78896426)

[PL/SQL 10](#_Toc78896427)

[1 – Corpo de um bloco Anônimo, Procedure e Função. 10](#_Toc78896428)

[Exemplo Procedure: 10](#_Toc78896429)

[Declarando variáveis: 10](#_Toc78896430)

[Tipos de Variáveis: 10](#_Toc78896431)

[2 – Exemplo utilizando o Dbms\_Output.Put\_Line 11](#_Toc78896432)

[SET SERVEROUTPUT ON 11](#_Toc78896433)

[Bind variable 11](#_Toc78896434)

[3 – Quantidade de Linhas Afetadas: 12](#_Toc78896435)

[4 – Loops 12](#_Toc78896436)

[Loop 12](#_Toc78896437)

[For Loop 12](#_Toc78896438)

[While Loop 13](#_Toc78896439)

[5 – Records 13](#_Toc78896440)

[Ex 1: 13](#_Toc78896441)

[Ex 2: 13](#_Toc78896442)

[6 – Collection 14](#_Toc78896443)

[Associative Arrays (Index-by-tables)-> Tamanho variável 15](#_Toc78896444)

[Nested Tables -> Tamanho variável 15](#_Toc78896445)

[Varrays -> Semelhante a matriz. 16](#_Toc78896446)

[7 – Cursores Explícitos 17](#_Toc78896447)

[8 – Tratamento de Erros (Exceções): 18](#_Toc78896448)

[9 – Procedures 19](#_Toc78896449)

[PL/SQL AVANÇADO 19](#_Toc78896450)

# SQL

## Instalar

Oracle 11xe.

Ir em Setup.exe -> Executar como Administrador -> Escolher a senha ->

SqlTools

## Configurar

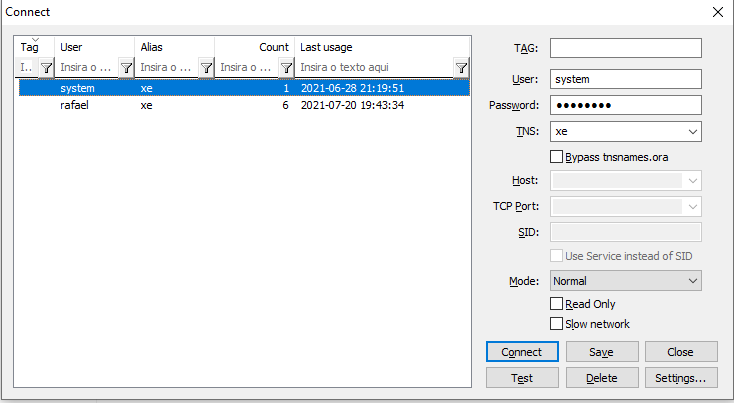
Oracle Cliente

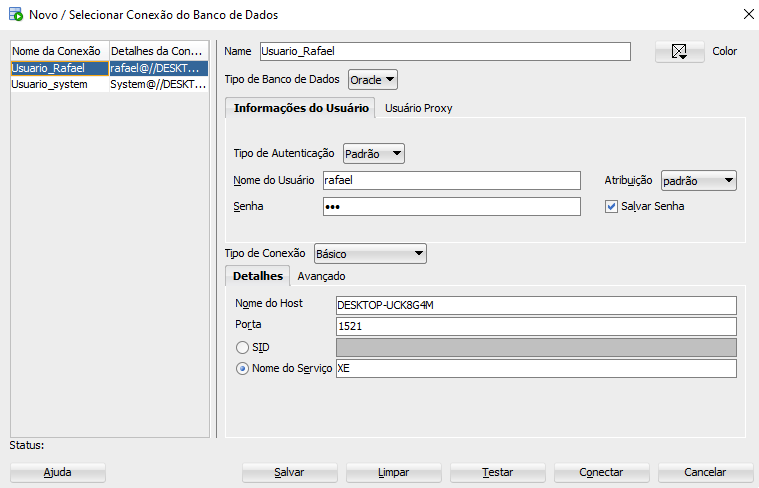
Copiar o Client e ir em C://OracleExe/Client

Networl/admin/tnsadmin.ora -> tem que criar esse arquivo para o SqlTools conseguir acessar o banco.

