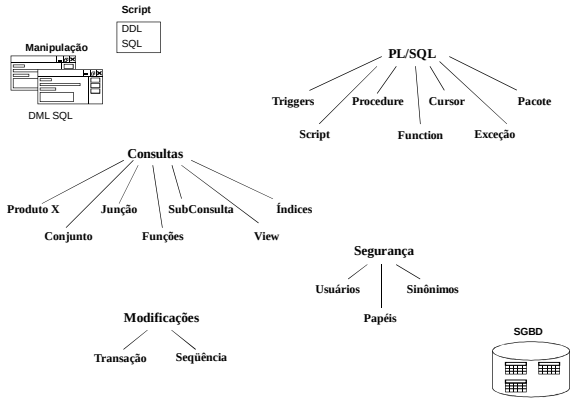
Banco de Dados II
PL/SQL

Osmar de Oliveira Braz Junior



1. Conceitos

- Extensão da linguagem SQL para Oracle
  - **Procedural Language extension to SQL**
  - **DB2 – SQL PL**
  - **SQL Server – Transaction SQL**
  - **Postgress – PL/pgSQL**
- É uma linguagem estruturada usada para a criação de diversos objetos de banco de dados como triggers, stored procedures, functions, packages e scripts para o SQL Plus.
- Trabalha com o conceito de bloco estruturado.
- Pode-se criar um bloco de comandos sem nome específico e executá-lo.

2. Vantagens

- **PORTABILIDADE**
  - Aplicações escritas em PL/SQL são portáveis para qualquer Máquina que rode RDBMS com PL/SQL.
- **INTEGRAÇÃO COM RDBMS**
  - Variáveis PL/SQL podem ser definidas a partir de definições das colunas das tabelas.
  - Redução de manutenção das aplicações, pois estas adaptam-se as mudanças da Base de Dados.
- **CAPACIDADE PROCEDURAL**
  - Comandos de controle de fluxo, comandos de repetições e tratamentos de erros;
- **PRODUTIVIDADE**
  - Desenvolvimento de Database Triggers, Procedures e Functions a nível do Banco de Dados

3.Blocos
3.1. Estrutura Básica

```
DECLARE (Opcional)
  Variáveis, cursores, exceptions definidas
  pelo    usuário
BEGIN (Obrigatório)
     - SQL
     - PL/SQL
     - EXCEPTION(Opcional)
           Ações que são executadas quando
              ocorrem os erros
END(obrigatório);
```


3.Blocos
3.1. Tipos de Blocos

- **Anonymous**

```
Declare
Begin
  .......
  Exception
End;
```

3.Blocos
3.1. Tipos de Blocos

- **Procedure**

```
Procedure name is
Begin
  .......
  Exception
End;
```

7

3.Blocos
3.1. Tipos de Blocos

- **Function**

```
Function name return datatype is
Begin
  .......
  Return value;
  Exception
End;
```

8

4. DataTypes
4.1. Mais Utilizados

- **char(n)-** Tamanho Fixo, pode conter uma seqüência de 1 a 255 bytes alfanuméricos;
- **varchar2(n)-** Tamanho Variável, pode conter uma seqüência de 1 a 2000 bytes - alfanuméricos.
- **Long-** Tamanho Variável até 2 Gigabytes alfanuméricos nota : só pode existir uma coluna long em cada tabela
- **Number(p,s)** - Numérico com sinal e ponto decimal,sendo precisão de 1 a 38 dígitos
- **Raw** - Binário Variável até 255 bytes
- **Long Raw** - Binário Variável até 2 gigabytes – imagem
- **Date** - Data c/ hora, minuto e segundo

9

4. DataTypes
4.2. Exemplo de Declaração

```
DECLARE
      NOME              CHAR(30);
      SALARIO           NUMBER(11,2);
      DEPARTAMENTO      NUMBER(4);
      DATANASC          DATE;
      SIM               BOOLEAN;
      CONT              NUMBER(6) :=0;
      PERC              CONSTANT NUMBER(4,2):= 36.00;
```

10

4. DataTypes
4.3. O atributo %TYPE

- Declara a variável de acordo com uma coluna definida no Banco de Dados;
- **Exemplo**

```
V_nome          empregado.nome%Type;
V_saldo         number(7,2);
V_saldo_min     V_saldo%Type := 10;
```

11

4. DataTypes
4.4. O atributo %ROWTYPE

- Declara uma variável composta equivalente à linha de uma tabela.
- O acesso aos campos da tabela, usa-se o nome da variável seguido de um ponto e do nome do campo
- **Exemplo**

```
DECLARE
  linha   cliente%ROWTYPE;
BEGIN
  Select * into linha from cliente where cliente_id = 1;
  dbms_output.put_line(linha.cliente_id || linha.nome);
END;
```

