Sesión 7: PL/SQL:
Conceptos básicos de bloque PL/SQL. Manejo de excepciones.

CONCEPTOS BÁSICOS DE PL/SQL

BLOQUES PL/SQL → Bloques de sentencias ejecutables por la BD.

Un bloque SQL tiene tres partes: una declarativa, una ejecutable y una última para el manejo de excepciones (warnings y condiciones de error). De estas tres partes, sólo la parte ejecutable es obligatoria.

```
[ DECLARE
        -- declaraciones]
BEGIN
        --  sentencias
[EXCEPTION
        --  manejo de excepciones]
END;
```

DECLARACIÓN de VARIABLES y CONSTANTES

En la parte declarativa se pueden declarar variables y constantes Por ejemplo:

```
DECLARE
total number(6,2);
nombre varchar(30):= 'JUANA';  --  valor por defecto pero se puede modificar
maximo CONSTANT number(4):=9999;    -- No se puede modificar
```

En la parte declarativa, las variables se pueden inicializar a un valor específico o no hacerlo, mientras que las constantes han de ser inicializadas en la parte declarativa.

DECLARACIÓN de VARIABLES y CONSTANTES

Si necesitamos que el tipo de datos coincida con el definido para una columna de una tabla podemos utilizar **%TYPE**

El atributo %TYPE proporciona el tipo de dato de una variable o columna de la base de datos.

```
DECLARE
    valor number(4,2);
    auxvalor valor%TYPE;
    auxcategoria habitacion.categoria%TYPE;
```

La variable auxvalor es del mismo tipo de datos que valor y
la variable auxcategoria tiene el mismo tipo de datos que la columna categoría de la tabla habitación.

Utilizar el atributo %TYPE hace que no sea necesario conocer el tipo de dato exacto de una variable o columna, y por otro lado, tiene la ventaja de que si la definición del tipo de datos de una columna cambia, el tipo de dato de la columna también cambia automáticamente.

Si necesitamos definir una estructura similar a una fila de una tabla podemos utilizar **%ROWTYPE**

El atributo %ROWTYPE obtiene un tipo de registro que representa una fila de una tabla. Los campos del registro y las correspondientes columnas de la tabla tienen el mismo nombre y el mismo tipo de datos

```
DECLARE
    pvp  pvptemporada%ROWTYPE;
```

ASIGNAR VALOR A UNA VARIABLE

Por defecto las variables se inicializan a NULL.
Se les puede asignar valor de dos formas: a través de una **expresión**, o a través de una **sentencia SELECT**. ¡**EN ESTE ÚLTIMO CASO LA SELECT SÓLO PUEDE DEVOLVER UNA FILA! En caso contrario se produciría una excepción**

```
aux_valor1:=25;
aux_valor2:=total-10;
```

```
SELECT categoria  INTO auxcategoria  FROM habitacion WHERE ….
SELECT cod, descripción  INTO auxcod, auxdesc  FROM ACTIVIDADES WHERE …
```

Para mostrar mensajes podemos utilizar:

MENSAJES en bloques PL/SQL

dbms_output.put_line(mensaje)

Deposita el mensaje en un buffer y **prosigue** con la ejecución

Para que los mensajes se muestren con DBMS_OUTPUT es necesario ejecutar en la sesión de trabajo

SET SERVEROUTPUT ON

Si lo que se quiere es mostrar un mensaje de error y **detener** la ejecución

raise_application_error (número_error, mensaje)
número_error debe ir entre –20000…-20999, el resto está reservado para errores propios de oracle

raise_application_error(-20101,'Alcanzado total de votantes')

ESTRUCTURAS DE CONTROL en bloques PL/SQL

Para trabajar dentro del bloque BEGIN-END se pueden utilizar **estructuras de control** similares a las de otros lenguajes de programación.