Criar uma variável de ambiente, em Path, contendo o caminho do cliente(C://OracleExe/Client).

Após isso, em SqlTools , só inserir as informações para conectar ao banco.





Ambiente de Desenvolvimento(Tablespace)

create tablespace tbs\_dados

datafile 'C:\oraclexe\app\oracle\oradata\tbd\_dados.dbf' size 100M reuse

autoextend on next 10M maxsize 200M

online;

#Criar um usuário vinculado ao Tablespace

create user Rafael --usuario

identified by "123" --senha

default tablespace tbs\_dados

temporary tablespace temp;

#Permisionamento

-- Cria e define a "role" de privilegios (perfil)

create role perfil\_desenv;

grant

create cluster,

create database link,

create procedure,

create session,

create sequence,

create synonym,

create table,

create any type,

create trigger,

create view

to perfil\_desenv;

grant alter session to perfil\_desenv;

grant create trigger to rafael;

grant perfil\_desenv to rafael;

grant unlimited tablespace to rafael;

## Seção 1- Criação das Tabelas e Comandos Iniciais

CREATE TABLE TALUNO(

COD\_ALUNO INTEGER NOT NULL,

NOME VARCHAR(30),

CIDADE VARCHAR2(30),

CEP VARCHAR(10),

PRIMARY KEY (COD\_ALUNO)

);

CREATE TABLE TCURSO

( COD\_CURSO INTEGER NOT NULL PRIMARY KEY,

NOME VARCHAR2(30),

VALOR NUMBER(8,2),

CARGA\_HORARIA INTEGER

);

CREATE TABLE TCONTRATO

( COD\_CONTRATO INTEGER NOT NULL PRIMARY KEY,

DATA DATE,

COD\_ALUNO INTEGER,

TOTAL NUMBER(8,2),

DESCONTO NUMBER(5,2)

);

CREATE SEQUENCE SEQ\_ALUNO START WITH 4;

CREATE TABLE TITEM

( COD\_ITEM INTEGER NOT NULL PRIMARY KEY,

COD\_CURSO INTEGER,

COD\_CONTRATO INTEGER,

VALOR NUMBER(8,2) );

## Seção 2 -> Tipos de Dados

INTEGER - 1, 2 -- numero inteiro -> number(38)

NUMBER(5,2) - 999,99

NUMERIC(5,2) - 999,99

DATE - '10/03/2011 00:00:00'

VARCHAR(10) -- Sinonimo

VARCHAR2(10) - 'MARCIO'

CHAR(10) - 'MARCIO ‘

## Seção 4 – Funções Básicas

### Single-Row

Concat -> SELECT Concat(COD\_ALUNO,NOME) FROM TALUNO;

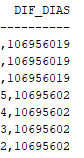
Instr -> SELECT nome, InStr(NOME,'R') FROM TALUNO; //Primeira ocorrência de determinada letra.

Subtr -> SELECT SubStr(NOME,Length(nome)-1, 2) FROM TALUNO;

### Datas

#### #Diferença de dias é considerado também as horas. Fazer o trunc no resultado.

SELECT SYSDATE - DATA AS DIF\_DIAS FROM TCONTRATO;



#### --Somando horas em uma data

SELECT SYSDATE, SYSDATE + 5 / 24 as ADD\_HORAS FROM TCONTRATO;

#### --Somar minutos

SELECT SYSDATE, SYSDATE + 15 / 1440 as ADD\_MINUTOS FROM TCONTRATO;

#### --Somar Segundos

SELECT SYSDATE, SYSDATE + 30 / (3600 \* 24) as ADD\_SEGUNDOS FROM TCONTRATO;