12

5. Funções

- **Matemática – ABS, ATN, COS, EXP, LN, LOG,POWER ROUND, SIN, SQRT, TAN, TRUNC**
- **Data - ADD_MONTHS, LAST_DAY, MONTHS_BETWEEN, NEXT_DAY, ROUND, SYSDATE, TIME, TRUNC, WEEKDAY**
- **Conversão – TO_CHAR, TO_DATE, TO_NUMBER**
- **Caracteres – ASCII, CHR, CONCAT, INITCAP, INSTR, LOWER, LEFT, LENGTH, LTRIM, REPLACE, RIGHT, RTRIM, STR, STRING, SUBSTR, TRIM, UPPER, VAL**
- **Diversas – DECODE, ISNULL, ISNUMERIC,NVL**
- **Erro – SQLERRCODE, SQLERRTEXT, SQLROWCOUNT**

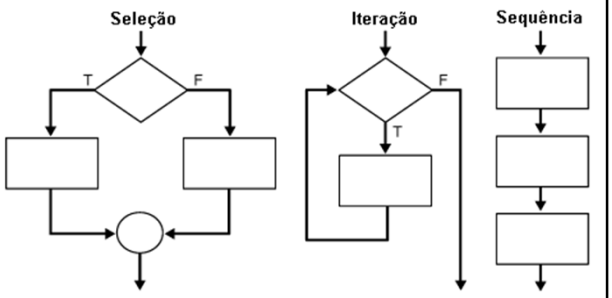
13

6. Estruturas de Controles

- Como a maioria das linguagens procedurais, o PL/SQL possui comandos para controlar o fluxo de execução do programa. São eles que fazem a diferença realizando desvios, analisando condições e permitindo a tomada de decisões dentro do programa

14

6. Estruturas de Controles



15

6. Estruturas de Controles
6.1. Comando IF

- O comando **IF...THEN** tem como finalidade avaliar uma condição e executa uma ou mais linhas de comandos somente se a condição analisada for verdadeira.

16

6. Estruturas de Controles
6.1. Comando IF(Sintaxe)

```
1. IF <condição> THEN
       <comandos>;
   END IF;

2. IF <condição> THEN
       <comandos>;
   ELSE
       <comandos>;
   END IF;

3. IF <condição> THEN
       <comandos>;
   ELSIF <condição> THEN
       <comandos>;
   END IF;
```

```
4. IF <condição> THEN
       <comandos>;
   ELSIF <condição> THEN
       <comandos>;
   ELSE
       <comandos>;
   END IF;

5. IF <condição> THEN
       IF <condição> THEN
           <comandos>;
       END IF;
   END IF;
```

17

6. Estruturas de Controles
6.1. Comando IF(Exemplo)

```
DECLARE
  QUANT NUMBER(3);
BEGIN
  SELECT ES.NR_QTD INTO QUANT
  FROM ESTOQUE ES
  WHERE CD_PROD = 30;
  IF QUANT > 0 AND QUANT < 3000 THEN
     UPDATE ESTOQUE SET NR_QTD = QUANT + 1
     WHERE CD_PROD = 30;
  ELSIF QUANT >= 3000 THEN
     INSERT INTO ALERTA(PROD,ERRO) VALUES(30, 'MÁXIMO');
  ELSE
     INSERT INTO ALERTA(PROD,ERRO) VALUES(30, 'MÍNIMO');
  END IF;
END;
```

18

6. Estruturas de Controles
6.1. Comando CASE

- O comando **CASE** tem como finalidade avaliar **condições múltiplas** e executa uma ou mais linhas de comandos somente se a condição analisada for verdadeira.

19

6. Estruturas de Controles
6.1. Comando CASE(Sintaxe)

```
CASE <expressao>
   WHEN condicao1 THEN comandos1;
   WHEN condicao2 THEN comandos2;
   WHEN condicao3 THEN comandos3;
   ...
   [ELSE comandoselse;]
END CASE;
```

20

6. Estruturas de Controles
6.1. Comando CASE(Sintaxe)

```
DECLARE
   hora number;
BEGIN
   SELECT to_char(sysdate,'HH24') INTO hora FROM dual;
   CASE hora
      WHEN 7 THEN dbms_output.put_line('Cafe da Manha');
      WHEN 12 THEN dbms_output.put_line('Almoco');
      WHEN 18 THEN dbms_output.put_line('Janta');
      ELSE dbms_output.put_line('Estudar');
   END CASE;
END;
```

21

6. Comandos de Repetição
6.2. Comando LOOP

- O comando **LOOP** executa um grupo de comandos indefinidamente ou até que uma condição force a “quebra” do LOOP e desvie a execução do programa para outro lugar. O comando LOOP é usado em conjunto com o comando EXIT, responsável pela parada de execução do LOOP.

