Introdução ao PL/SQL

(*Procedure Language/Structured Query Language*)

MSc. Eugénio Alberto Macumbe

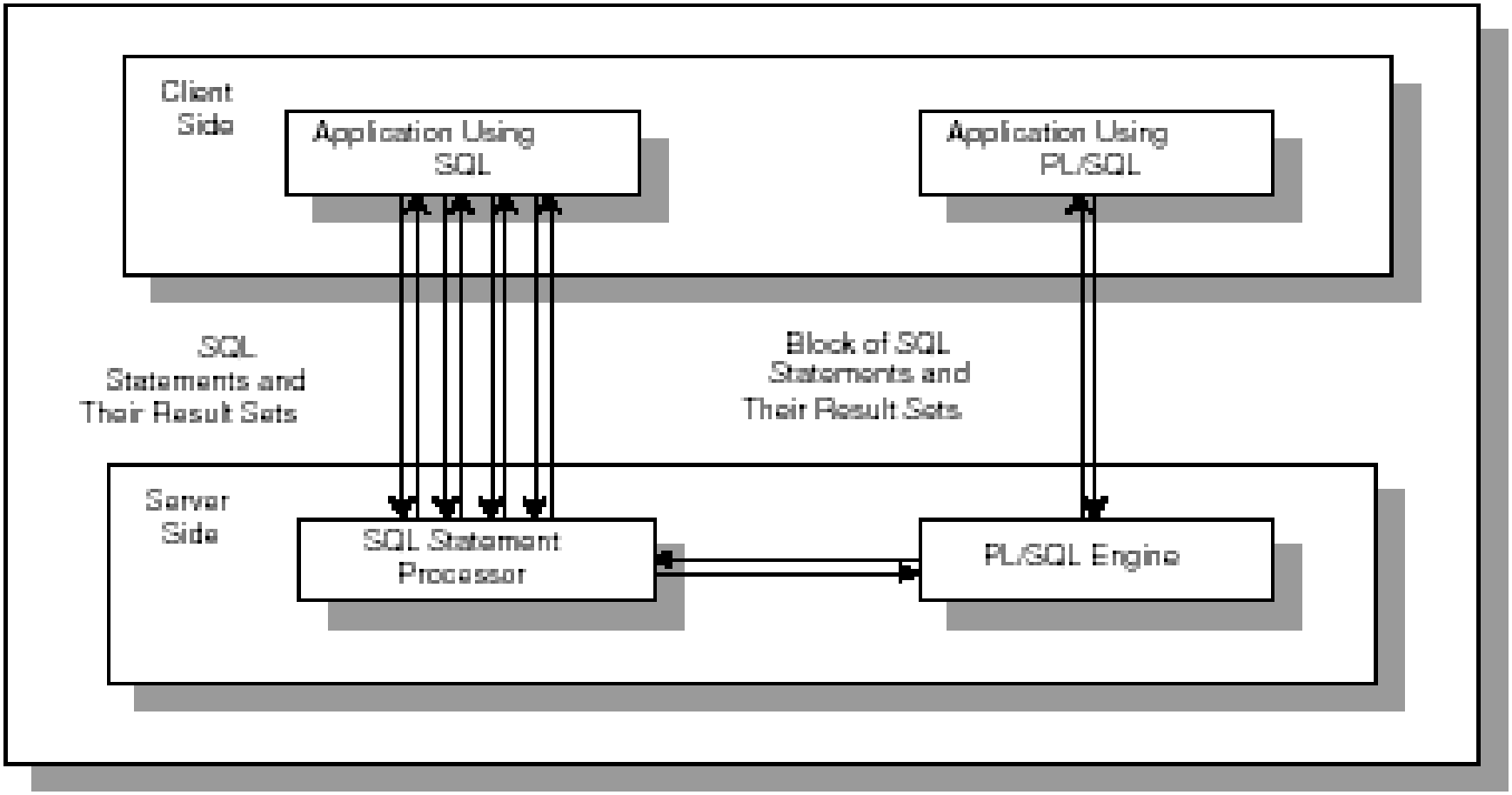
Nel Chamo

O que iremos discutir?....