#### --Proxima data a partir de uma dia da semana

SELECT Next\_Day(SYSDATE, 'QUARTA-FEIRA') AS PROXIMA\_QUARTA\_DATA FROM DUAL;

#### --Ultimo dia do mes

SELECT Last\_Day(SYSDATE) AS ULTIMO\_DIA\_MES FROM DUAL;

#### --Primeiro dia do mes

SELECT Trunc(SYSDATE,'MONTH') AS PRIMEIRO\_DIA\_MES\_CORRENTE FROM DUAL;

#### --Primeiro dia do proximo mês

--Até dia 15 do mes pega o primeiro dia do mes atual,--a partir do dia 16 retorna o primeiro dia do proximo mes

SELECT Round(SYSDATE+14, 'MONTH') AS PRIMEIRO\_DIA\_PROXIMO\_MES FROM DUAL;

### ---Formatação de data

#### --Busca o nome do mês

SELECT To\_Char(SYSDATE,'MONTH') MES1 FROM DUAL;

MES1

------------------------------------

AGOSTO

#### --Busca o nomed do dia do mês

SELECT To\_Char(SYSDATE,'DAY') DIA\_SEMANA1 FROM DUAL;

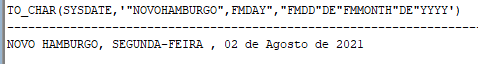
#### --Nome, em extenso, do Ano

SELECT To\_Char(SYSDATE,'YEAR') ANO FROM DUAL;

#### --Concatenar String no To\_char .

o Fm serve para tirar os espaços

SELECT To\_Char(SYSDATE,'"NOVO HAMBURGO", fmDAY "," fmDD "de" fmMonth "de" YYYY') FROM DUAL;



### --Convertendo números no TO\_CHAR

--L -> Coloca o R$

--G -> ponto

--D -> casas decimais

SELECT Salario,Trim(To\_Char(Salario,'L99999.99')) salario1, trim(To\_Char(Salario,'L99G999D99')) salario2 FROM TALUNO;



## Seção 7 – SubConsultas

### Exemplos 1:

SELECT COD\_ALUNO, NOME, CIDADE, ESTADO

FROM TALUNO

WHERE (CIDADE,ESTADO) =

( SELECT CIDADE,ESTADO FROM TALUNO

WHERE COD\_ALUNO = 1 )

AND COD\_ALUNO <> 1 ;

### Exemplo 2:

SELECT COD\_CURSO, Min(VALOR),Sum(VALOR),

Count(\*) QTDE

FROM TITEM

WHERE cod\_curso > 0

GROUP BY COD\_CURSO

HAVING Min(VALOR) >=

(SELECT Avg(VALOR) FROM TCURSO)

ORDER BY Cod\_Curso;

### Exemplo 3:

--SubConsulta na clausula From

SELECT ITE.COD\_CONTRATO, ITE.VALOR, ITE.COD\_CURSO,

CUR.COD\_CURSO codigo, CUR.VALOR

FROM TITEM ITE,

( SELECT COD\_CURSO, VALOR

FROM TCURSO WHERE VALOR > 500 ) CUR

WHERE CUR.COD\_CURSO = ITE.COD\_CURSO

## Seção 8: Variáveis de Substituição e SavePoint

Variáveis de Substituição:

SELECT \* FROM TDESCONTO

WHERE CLASSE = '&cla' and INFERIOR=&inf;

Só executar o comando que aparece uma janela perguntando sobre os dados.

SavePoint -> Ponto de Restauração.

