PL/SQL

PL/SQL (acrónimo para a expressão inglesa Procedural Language/Structured Query Language) é uma extensão da linguagem padrão SQL para o SGBD Oracle da Oracle Corporation. É uma linguagem procedural da Oracle que estende a linguagem SQL.

Permite que a manipulação de dados seja incluída em unidades de programas. Blocos de PL/SQL são passados e processados por uma PL/SQL Engine que pode estar dentro de uma ferramenta Oracle ou do Server. A PL/SQL Engine filtra os comandos SQL e manda individualmente o comando SQL para o SQL Statement Executor no Oracle Server, que processa o PL/SQL com os dados retornados do Server.

É a linguagem básica para criar programas complexos e poderosos, não só no banco de dados, mas também em diversas ferramentas Oracle.

Antes de 1991 a única forma de usar construções procedurais com o SQL era usar PRO*C. Foi onde as instruções SQL do Oracle foram embutidas em código C. O código C era pré-compilado para converter as instruções SQL em chamadas de bibliotecas.

Em 1991 o PL/SQL 1.0 foi lançado com o Oracle Versão 6.0. Ele era muito limitado nas suas capacidades.

Já a versão 2.0 era uma atualização maior, que suportava stored packages, procedures, funções, tabelas PL/SQL, registros definidos pelo programador e package extensions. Esta versão foi lançada com o Oracle Versão 7.0.

O PL/SQL Versão 2.1 foi liberado com a Versão 7.1 do Oracle. Isto permitiu o uso de stored functions dentro de instruções SQL e a criação de SQL dinâmico pelo uso do pacote DBMS_SQL. Foi também possível executar instruções de Linguagens de Definição de Dados de programas PL/SQL.

A Versão 2.2 PL/SQL foi lançada com a Versão 7.2 do Oracle. Ele implementava uma proteção do código para programas PL/SQL e também o agendamento de trabalhos do banco de dados com o pacote DBMS_JOB.

A Versão 2.3 do PL/SQL foi lançado com a Versão 7.3 do Oracle. Esta versão aumentou as capacidades das tabelas PL/SQL e adicionou funcionalidades de E/S de arquivos.

A Versão 2.4 do PL/SQL foi liberada com a Versão 8.0 do Oracle. Esta versão suporta os melhoramentos do Oracle 8, incluindo Large Objects, projeto orientado a objetos, tabelas aninhadas e Oracle advanced queuing.

Estrutura Básica da PL/SQL

Com PL/SQL, você pode usar instruções SQL para manipular dados do Oracle e controle de fluxo para processar os dados. Além disso, você pode declarar constantes e variáveis, definir procedimentos e funções.

Assim, PL/SQL combina o poder de manipulação dos dados de SQL com o poder de processamento de dados das linguagens procedurais.

As unidades básicas (procedimentos, funções e blocos anônimos) que compõem um programa PL/SQL são blocos lógicos, que podem conter qualquer número de sub-blocos aninhados.

Tipicamente, cada bloco lógico corresponde a um problema ou subproblema a ser resolvido. Assim, PL/SQL suporta a "abordagem dividir para conquistar" para a resolução de problemas chamado refinamento passo a passo.

Um bloco (ou sub-bloco) permite agrupar declarações logicamente relacionadas. Dessa forma, você pode colocar declarações perto de onde elas são usadas. As declarações são locais para o bloco e deixam de existir quando o bloco é concluído.

A unidade básica em PL/SQL é um bloco. Todos os programas em PL/SQL são compostos por blocos, que podem estar localizados uns dentro dos outros. Geralmente, cada bloco efetua uma ação lógica no programa. Um bloco tem basicamente a seguinte estrutura:

Declare

Seção para declaração de variáveis,tipos e subprogramas locais.

Begin

Seção Executável, nesta seção ficam as instruções procedimentais e SQL. Esta é a única seção do bloco que é indispensável e obrigatória.

Exception

Seção/Setor onde ficam as instruções de exceção. Ex: valor vazio, valor duplicado

End

A ordem das partes segue uma lógica. Primeiro vem a parte declarativa, em quais objetos podem ser declarados. Uma vez declarados, os objetos podem ser manipulados na parte executável. Exceções que surgem durante a execução podem ser tratadas na parte de tratamento de exceção.