PL/SQL

- Bloco genérico PL/SQL;
- Tipos de Dados;
- Variáveis e Constantes;
- Operadores;
- Estruturas de controle;
- Literais.

PL/SQL - Comparação



PL/SQL - Vantagens

- Suporte a declarações e manipulação de objectos e colecções;
- Uso de funções e procedimentos externos;
- Uso de bibliotecas pré-embutidas;
- Portabilidade e Reaproveitamento do código;
- Optimização da combinação dos comandos SQL;
- Redução do custo de manutenção e alteração da aplicação;

PL/SQL – Bloco genérico

DECLARE (opcional)

- Declaração de variáveis, Constantes, etc.

BEGIN (obrigatório)

- SQL
- PL/SQL

EXCEPTION (opcional)

- Acção a executar perante a um erro

END(obrigatório);

PL/SQL -Tipos

- Nomeado
  - Usado na criação de subrotinas (procedimentos, funções, pacotes);
  - São armazenadas na DB e chamadas pelo nome;
- Anónimo
  - Não tem nomes e não são armazenadas na DB;
  - **SET** SERVEROUTPUT **ON** (SQL PLUS somente) para visualizar o output**;**

PL/SQL -Tipos

Anónimo

```
DECLARE
BEGIN
    -comandos
EXCEPTION
END;
```

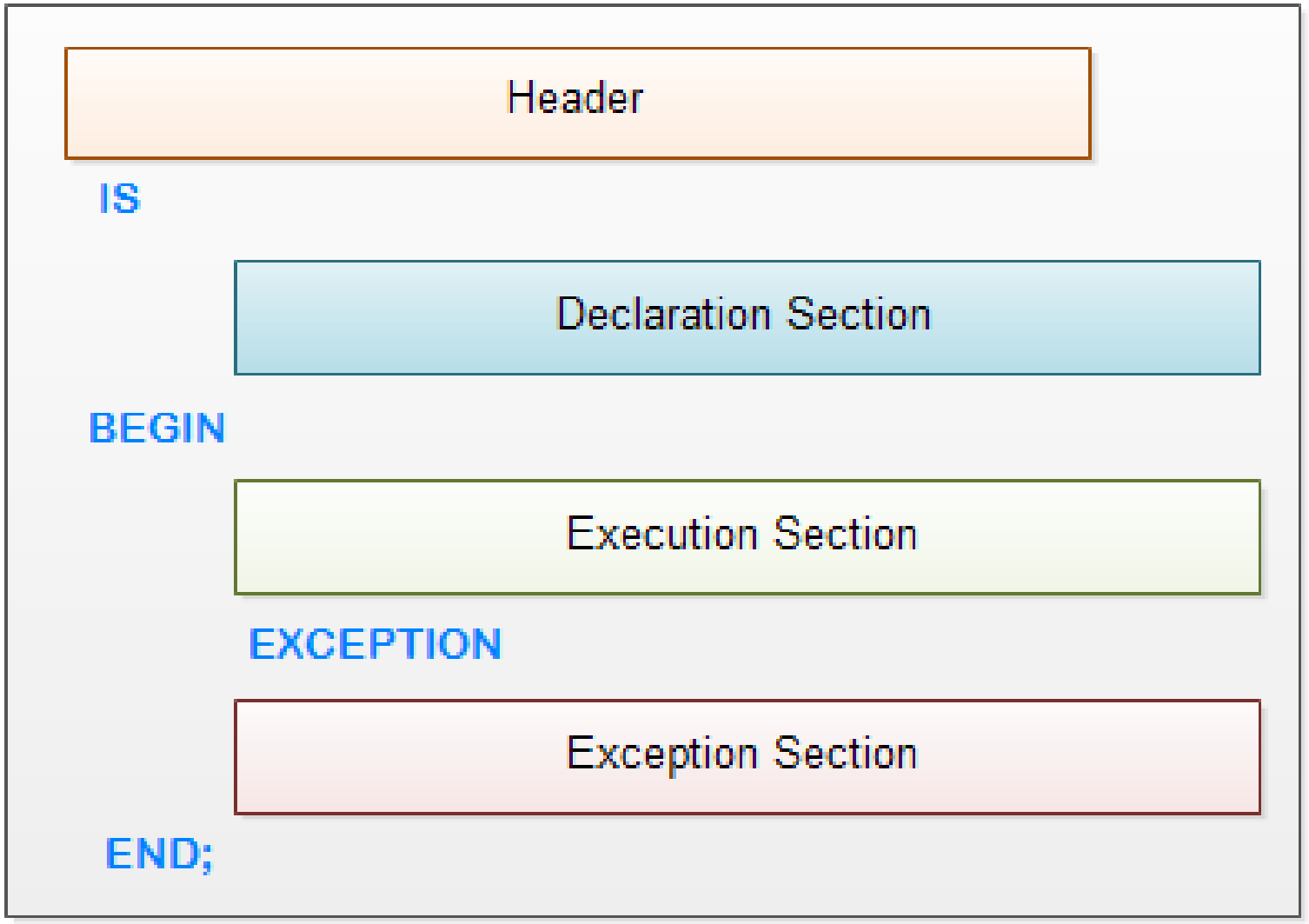
Procedimento

```
PROCEDURE <nome>
IS
BEGIN
    - comandos
EXCEPTION
END;
```