SAVEPOINT upd\_b;

UPDATE TDESCONTO SET

SUPERIOR = 88

WHERE CLASSE = 'B';

--ponto de restauraçao

SAVEPOINT ins\_Ok;

--

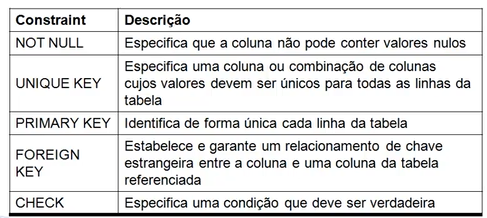
INSERT INTO tdesconto(classe, inferior, superior) VALUES ('&cla', &inf, &sup);

ROLLBACK TO SAVEPOINT ins\_Ok;

ROLLBACK TO SAVEPOINT upd\_a;

ROLLBACK TO SAVEPOINT upd\_b;

## Seção 10 – Constraints:



### --Desabilitando/Habilitando constraint

ALTER TABLE TPESSOA DISABLE CONSTRAINT uk\_cpf;

ALTER TABLE TPESSOA ENABLE CONSTRAINT uk\_cpf;

### --Excluir Constraint

ALTER TABLE TPESSOA DROP CONSTRAINT uk\_cpf;

## Seção 11 – Views (Visões)

É um SELECT em formato de objeto.

Existem 2 tipos:

Views Simples -> Uma tabela, Não Contém Função, Não Possui Grupos de Dados, DML atráves de visões.

Views Complexas -> Uma ou mais tabelas. Contém Função. Possui grupos de Dados. Pode executar DML.

### Ex:

CREATE OR REPLACE VIEW V\_ALUNO

AS

SELECT COD\_ALUNO AS CODIGO, SALARIO, ESTADO,

NOME AS ALUNO, CIDADE

FROM TALUNO

WHERE ESTADO='RS';

### --View com parametros de saida

CREATE OR REPLACE VIEW V\_ALUNO2(COD, ALUNO, SAL)

AS

SELECT COD\_ALUNO, NOME, SALARIO

FROM TALUNO;

### --View Complexa

**--(nao pode fazer DML em view complexa)**

CREATE OR REPLACE VIEW V\_CONTRATO

AS

SELECT Trunc(DATA) AS DATA,

Max(DESCONTO) MAXIMO,

Avg(DESCONTO) MEDIA,

Count(\*) QTDE

FROM TCONTRATO

GROUP BY Trunc(DATA);

### --OPERACAO DML NA VIEW

CREATE OR REPLACE VIEW vcursos1000ck

AS

SELECT cod\_curso, nome, valor

FROM TCurso

WHERE VALOR = 1000

**WITH CHECK OPTION CONSTRAINT vcursos1000\_ck; --Isso serve para ter um critério de Insert. Só insere caso o VALOR seja igual a 1000.**

INSERT INTO vCursos1000ck

(cod\_curso, nome, valor)

VALUES (52,'TESTE Y', 1000);

### --View somente leitura (Nao permite DML)

CREATE OR REPLACE VIEW V\_ALUNO3

AS

SELECT COD\_ALUNO CODIGO,

NOME ALUNO, CIDADE

FROM TALUNO

WHERE ESTADO='RS'

**WITH READ ONLY**;

## Seção 13 - Sequence

CREATE SEQUENCE SEQ\_ALUNO1

START WITH 60

--Valor Inicial

INCREMENT BY 2

--Qtde a Incrementar

MINVALUE 60

--Valor Minimo

MAXVALUE 100

--Valor Maximo

NOCACHE –Não guarda o Cache. Por padrão é 20.

--nao guarda em cache faixa de valores -- 20

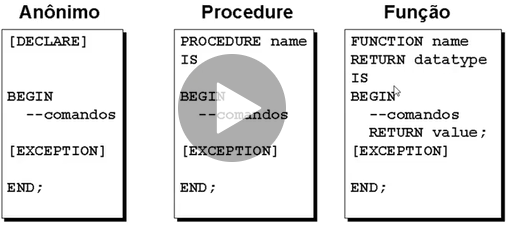
NOCYCLE; -- No Cycle serve para, caso chegue no máximo da sequence, ele reinicia o contador.

