Gestión de bases de datos

Construcción de Guiones

CEU

Introducción PL / SQL

Una vez aprendido el potencial y alcance de SQL en los temas 4 y 5, se observa que hay algunas acciones que no se han podido realizar directamente con SQL.

Oracle desarrolló un lenguaje procedimental para ampliar las funcionalidades de SQL, denominado **P**rocedural **L**anguage / **S**tructured **Q**uery **L**anguage, es decir, **PL / SQL**.

LENGUAJE DE PROGRAMACIÓN INSCRUSTADO EN ORACLE

CEU

Introducción PL / SQL

PL / SQL **SOPORTA** EL LENGUAJE DE MANIPULACIÓN DE DATOS (**DML**): SELECT, INSERT, UPDATE Y DELETE.

PL / SQL **NO SOPORTA** LOS LENGUAJES DE DEFINICIÓN DE DATOS (DDL) NI DE CONTROL DE DATOS (DCL): CREATE, ALTER, DROP… GRANT, REVOKE.

CEU

Introducción PL / SQL

PL / SQL incluye las características propias de un lenguaje procedimental:

- Variables.
- Estructuras de control de flujo y toma de decisiones.
- Control de excepciones.
- Paquetes, procedimientos y funciones.

Es habitual reutilizar los códigos desarrollados puesto que este se almacena como objetos de la base de datos (mediante paquetes, procedimientos, funciones, etc.).

CEU

Introducción PL / SQL

El código PL / SQL se ejecuta en el servidor.

Los **triggers o disparadores** es un tipo de código especial que permite realizar una acción concreta sobre una base de datos cuando se ha producido un evento, como la modificación de un registro en una tabla, la inserción de un dato o bien la eliminación de una fila de información.

CEU

Introducción PL / SQL

PL / SQL es una solución desarrollada por Oracle y por tanto se usa en el **SGBD Oracle Database**.

Hay algunos SGBD que han incorporado PL / SQL, como por ejemplo DB2.

Otros SGBG como PostgreSQL han implementado una solución similar a PL / SQL. En su caso es PL / pgSQL.

CEU

Bloques de Código Anónimos

- Constituyen el código más elemental en PL / SQL.
- Son instrucciones que se pueden lanzar directamente desde la consola de SQL.
- Se ejecutan tras introducir el carácter /
- Este código no se guarda en la BD por lo que para reutilizarlo hay que volver a escribirlo.

CEU

Bloques de Código Anónimos

La estructura que tiene es:

```
[DECLARE]
-- variables, cursores, excepciones, etc.
BEGIN
        --sentencias SQL
        --sentencias de control PL / SQL
[EXCEPTION]
        --acciones a realizar cuando hay errores
END;
/
```

Lo único obligatorio es BEGIN y END.

CEU

Bloques de Código Anónimos

Explicación en detalle de la estructura:

- En la primera parte, DECLARE, se van a declarar todas las variables, constantes, cursores, etc. que se necesitarán en el script.
- Entre BEGIN y END se indicarán todas las sentencias de control:
  - Secuencias: órdenes, asignaciones, llamadas a funciones, etc. Cada secuencia termina siempre en punto y coma (;).
  - Alternativas: se evalúa una expresión y dependiendo de su valor se redirige el flujo del programa a una serie de instrucciones o hacia otras.

CEU

Bloques de Código Anónimos

Explicación en detalle de la estructura:

- Entre BEGIN y END…
  - Bucles (iteraciones): repetición de instrucciones mientras se cumpla cierta condición o se finalice una.

CEU

Tipos de datos

Las constantes (valores inalterables) y variables (datos que pueden cambiar) tienen que tener un tipo de dato soportado por PL / SQL.

El tipo de dato condicionará el formato de almacenamiento del dato, las restricciones y los rangos de valores válidos.

PL / SQL proporciona una variedad predefinida de tipos de datos, que conforman los vistos hasta ahora en DDL más algunos otros propios de PL / SQL.

Tipos de datos

Los más habituales son:

- NUMBER. Funciona igual que en DDL.
- CHAR. Igual que en DDL.
- VARCHAR2. Igual que en DDL.
- BOOLEAN. Almacena TRUE o FALSE.
- DATE.
- Atributos de tipo. Es un modificador que se emplea para obtener información de un objeto de la BD.

Tipos de datos

Atributos de tipo

Es un modificador que se emplea para obtener información de un objeto de la BD.

- Atributo %TYPE: devuelve el tipo de una variable, constante, campo, etc.
- Atributo %ROWTYPE: devuelve los tipos de todas las columnas de una tabla, etc.