Função

```
FUNCTION <nome>
RETURN <datatype>
IS
BEGIN
    - comandos
EXCEPTION
END;
```

PL/SQL -Tipos


Header
IS
Declaration Section
BEGIN
Execution Section
EXCEPTION
Exception Section
END;

PL/SQL – Declare

- É a primeira secção;
- Contém a definição de variáveis, contantes, cursores, excepções user-defined, etc.

```
DECLARE
        v_primeiro_nome VARCHAR2(40);
        v_ultimo nome VARCHAR2(80);
        v_saldo NUMBER:=0;
```

PL/SQL – Declare

- **%type**
  - Referencia ao tipo de dado de uma dada coluna em certa tabela;
  - Ex: v_saldo estudante.saldo%type;
- **%rowtype**
  - Referencia os nomes e tipos de dados das colunas de uma dada tabela;
  - Ex: v_estudante estudante%rowtype;

PL/SQL – Declare

- **Definição:**

<nome_variável> [constant] <data_type> [not null] [:= <value/expression>]

Ex:

```
Declare
        v_dia_hoje date;
        v_nome varchar2(100):=‘John Doe’;
        v_aumento constant number(15,2):=2000.50;
Begin
…
End;
```

PL/SQL – BEGIN

- É a segunda secção;
- Contém o bloco de instruções para manipular as variáveis e outras funções;

```
BEGIN
        SELECT primeiro_nome, ultimo_nome, saldo
        INTO v_primeiro_nome, v_ultimo_nome, v_saldo
        FROM estudante
        WHERE id = 1234;
        DBMS_OUTPUT.PUT_LINE(' Estudante: ' || v_primeiro_nome || '
        '||v_ultimo_nome);
END;
```

PL/SQL – BEGIN

- Estruturas de controle
  - Condicional (IF)
  - Interactiva (LOOP, WHILE, FOR LOOP)

PL/SQL – BEGIN

- Estruturas de controle
  - Condicional (IF)

```
if <condição> then
    <comandos>
end if;
```

PL/SQL – BEGIN

- Estruturas de controle
  - Interactiva (LOOP, WHILE, FOR LOOP)

```
Loop
    <comandos>
End loop;
```

```
While <condicao>
Loop
    <comandos>
End loop;
```

```
For <contador> in
<inicio> .. <fim>
Loop
    <comandos>
End loop;
```

PL/SQL – BEGIN

- Ex (Loop):

```
declare
        v_a number:=100;
Begin
        loop
                v_a:=v_a+25;
                exit when v_a=250;
        end loop;
        dbms_output.put_line('O valor de v_a : ' || v_a);
End;
/
```