#### --Valor Atual

SELECT SEQ\_ALUNO1.CURRVAL FROM DUAL;

## PL/SQL

### 1 – Corpo de um bloco Anônimo, Procedure e Função.



#### Exemplo Procedure:

CREATE OR REPLACE PROCEDURE P\_AUMENTA\_PRECOCURSO

IS

--variaveis

BEGIN

UPDATE TCURSO SET VALOR = VALOR \* 1.20;

END;

--Executa procedure 1

EXECUTE P\_AUMENTA\_PRECOCURSO;

--Executa procedure 2 - Bloco anonimo

BEGIN

P\_AUMENTA\_PRECOCURSO;

END;

#### Declarando variáveis:

vData Date;

vPreco Number(2) NOT NULL :=10;

vCidade varchar2(60) := ‘Novo Hamburgo’;

cDesconto NUMBER := 100;

#### Tipos de Variáveis:

Long -> Dados de Caractéres do tamanho variável de até 2GB.

Long Row -> Dados Binários e strings do tipo byte de até 32760 bytes.

Não são interpretados pelo PL/SQL.

Binary\_float -> Ponto flutuante com precisão simples.

Binary\_double -> Ponto flutuante com precisão dupla.

Binary\_integer -> Tipo básico para inteiro.

PLS\_INTEGER -> Tipo básico para inteiro com sinal.

São mais rápidos do que NUMBER e BINARY\_INTEGER.

Atributo %Type -> Utilizar o tipo da variável de uma tabela.

vNome tcliente.nome%TYPE;

Boolean -> vFlag BOOLEAN := (vPreco < vValor)

Record -> Similar a um Array.

Collection ->

Lob ->

CLOB = Texto

BLOB = Foto

BFILE = Filme

NCLOB = NCLOB

Variáveis não PL/SQL -> São mais performáticos, a partir da segunda execução.

:gPReco\_final := vPreco \* 0.85;

### 2 – Exemplo utilizando o Dbms\_Output.Put\_Line

#### SET SERVEROUTPUT ON

DECLARE

VDESCONTO NUMBER(6,2) := 0.50;

VCIDADE VARCHAR(30) := 'NOVO HAMBURGO';

VCOD\_ALUNO TALUNO.COD\_ALUNO%TYPE := 5;

VTOTAL NUMBER(8,2) := 1345.89;

vFlag BOOLEAN;

BEGIN

VTOTAL := Round(VTOTAL \* VDESCONTO, 2);

vFlag := (VCOD\_ALUNO>6);

IF (vFlag=True) THEN --Se vFlag = True Entao

Dbms\_Output.Put\_Line('Verdadeiro');

ELSE --Senao

Dbms\_Output.Put\_Line('Falso');

END IF;

Dbms\_Output.Put\_Line('Total: '|| vTotal);

VDESCONTO := 1.20;

vCIDADE := InitCap(VCIDADE);

Dbms\_Output.Put\_Line('Cidade: '||vCidade);

Dbms\_Output.Put\_Line('Desconto: '||VDESCONTO);

Dbms\_Output.Put\_Line('Aluno: '||VCOD\_ALUNO);

END;

#### Bind variable

--Variável fora do Contexto. É mais rápido

-- Utilizar o “:”. Isso ocorre porque ele identifica que não precisa recompilar novamente esta variável.

VARIABLE vDESCONTO2 NUMBER

DECLARE

VCOD\_ALUNO NUMBER := 1;

BEGIN

:vDESCONTO2 := 0.90;

Dbms\_Output.put\_line('desconto 2: '||:vDESCONTO2);

UPDATE TContrato SET

TOTAL = TOTAL \* :vDESCONTO2

WHERE COD\_ALUNO = VCOD\_ALUNO;

END;

### 3 – Quantidade de Linhas Afetadas:

SET SERVEROUTPUT ON

DECLARE

vContrato NUMBER := &cod\_contrato;

vtexto VARCHAR2(50);

BEGIN

UPDATE TContrato SET

desconto = desconto + 2

WHERE Cod\_Contrato = VContrato;