PL / SQL permite la creación de tipos personalizados (REGISTROS) y colecciones (TABLAS).

CEU

Declaración de variables

Las variables deben ser declaradas para poder usarlas.

Hay que recordar que las variables se utilizan entre BEGIN y END, y se declaran al principio en DECLARE.

La declaración implica la reserva del espacio de memoria necesario para almacenar dicha variable.

En la declaración se puede inicializar la variable, si se quiere.

CEU

Declaración de variables

Sintaxis:
nombre [CONSTANT] tipoDato [NOT NULL] [:= | DEFAULT | Expresión];

- Recordar que […] indica OPCIONAL.
- := inicializar una variable o constante con un valor.
- Las cadenas de caracteres van entre comillas simples '…'.
- Los valores numéricos van siempre sin comillas.
- Las fechas se pueden inicializar con '01-JAN-2021' o con to_date('01/01/2021','dd/mm/yyyy').

CEU

Ejercicio 1

- Declara una variable tipo fecha sin inicializarla.
- Declara una variable tipo fecha inicializada a día 6 de abril de 2021.
- Declara una variable numérica (NUMBER) con 3 dígitos que no sea null sin inicializar.
- Declara una variable numérica (NUMBER) con 4 dígitos de precisión de los cuales 2 son decimales, e inicialízala a 10,99.
- Declara una variable carácter (VARCHAR2) de tamaño 10 con la frase CLASE DE DAMDAW.
- Declara una constante numérica y valor 11.

CEU

Declaración de variables

Es muy útil emplear los atributos especiales (%TYPE y %ROWTYPE) para declarar variables del tipo de dato de un campo de una tabla en la BD.

¿Por qué?

CEU

Declaración de variables

Es muy útil emplear los atributos especiales (%TYPE y %ROWTYPE) para declarar variables del tipo de dato de un campo de una tabla en la BD.

¿Por qué?

Si se cambia el tipo de dato del campo de la BD, todas las variables declaradas usando %TYPE de ese campo se cambiarán automáticamente sin tocar nada.

Declaración de variables

Una variable del tipo registro permite referirnos a cada campo de una tabla simplemente usando un punto y a continuación indicando el nombre de del campo (tras el nombre de la variable).

Ej.: registro Usuarios%ROWTYPE;
registro.Id

Tabla USUARIOS tiene como primer campo ID.

CEU

Ejercicio 2

- Declara una variable que tenga el tipo de dato igual que el de la columna ENAME de la tabla EMP.
- Declara una variable que haga referencia a toda una fila de la tabla EMP.

CEU

Declaración de variables

La asignación a una variable se puede hacer al declararla (DECLARE), también denominado inicialización, o posteriormente (BEGIN … END).

Si se realiza en cualquier momento posterior a su declaración, solo hay que usar := valor tras el nombre de la variable.

Ej.: numero := 10;

Las constantes deben ser inicializadas al declararse y **NO PUEDEN** cambiar de valor durante la ejecución.

Declaración de variables

También se pueden asignar valores a partir de los datos de una consulta SQL (empleando INTO).

Sintaxis:
SELECT campo INTO variable FROM nombreTabla;

Si se declara una **variable registro**, entonces se le podrá asignar el valor de una fila completa.

Sintaxis:
SELECT * INTO variableRegistro FROM nombreTabla WHERE condicion; --*es muy importante que devuelva un único valor para que no dé error.*

CEU

Ejercicio 3

- Utilizando las dos variables declaradas en el ejercicio 2, asígnales los valores siguientes:
  - Primer caso: el valor del campo ENAME cuando EMPNO vale 7839.
  - Segundo caso: la fila completa cuando EMPNO vale 7698.

CEU

Operadores y expresiones

Los operadores se aplican sobre variables y constantes dando lugar a expresiones, las cuales constituirán la lógica de nuestra aplicación.

Al igual que se vio en SQL, los distintos operadores tienen un orden de ejecución que es importante conocer para que el resultado de nuestra programación sea el esperado.

Si se quiere “romper” este orden, tan solo hay que hacer uso de paréntesis para que esos operadores se interpreten antes que el resto.