Você pode aninhar sub-blocos na parte executável e na de tratamento de exceções de um bloco PL/SQL ou subprograma, mas não na parte declarativa. Além disso, você pode definir subprogramas locais na parte declarativa de qualquer bloco. No entanto, você pode chamar subprogramas locais só a partir do bloco em que eles são definidos.

Tipos de Dados da PL/SQL

Toda constante, variável, parâmetro e função da PL/SQL retorna um valor como um tipo de dado, tipo esse, que determina seu formato de armazenamento e seus valores e operações válidos.

A PL/SQL tem muitos tipos e subtipos de dados no pacote STANDARD e permite que você defina seus próprios subtipos. Os principais tipos de dados da PL/SQL são:

Os tipos de dados SQL;

BOOLEAN;

PLS_INTEGER e BINARY_INTEGER;

REF CURSOR e SYS_REFCURSOR;

Subtipos definidos pelo usuário.

BOOLEAN

O tipo BOOLEAN armazena valores lógicos, que são os valores booleanos TRUE e FALSE e o valor NULL que representa um valor desconhecido.

A sintaxe para declarar uma variável BOOLEAN é:

nome_variavel BOOLEAN

O único valor que você pode atribuir a uma variável BOOLEAN é uma expressão booleana. Por exemplo:

```
DECLARE
  nome_variavel BOOLEAN;
BEGIN
  -- Essas estruturas de repetição do tipo WHILE são equivalentes

  nome_variavel := FALSE;
  WHILE nome_variavel = FALSE
    LOOP
      nome_variavel := TRUE;
    END LOOP;

  nome_variavel := FALSE;
  WHILE NOT (nome_variavel = TRUE)
    LOOP
      nome_variavel := TRUE;
    END LOOP;

  nome_variavel := FALSE;
  WHILE NOT nome_variavel
    LOOP
      nome_variavel := TRUE;
    END LOOP;
END;
```

Como a SQL não tem um tipo de dado equivalente a BOOLEAN, você não pode:

Atribuir um valor booleano a uma coluna da tabela de banco de dados;

Selecionar ou buscar o valor de uma coluna da tabela do banco de dados em uma variável BOOLEAN;

Usar um valor do tipo BOOLEAN em uma instrução SQL, função SQL ou função PL/SQL chamada de uma instrução SQL.

PLS_INTEGER

Os tipos PLS_INTEGER e BINARY_INTEGER são idênticos. Para simplificar, vamos usar PLS_INTEGER para significar PLS_INTEGER e BINARY_INTEGER.

PLS_INTEGER armazena valores inteiros que vão de -2.147.483.648 a 2.147.483.647, representados em 32 bits.

Vantagens do PLS_INTEGER sobre o tipo de dados NUMBER e os subtipos NUMBER:

Os valores PLS_INTEGER ocupam menos espaço de armazenamento;

As operações PLS_INTEGER usam ULA, portanto, são mais rápidas do que as operações NUMBER, que usam library arithmetic (biblioteca aritmética de software).

Para maior eficiência, use os valores PLS_INTEGER para todos os cálculos de números inteiros que estejam dentro de seu intervalo.

Subtipos de PLS_INTEGER:

NATURAL: valores positivos de PLS_INTEGER incluído o 0;

NATURALN: valores do subtipo NATURAL e não nulos;

POSITIVE: valores positivos de PLS_INTEGER excluído o 0;

POSITIVEN: valores do subtipo POSITIVE e não nulos;

SIGNTYPE: valores -1, 0 ou 1 (útil para programar lógica de estado triplo);

SIMPLE_INTEGER: valores de PLS_INTEGER não nulos.

REF CURSOR

Um cursor é um ponteiro para uma área privada da SQL que armazena informações sobre o processamento de uma instrução SELECT ou DML específica. Um cursor de sessão permanece na memória da sessão até que a sessão termine. Um cursor de sessão que é construído e gerenciado pela própria PL/SQL é um cursor implícito. Um cursor de sessão que você constrói e gerencia é um cursor explícito.

Uma variável de cursor é como um cursor explícito, exceto que:

Não se limita a uma consulta. Você pode abrir uma variável de cursor para uma consulta, processar o conjunto de resultados e usar a variável de cursor para outra consulta;

Você pode atribuir um valor a ele;

Você pode usá-lo em uma expressão;

Pode ser um parâmetro de subprograma. Você pode usar variáveis de cursor para passar conjuntos de resultados de consulta entre subprogramas;

Pode ser uma variável de host. Você pode usar variáveis de cursor para passar resultados de consultas entre subprogramas armazenados e seus clientes;

Não pode aceitar parâmetros. Você não pode passar parâmetros para uma variável de cursor, mas pode passar consultas inteiras para ela.