**vtexto := SQL%ROWCOUNT**;

--Retorna qtde de registros afetados

--pelo comando sql

Dbms\_Output.Put\_Line(vtexto|| ' linhas atualizadas.');

END;

### 4 – Loops

#### Loop

--Laço de repetição 1 até 10

DECLARE

vContador INTEGER := 0;

BEGIN

**LOOP**

vContador := vContador + 1;

Dbms\_Output.Put\_Line(vContador);

EXIT WHEN vContador = 10;

**END LOOP**;

Dbms\_Output.Put\_Line('Fim do LOOP');

END;

#### For Loop

--For loop -> mais indicado para laços em tabelas

DECLARE

vContador INTEGER;

**BEGIN**

**FOR vContador in 1..10**

**LOOP**

--vContador := vContador + 1;

Dbms\_Output.Put\_Line(vContador);

--EXIT WHEN vContador = 5;

**END LOOP;**

END;

#### While Loop

DECLARE

vContador INTEGER := 0;

vTexto VARCHAR(10);

BEGIN

**WHILE vContador < 10**

**LOOP**

vContador := vContador + 1;

IF (vContador Mod 2)=0 THEN

vTexto := 'Par';

ELSE

vTexto := 'Impar';

END IF;

Dbms\_Output.Put\_Line(vContador|| ' -> '||vTexto);

**END LOOP;**

END;

### 5 – Records

Grupo de Itens de Dados.

Como se fosse uma tabela.

#### Ex 1:

DECLARE

TYPE Rec\_aluno IS RECORD

( cod\_aluno NUMBER NOT NULL := 0,

nome TALUNO.Nome%TYPE,

cidade TALUNO.Cidade%TYPE );

Registro rec\_aluno;

BEGIN

registro.cod\_aluno := 50;

registro.nome := 'Master';

registro.cidade := 'Novo Hamburgo';

Dbms\_Output.Put\_Line('Codigo: '||registro.cod\_aluno);

Dbms\_Output.Put\_Line(' Nome: '||registro.nome);

Dbms\_Output.Put\_Line('Cidade: '||registro.cidade);

END;

#### Ex 2:

DECLARE

reg TAluno%ROWTYPE; --Record

vcep VARCHAR(10) := '98300000';

BEGIN

SELECT COD\_ALUNO, NOME, CIDADE

INTO Reg.cod\_aluno, Reg.nome, Reg.cidade

FROM TALUNO

WHERE COD\_ALUNO = 1;

vCep := '93500000';

reg.cep := vCep;

Dbms\_Output.Put\_Line('Codigo: ' ||reg.cod\_aluno);

Dbms\_Output.Put\_Line('Nome : ' ||reg.nome);

Dbms\_Output.Put\_Line('Cidade: ' ||reg.cidade);

Dbms\_Output.Put\_Line('Cep : ' ||reg.cep);

END;

### 6 – Collection



DECLARE

TYPE nome\_type IS TABLE OF taluno.nome%TYPE;

nome\_table nome\_type := nome\_type(); --Criando Instancia

BEGIN

nome\_table.EXTEND; -- alocando uma nova linha

nome\_table(1) := 'Marcelo';

nome\_table.EXTEND; -- alocando uma nova linha

nome\_table(2) := 'Marcio';

Dbms\_Output.Put\_Line('Nome 1: '||nome\_table(1));

Dbms\_Output.Put\_Line('Nome 2: '||nome\_table(2));

END;

Gerenciamento de múltiplas linhas de dados. Ex: Vetores

Tipos:

#### Associative Arrays (Index-by-tables)-> Tamanho variável

Não necessita da utilização do método EXTEND para alocação de novos elementos.

Pode ser indexado com qualquer valor.