CEU

Operadores y expresiones

OPERADOR	ACCIÓN
**	Potencia
+ - (unarios)	Signos (positivo o negativo)
* /	Multiplicación, división
+ - ||	Suma, resta, concatenación
= **<** > **<=** >= **<>** != **IS NULL** LIKE **BETWEEN** IN	Igual, menor, mayor, menor o igual, mayor o igual, distinto, distinto, es nulo, como, entre, en
NOT AND OR	Negación, AND lógico, OR lógico

CEU

Operadores y expresiones

Resultado de la operación AND

A	B	A AND B
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE
FALSE	NULL	FALSE
NULL	FALSE	FALSE
TRUE	NULL	NULL
NULL	TRUE	NULL

CEU

Operadores y expresiones

Resultado de la operación OR

A	B	A OR B
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE
FALSE	NULL	NULL
NULL	FALSE	NULL
TRUE	NULL	TRUE
NULL	TRUE	TRUE

CEU

Operadores y expresiones

Resultado de la operación NOT

A	NOT A
FALSE	TRUE
TRUE	FALSE
NULL	NULL

Ejercicio 4

- Analiza las expresiones siguientes e indica en cada caso si el resultado es TRUE, FALSE o NULL.
  1. (16 – 8) / 2 = 4
  2. (7 + 3) != 10
  3. 0 IS NOT NULL
  4. (3 = (9/3)) AND NULL
  5. NULL OR (2 * 5 = 10)
  6. 0 <> NULL
  7. 4 BETWEEN 3 AND 9
  8. NOT(2 ** 3 = 8)
  9. ‘a12’ = ‘a’ || ’12’
  10. 0 IS NULL
  11. ‘ANA’ LIKE ‘%N’
  12. ‘B’ IN (‘A’,’D’)

CEU

Entrada y salida

PL / SQL no tiene instrucciones especializadas en la entrada y salida porque no está pensado para ello, pero existen variables y métodos de entrada y salida que serán muy útiles para depurar el script programado.

Salida

La iniciar una sesión hay que activar la variable SERVEROUTPUT.

SET SERVEROUTPUT ON

Para escribir por pantalla se utiliza el método dbms_output.put_line().

Ej.: dbms_output.put_line('test');

Salida

¿Qué información se puede mostrar?

- Variables.
- Constantes.
- Textos.
- Resultado de funciones propias y de Oracle (Ej.: sysdate).

Se puede emplear el operador || para concatenar diferentes cadenas y valores de variables.

Ej.: dbms_output.put_line(‘Error: ‘ || nombreVariable);

Entrada

Se hace una asignación de los valores introducidos por teclado a una variable. Si se pide una cadena de caracteres, el usuario debe introducirla con ‘’.

Hay que indicar mediante una cadena el mensaje a mostrar. Para ello se asignará como valor a la variable que captura del teclado mediante el símbolo & y a continuación el texto a mostrar.

El texto a mostrar debe ser una cadena sin espacios en blanco, corta y con las palabras separadas por _.

Ej.: variable := &Texto_que_se_muestra;

Ejercicio 5

- Crea un bloque de código anónimo que pida la base y altura de un triángulo, y devuelva el área de este.

NOTA: El área de un triángulo es base por altura entre dos.

CEU

Ejercicio 6

- Crea un bloque de código anónimo que requiera por pantalla un nombre, luego un apellido y muestre como resultado “Hola nombre apellido”.

Ejercicio 7

Crea un programa que realiza la suma, resta, multiplicación y división de dos números enteros, num1 y num2 (8 y 4).

CEU

Estructuras de control

Una función computable puede ser implementada en un lenguaje de programación a través de una de las siguientes tres estructuras lógicas:

- **Secuencia**: ejecución de instrucción tras instrucción.
- **Selección**: ejecución de una o varias instrucciones según una condición.
- **Iteración o bucle**: ejecución de una o varias instrucciones mientras una condición se mantenga verdadera.

Estructuras de control: IF

Son estructuras condicionales o de selección.

Se evalúa una expresión/condición y según sea su valor, se ejecutarán unas instrucciones u otras.

Sintaxis:

```
IF condición THEN
        …
ELSE
        …
END IF;
```

CEU

Estructuras de control: IF

Se pueden anidar diferentes if/else.

Sintaxis:

```
IF condición THEN
        …
        ELSIF condición THEN
                …
                ELSIF condición THEN
                        …
                ELSE
                        …
        ELSE
                …
ELSE
        …
END IF;
```

Ejercicio 8

Crea un programa que realiza la suma de dos números enteros, num1 y num2, si num1 es mayor que num2. En caso contrario que no haga nada.

Asigna por ejemplo los valores 7 y 3 a los números.

Ejercicio 9

Crea un programa que realiza la resta de dos números enteros, num1 y num2, si num1 es mayor que num2. En caso contrario, que muestre por pantalla ‘num1 es menor que num2’, sustituyendo num1 y num2 por sus valores.

Asigna por ejemplo los valores 7 y 3 a los números.