22

6. Comandos de Repetição
6.2. Comando LOOP(Sintaxe)

```
LOOP
   <comandos>;
   EXIT;
   <comandos>;
END LOOP;
```

23

6. Comandos de Repetição
6.2. Comando LOOP(Exemplo)

```
DECLARE
  X         NUMBER := 0;
  CONTADOR  NUMBER := 0;
BEGIN
   LOOP
     X := X + 1000;
     CONTADOR := CONTADOR + 1;
     IF CONTADOR > 4 THEN
        EXIT;
     END IF;
     DBMS_OUTPUT.PUT_LINE (X ||' '|| CONTADOR || ' LOOP');
   END LOOP;
END;
```

24

6. Comandos de Repetição
6.2. Comando FOR...LOOP

- O comando **FOR...LOOP** é uma variação do comando LOOP. Aqui os comandos são executados automaticamente até que uma condição avaliada retorne falsa.

25

6. Comandos de Repetição
6.2. Comando FOR...LOOP(Sintaxe)

```
FOR <VARIAVEL> IN <INICIO>..<FINAL> LOOP
    <comandos>;
    <comandos>;
END LOOP;

FOR <VARIAVEL> IN [REVERSE] <INICIO>..<FINAL> LOOP
    <comandos>;
    <comandos>;
END LOOP;
```

26

6. Comandos de Repetição
6.2. Comando FOR...LOOP(Exemplo)

```
DECLARE
  X        NUMBER := 0;
  CONTADOR NUMBER := 0;
BEGIN
  --CONTADOR Inicia com 1 e termina em 4
  FOR CONTADOR IN 1..4 LOOP
      X := X + 1000;
      DBMS_OUTPUT.PUT_LINE (X ||' '|| CONTADOR ||
         ' LOOP');
  END LOOP;
END;
```

27

6. Comandos de Repetição
6.2. Comando FOR...LOOP(Ex Reverso)

```
DECLARE
  X        NUMBER := 0;
  CONTADOR NUMBER := 0;
BEGIN
  --CONTADOR Inicia com 4 e termina em 1
  FOR CONTADOR IN REVERSE 1..4 LOOP
      X := X + 1000;
      DBMS_OUTPUT.PUT_LINE (X ||' '|| CONTADOR ||
         ' LOOP');
  END LOOP;
END;
```

28

6. Comandos de Repetição
6.3. Comando WHILE LOOP

- Essa estrutura analisa uma condição e somente se ela for verdadeira executa os comandos contidos dentro da estrutura.

29

6. Comandos de Repetição
6.3. Comando WHILE LOOP(Sintaxe)

```
WHILE <condição> LOOP
  <comandos>;
  <comandos>;
END LOOP;
```

30

6. Comandos de Repetição
6.3. Comando WHILE LOOP(Exemplo 1)

```
DECLARE
  X        NUMBER := 0;
  CONTADOR NUMBER := 0;
BEGIN
   WHILE CONTADOR < 4 LOOP
     X := X + 1000;
     CONTADOR := CONTADOR + 1;
     DBMS_OUTPUT.PUT_LINE (X ||' '|| CONTADOR ||
         ' LOOP');
   END LOOP;
END;
```

31

6. Comandos de Repetição
6.3. Comando WHILE LOOP(Exemplo 2)

```
DECLARE
   X NUMBER(3);
   Y VARCHAR2(30);
   K DATE;
   J NUMBER(3);
BEGIN
   X:= 0;
   WHILE X<= 100 LOOP
       K:= SYSDATE-X;
       Y := 30;
       INSERT INTO TESTE VALUES (X,Y,K);
       X := X + 1;
   END LOOP;
   COMMIT;
END;
```

```
Create Table Teste
    (atributo1 number(3),
     atributo2 varchar2(30),
     atributo3 date)
```

32

Bibliografia

- **Principal**
  - DATE, C. J. **Introdução a Sistemas de Banco de Dados.** 7. ed. Rio de Janeiro: Campus, 2000. 803 p.
  - ELMASRI, S. N.; NAVATHE, B.S.. **Sistemas de Banco de Dados:** Fundamentos e Aplicações. 3. ed. Rio de Janeiro: Livros Técnicos e Científicos, 2002. 837 p.
  - SILBERSCHATZ, A. ; KORTH, H.F. ; SUDARSHAN, S. **Sistema de Banco de Dados. 5**. ed. Rio de Janeiro: Elsevier, 2006.
- **Complementar**
  - FREEMAN, R. **Oracle, referência para o DBA:** técnicas essenciais para o dia-a-dia do DBA. Rio de Janeiro: Elsevier, 2005.
  - RAMALHO, J. A. **Oracle:** Oracle 10g, ed. São Paulo: Pioneira Thomsom Learning, 2005.

Fim

34