Uma variável de cursor tem essa flexibilidade porque é um ponteiro, ou seja, seu valor é o endereço de um item, não o próprio item.

Antes de fazer referência a uma variável de cursor, você deve fazer com que ela aponte para uma área de trabalho SQL, abrindo-a ou atribuindo a ela o valor de uma variável de cursor ou variável de cursor de host já aberta.

Para criar uma variável de cursor, declare uma variável do tipo SYS_REFCURSOR ou defina um tipo REF CURSOR e então declare a variável. Informalmente, uma variável de cursor às vezes é chamada de REF CURSOR. Para simplificar, vamos usar REF CURSOR para significar SYS_REFCURSOR e REF CURSOR.

A sintaxe básica de um tipo REF CURSOR é:

```
DECLARE
  TYPE nome_variavel1 IS REF CURSOR RETURN nome_tabela1%ROWTYPE;  -- tipo forte
  TYPE nome_variavel2 IS REF CURSOR;                    -- tipo fraca

  cursor1  nome_variavel1;       -- variável de cursor forte
  cursor2  nome_variavel2;   -- variável de cursor fraca
  meu_cursor1 SYS_REFCURSOR;  -- variável de cursor fraca

  TYPE meu_cursor2 IS REF CURSOR RETURN nome_tabela2%ROWTYPE;  -- tipo forte
  cursor3 meu_cursor2;  -- variável de cursor forte
BEGIN
  NULL;
END;
```

As variáveis nome_tabela1 e nome_tabela2 podem ser dos tipos de dados:

RECORD;

Uma tabela ou uma consulta SQL;

Se você especificar o nome_retorno então a sua variável de cursor é forte, senão ela é fraca. Com uma variável de cursor forte você só pode relacionar consultas que retornam o tipo especificado na própria variável de cursor, entretanto, com uma fraca você pode relacionar qualquer consulta.

Abrindo uma variável de cursor:

```
DECLARE
  TYPE nome_variavel1 IS REF CURSOR RETURN nome_tabela1%ROWTYPE;  -- tipo forte
  TYPE nome_variavel2 IS REF CURSOR;                    -- tipo fraca

  cursor1  nome_variavel1;       -- variável de cursor forte
```

```
  cursor2  nome_variavel2;   -- variável de cursor fraca
  meu_cursor1 SYS_REFCURSOR;  -- variável de cursor fraca

  TYPE meu_cursor2 IS REF CURSOR RETURN nome_tabela2%ROWTYPE;  -- tipo forte
  cursor3 meu_cursor2;  -- variável de cursor forte

BEGIN
  OPEN nome_variavel1 FOR SELECT * FROM nome_tabela1;
END;
```

Fechando uma variável de cursor:

```
DECLARE
  TYPE nome_variavel1 IS REF CURSOR RETURN nome_tabela1%ROWTYPE;  -- tipo forte
  TYPE nome_variavel2 IS REF CURSOR;                    -- tipo fraca

  cursor1  nome_variavel1;       -- variável de cursor forte
  cursor2  nome_variavel2;   -- variável de cursor fraca
  meu_cursor1 SYS_REFCURSOR;  -- variável de cursor fraca

  TYPE meu_cursor2 IS REF CURSOR RETURN nome_tabela2%ROWTYPE;  -- tipo forte
  cursor3 meu_cursor2;  -- variável de cursor forte

BEGIN
  OPEN nome_variavel1 FOR SELECT * FROM nome_tabela1.
  CLOSE nome_variavel1;
END;
```

Subtipos Definidos Pelo Usuário

A PL/SQL permite que você defina seus próprios subtipos. O tipo base pode ser qualquer tipo de dados PL/SQL como CHAR, DATE ou RECORD incluindo um subtipo definido, anteriormente, pelo usuário.[12]

Os subtipos podem:

Fornecer compatibilidade com tipos de dados ANSI/ISO;

Mostrar o uso pretendido para o subtipo em questão;

Detectar valores fora do intervalo.

```
DECLARE
  SUBTYPE Cep IS POSITIVEN(8);

BEGIN
  Cep := 12345678;
END;
```