Ejercicio 10

Crea un programa que tome pida una variable al usuario para que la introduzca por teclado (tiene que ser un número), y según sea su valor entre 0 y 10, devuelva por pantalla el valor de la nota: SUSPENSO, APROBADO, BIEN, NOTABLE, SOBRESALIENTE. En caso contrario, que devuelva ‘El valor introducido es incorrecto’.

CEU

Estructura de control CASE

Evalúa cada condición hasta encontrar alguna que se cumpla, y en caso de que no encuentre ninguna, se ejecuta la/s instrucción/es de ELSE, si la hubiera.

Sintaxis:

```
CASE [expresión]
WHEN ‘{cond1|val1} THEN
        …
WHEN ‘{cond2|val2} THEN
        …
WHEN ‘{condn|valn} THEN
        …
ELSE
        …
END CASE;
```

Ejercicio 11

Realiza el mismo ejercicio 11 pero utilizando CASE en lugar de IF | ELSIF | END IF.

Ejercicio 12

Realiza un programa que lea por teclado dos números enteros. El primero será los goles del equipo de casa, y el segundo los goles del equipo de fuera.
Se quiere devolver por pantalla quién ha ganado: “El equipo de casa/visitante ha ganado”. En caso de empate se indicará “El resultado del partido ha sido de empate”.

Estructura de control: iteraciones

Existen tres tipos de bucles:

- LOOP. Se producen acciones repetitivas sin condiciones globales.
- FOR. En este caso el bucle se repite según cumpla una condición un contador.
- WHILE. Se basa en una condición que mientras se cumpla se seguirá ejecutando el bucle.

CEU

Estructura de control LOOP

Sintaxis:

```
LOOP
        instrucciones
END LOOP;
```

Al ser un bucle sin condiciones, se ejecutaría para siempre (bucle infinito). Para evitar esto, hay que provocar una salida del mismo, bien mediante EXIT WHEN condición, bien a través de un IF condición THEN EXIT.

Ejercicio 13

Realiza un programa que ejecute un bucle LOOP y se salga con un EXIT WHEN. Para ello crea una variable entero inicializada a 0 y que se vaya incrementando en el bucle, además de mostrar por pantalla su valor; la condición de salida será cuando dicha variable valga más de 20.

Ejercicio 14

Realiza un programa que haga lo indicado en el ejercicio 13 pero que se salga mediante un IF condición THEN EXIT.

Estructura de control WHILE

Sintaxis:

```
WHILE condición LOOP
        instrucciones
END LOOP;
```

Es un bucle que se ejecutará mientras se cumpla cierta condición. Para evitar un bucle infinito, habrá que incluir en el código alguna variación de las variables que se empleen en la condición para que, en algún momento, no se cumpla.

Ejercicio 15

Realiza un programa que haga lo indicado en el ejercicio 13 pero emplea un bucle WHILE.

CEU

Estructura de control FOR

Sintaxis:

```
FOR índice IN [REVERSE] val_ini .. val_fin LOOP
        instrucciones
END LOOP;
```

Es un bucle que se ejecutará mientras se cumpla cierta condición para una variable contador. Es idóneo si se sabe a priori cuántas veces se quiere repetir las instrucciones del bucle.

CEU

Estructura de control FOR

La variable índice o contador no se declara en el programa sino en el bucle FOR.

Al estar declarado para ese bucle FOR no se puede usar fuera del bucle en cualquier otra parte del programa.

No se debe usar dentro del bucle para asignarle otro valor puesto que modificaría la condición de salida del bucle.

Estructura de control FOR

El bucle FOR se encarga de incrementar esa variable contador o índice de uno en uno cada vez que termina de ejecutar todas las instrucciones del bucle.

La etiqueta opcional REVERSE se emplea para ir decrementando la variable contador o índice, en lugar de incrementarla (los valores iniciales y finales se indican de esta forma, primero el menor, y luego el mayor).

Ejercicio 16

Realiza un programa que haga lo indicado en el ejercicio 13 pero emplea un bucle FOR.

Ejercicio 17

Realiza un programa que haga lo indicado en el ejercicio 14 pero, en lugar de ir incrementando de 0 a 20, que haga lo opuesto, que vaya mostrando 20, 19, 18… y que termine con el 0. Emplea un bucle FOR.

Ejercicio 18

Realiza un programa que muestre por pantalla los números pares hasta llegar a 40, inclusive.

Nota: la función MOD(m,n) devuelve el resto de dividir el parámetro m entre el parámetro n.

Ejercicio 19

Realiza un programa que muestre por pantalla las tablas de multiplicar del 1 al 10.

CEU

CEU

Centro de Estudios Profesionales

Fundación San Pablo Andalucía