PL/SQL – BEGIN

- Ex (For loop):

```
BEGIN
        FOR v_estudante IN (SELECT * FROM estudante)

        LOOP

        update estudante set email = v_estudante.nome || '@up.ac.mz' where id = v_estudante.id;

        END LOOP;

END;
/
```

PL/SQL – BEGIN

- **Funções STRING comuns:**
  - LOWER(string)
  - SUBSTR(string,start,substrlength)
  - LTRIM(string)
  - RTRIM(string)
  - LPAD(string_to_be_padded, spaces_to_pad, |string_to_pad_with|)
  - RPAD(string_to_be_padded, spaces_to_pad, |string_to_pad_with|)
  - REPLACE(string, searchstring, replacestring)
  - UPPER(string)
  - INITCAP(string)
  - LENGTH(string)

PL/SQL – BEGIN

- **Funções NUMERIC comuns:**
  - ABS(value)
  - ROUND(value, precision)
  - MOD(value,divisor)
  - SQRT(value)
  - TRUNC(value,|precision|)
  - LEAST(exp1, exp2…)
  - GREATEST(exp1, exp2…)

PL/SQL – BEGIN

- **Comentários:**
  - São usados os símbolos:
    - **Multilinhas**: Abrir (/*) e fechar(*/)
    - **Única linha**: --

PL/SQL – EXCEPTION

- É a última secção;
- Contém o bloco de instruções a serem executadas perante a ocorrência de algum tipo de erro;
- Existem as excepções predefinidas e as excepções user-defined;

```
EXCEPTION
        WHEN <nome da excepção> THEN
        comandos a executar;
END;
```

PL/SQL – EXCEPTION

No_data_found	This exception is raised when select statement returns no rows.
Zero_divide	This exception is raised when we try to divide a number by zero.
Cursor_already_open	This exception is raised when we try to open a cursor which is already opened.
Storage_error	This exception is raised if PL/SQL runs out of memory or if the memory is corrupted.
Too_many_rows	Raised when the select into statement returns more than one row.

PL/SQL – EXCEPTION

Invalid_number	This exception is raised if the conversion of a character string to a number fails because the string does not represent a valid number. For example, inserting 'john' for a column of type number will raise this exception.
Dup_val_on_index	This exception is raised when we insert duplicate values in a column, which is defined as unique index.
Program_error	This exception is raised if PL/SQL has an internal problem.
Invalid_cursor	This exception is raised when we violate cursor operation. For example, when we try to close a cursor which is not opened.
Login_denied	This exception is raised raised when we try to enter Oracle using invalid username/password.

PL/SQL – EXCEPTION

EXEMPLOS:

- **PREDEFINIDA**

```
EXCEPTION
        WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Estudante não encontrado!');
END;
```

PL/SQL – EXCEPTION

- **USER-DEFINED:**

```
DECLARE
        tem_divida exception;
        v_saldo estudante.saldo%type;
BEGIN
        select saldo into v_saldo from estudante where id = ‘1234’;
        if v_saldo > 0 then
                raise tem_divida
        end if;
EXCEPTION
        WHEN tem_divida THEN
        DBMS_OUTPUT.PUT_LINE('Estudante tem divida de '|| v_saldo);
END;
```

Quando usar PL/SQL?

•Quando é algo muito complexo para resolver-se com apenas SQL;

•Quando existe necessidade de verificações e rotinas.

Exercício

Dada a tabela :

Estudante (**estudante_id**, primeiros_nomes, apelido, data_nascimento);

1.Crie a tabela e insira valores arbitrário.

2.Modifique a tabela, adicione o atributo email e escreva um programa PL/SQL que atribua endereços de email a cada estudante no domínio uniXY.ac.mz

3.As regras devem ser:

- O endereço de email deve ser todo em letras minúsculas;
- A primeira letra do email deve ser a primeira letra do apelido seguida pelas primeiras 9 letras do nome;
- O email não pode possuir mais de 10 caracteres, excluindo o domínio.

  Ex: Para o estudante António com apelido Marrengula seria

  mantonio@uniXY.ac.mz