Obrigatória a cláusula INDEX BY

##### Ex 1:

DECLARE

TYPE T\_ALUNO IS TABLE OF TALUNO.NOME%TYPE

INDEX BY BINARY\_INTEGER; --Matriz

REGISTRO T\_ALUNO; --Record

teste INTEGER;

tetste2 INTEGER;

BEGIN

REGISTRO(1) := 'MARCIO';

REGISTRO(2) := 'JOSE';

REGISTRO(3) := 'PEDRO';

teste:=REGISTRO.count();

FOR tetste2 IN 1..REGISTRO.count()

LOOP

Dbms\_Output.Put\_Line('Nome 1: '||registro(tetste2));

-- Dbms\_Output.Put\_Line('Nome 2: '||registro(2));

-- Dbms\_Output.Put\_Line('Nome 3: '||registro(3));

END LOOP;

END;

#### Nested Tables -> Tamanho variável

##### Ex 1:

DECLARE

TYPE T\_ALUNO IS TABLE OF TALUNO%ROWTYPE

INDEX BY BINARY\_INTEGER; --Matriz

C NUMBER;

REGISTRO T\_ALUNO; --Record

BEGIN

C:=1;

for cada\_linha in (Select \* from TALUNO) LOOP

REGISTRO(C).COD\_ALUNO := cada\_linha.COD\_ALUNO;

REGISTRO(C).NOME := cada\_linha.NOME;

REGISTRO(C).CIDADE := cada\_linha.CIDADE;

REGISTRO(C).CEP := cada\_linha.CEP;

REGISTRO(C).ESTADO := cada\_linha.ESTADO;

REGISTRO(C).SALARIO := cada\_linha.SALARIO;

REGISTRO(C).NASCIMENTO := cada\_linha.NASCIMENTO;

C:=C+1;

END LOOP;

FOR cada\_linha IN 1..REGISTRO.Count()

LOOP

Dbms\_Output.Put\_Line('Cod\_Aluno : '||registro(cada\_linha).COD\_ALUNO);

Dbms\_Output.Put\_Line('Nome : '||registro(cada\_linha).NOME);

END LOOP;

END;

##### Ex 2:

DECLARE

TYPE tipo IS TABLE OF VARCHAR2(30) INDEX BY VARCHAR2(2);

--

uf\_capital tipo;

BEGIN

uf\_capital('RS') := 'PORTO ALEGRE';

uf\_capital('RJ') := 'RIO DE JANEIRO';

uf\_capital('PR') := 'CURITIBA';

uf\_capital('MT') := 'CUIABA';

dbms\_output.put\_line(uf\_capital('RS'));

dbms\_output.put\_line(uf\_capital('RJ'));

dbms\_output.put\_line(uf\_capital('PR'));

dbms\_output.put\_line(uf\_capital('MT'));

END;

#### Varrays -> Semelhante a matriz.

DECLARE

TYPE nome\_varray IS VARRAY(5) OF taluno.nome%TYPE;

nome\_vetor nome\_varray := nome\_varray();

BEGIN

nome\_vetor.EXTEND;

nome\_vetor(1) := 'Master';

Dbms\_Output.Put\_Line(nome\_vetor(1));

END;

### 7 – Cursores Explícitos

Declare -> Open -> Fetch -> Vaz

%ISOPEN -> Retorna se o Cursor estiver aberto

%NOTFOUND -> Retorna TRUE se o fetch mais recente não retornou uma linha.

%FOUND -> Contrário do NOTFOUND

%ROWCOUNT -> Retorna o número total de linhas recuperadas até o momento.

##### Exemp 1 – Cursor para recuperar campos:

DECLARE

vcod\_aluno TAluno.Cod\_Aluno%TYPE;

vNome TAluno.nome%TYPE;

CURSOR c1 IS

SELECT cod\_aluno, nome

FROM taluno;

BEGIN

OPEN c1; -- abre cursor

LOOP

FETCH c1 INTO vCod\_Aluno, vNome; --pega registro atual

EXIT WHEN c1%ROWCOUNT > 10 OR c1%NOTFOUND;

Dbms\_Output.Put\_Line('Codigo: '||

LPad(vcod\_aluno,4,'0')||' - '||'Nome: '||vNome);

END LOOP;

CLOSE c1; --fecha cursor

END;

##### Exemp 2 – Cursor para recuperar vários campos:

DECLARE

CURSOR c1 IS

SELECT \* FROM TAluno;

Reg c1%ROWTYPE; --record

BEGIN

OPEN c1; --

LOOP

FETCH c1 INTO reg;

EXIT WHEN c1%ROWCOUNT > 10 OR c1%NOTFOUND;

Dbms\_Output.Put\_Line('Codigo: '||

LPad(reg.cod\_aluno,5,'0')||'-'||

'Nome: '||reg.nome);

END LOOP;

CLOSE c1; --

END;

##### Exemp 3- Cursos passando parâmetro

DECLARE

CURSOR c1 (pCod\_aluno NUMBER) IS

SELECT \* FROM TAluno

WHERE Cod\_aluno = pCod\_aluno

FOR UPDATE OF NOME NOWAIT;--bloquia

--bloquea a coluna nome para alteracao

Reg c1%ROWTYPE;

BEGIN

OPEN c1(&codigo);

FETCH c1 INTO reg;

Dbms\_Output.Put\_Line(reg.cod\_aluno ||' - ' || reg.nome);

CLOSE c1; --libera o registro para alteracao

END;

##### Exemp 4- Cursos travando tabela

DECLARE

CURSOR c1 IS

SELECT \* FROM TALUNO

FOR UPDATE;

Reg\_aluno c1%ROWTYPE;

BEGIN

FOR reg\_aluno IN c1

LOOP

UPDATE TALUNO

SET nome = InitCap(reg\_aluno.nome)

WHERE CURRENT OF c1; --bloqueia somente o reg atual

Dbms\_Output.Put\_Line('Nome: '||InitCap(reg\_aluno.nome));

END LOOP;

COMMIT;

END;

### 8 – Tratamento de Erros (Exceções):

##### Exemplo 1 – Exceções por padrão:

DECLARE

vCod taluno.cod\_aluno%TYPE := 566;

vCidade taluno.cidade%TYPE; x NUMBER;

BEGIN

SELECT Cidade INTO vCidade

FROM TAluno

WHERE nome LIKE '%';

--WHERE cod\_aluno = vCod;

X := 0 / 0;

Dbms\_Output.Put\_Line('Cidade: '||vCidade);

EXCEPTION

WHEN no\_data\_found THEN

RAISE\_APPLICATION\_ERROR(-20001,

'Aluno Inexistente! '||SQLCODE||' '||SQLERRM);

WHEN too\_many\_rows THEN

RAISE\_APPLICATION\_ERROR(-20002,

'Registro Duplicado! '||SQLCODE||' '||SQLERRM);

WHEN others THEN

RAISE\_APPLICATION\_ERROR(-20003,

'Exceção Desconhecida! '||SQLCODE||' '||SQLERRM);

END;

##### Exemplo 2 – Exceções desenvolvidas manualmente

DECLARE

vDt\_vencimento DATE;

vConta NUMBER := 100; --codigo da conta

**eConta\_vencida EXCEPTION;**

BEGIN

SELECT vencimento INTO vDt\_vencimento

FROM CONTAS WHERE codigo = vConta;

IF vDt\_vencimento < TRUNC(SYSDATE) THEN

RAISE eConta\_vencida;

END IF;

EXCEPTION

**WHEN eConta\_vencida THEN**

Dbms\_Output.Put\_Line('Conta vencida');

UPDATE contas SET valor = valor + juros

WHERE codigo = vConta;

END;

### 9 – Procedures

Parâmetros devem ser declarados como um tipo de dado escalar (varchar2, number, etc..) sem precisão ou com um tipo %TYPE ou %RowType.

3 Tipos de Parâmtros:

IN

OUT

IN OUT

# PL/SQL AVANÇADO