ESTRUCTURAS DE CONTROL CONDICIONAL

```
IF … THEN…
END IF;
```

```
IF … THEN…
ELSE …
END IF;
```

```
IF … THEN…
ELSIF … THEN …
ELSIF … THEN …
[ELSE …]
END IF;
```

ESTRUCTURAS DE CONTROL ITERATIVO

```
FOR i IN min .. max LOOP
  …
  …
END LOOP;
```

```
WHILE condición LOOP
  …
END LOOP;
```

```
LOOP
 …
…
…
EXIT WHEN …;
END LOOP;
```

```
LOOP
 …
  IF condición THEN
        …
        EXIT;
  END IF;
…
END LOOP;
```

CONTROL DE EXCEPCIONES EN BLOQUES EN PL/SQL

En la mayoría de los lenguajes de programación se emplean EXCEPCIONES para el manejo de errores en tiempo de ejecución. En Oracle también podemos definir excepciones. Estas definiciones hacen que, cuando se produce un error, salte una excepción y pase el control a la sección EXCEPTION correspondiente. Las acciones a seguir se incluyen en la sección EXCEPTION definida al final del bloque BEGIN-END.

```
BEGIN

  ...

EXCEPTION
  WHEN <nombre_excepción> THEN
    <bloque de sentencias>;
 WHEN <nombre_excepción> THEN
    <bloque de sentencias>;

  ...
  [WHEN OTHERS THEN <bloque de sentencias>;]
END;
```

Existen dos tipos de excepciones:

- Las excepciones predefinidas por ORACLE.
- Las excepciones definidas por el usuario.

EXCEPCIONES PREDEFINIDAS POR ORACLE

Son aquellas que se disparan automáticamente al producirse determinados errores. Las más frecuentes son (hay muchas más):

- **too_many_rows**: Se produce cuando SELECT … INTO devuelve más de una fila.
- **no_data_found**: se produce cuando un SELECT… INTO no devuelve ninguna fila.
- **value_error**: se produce cuando hay un error aritmético o de conversión (en una sentencia del bloque PL/SQL).
- **zero_divide**: se puede cuando hay una división entre 0.
- **Dup_val_on_index**: se crea cuando se intenta almacenar un valor que crearía duplicados en la clave primaria o en una columna con restricción UNIQUE.
- **invalid_number**: se produce cuando se intenta convertir una cadena a un valor numérico (en una sentencia SQL).

EXCEPCIONES PREDEFINIDAS POR EL USUARIO

Son las que podemos definir nosotros en nuestros módulos PL/SQL. Para trabajar con excepciones definidas por el usuario, es necesario que se realicen 3 pasos:

1. Definición de la etiqueta (o nombre) de la excepción en la sección DECLARE. La sintaxis es:

   Nombre_de_excepción EXCEPTION

2. Lanzar la excepción. Se hace mediante la orden RAISE.
3. Definir las sentencias a ejecutar, cuando se lance la excepción. Esto se hace en la sección EXCEPTION.

   WHEN nombre_de_excepción THEN …

Ejemplos de manejo de excepciones

Ejemplo 1: Excepciones controladas por Oracle

Crea un bloque PL/SQL en el que, dado el código de un recurso, que hemos almacenado en una variable previamente, muestre su descripción y fecha de alta. Si ese recurso no existe, debe mandarse un mensaje diciendo que no se encuentra en la base de datos, pero continúe la ejecución sin dar un fallo.

```
DECLARE
        auxdescripcion recurso.descripcion%type;
        auxfechaalta recurso.falta%type;
        auxcodigo recurso.codigo%type := 1;
BEGIN
 select descripcion, falta into auxdescripcion, auxfechaalta
 from recurso where codigo = auxcodigo;
 dbms_output.put_line('Descripcion:  '|| auxdescripcion || '   Fecha alta: ' || auxfechaalta);
 EXCEPTION
     WHEN no_data_found then
      dbms_output.put_line('No existe ese recurso');
END;
dbms_output.put_line('Continúo con la ejecución del bloque PL/SQL');
```

Ejemplo 2: Excepciones controladas por el usuario.

Crea un PL/SQL en el que dado el código de un recurso almacenado en una variable previamente, incluya ese recurso como gratuito siempre que existan ya más de 20 de pago. Si el recurso no está dado de alta no se producirá un error sino que se dará de alta en la base de datos en la tabla recurso y también en la tabla recurso_gratuito.

```
DECLARE
      auxdescripcion recurso.descripcion%type;
      total number(3);
      no_llega exception;
      auxcodigo recurso.codigo%type := 5;
BEGIN
--Obtenemos los recursos de pago
 select count(*) into total from recurso_pago;

 if total <=20  then raise no_llega; end if;

--Vemos si ya existe el recurso

 Select descripcion  into auxdescripcion
 from recurso where codigo = auxcodigo;
--Como existe mostramos un mensaje
      dbms_output.put_line('El recurso ' || auxcodigo || ` ya existe.');
EXCEPTION
    WHEN no_data_found then
--no existe el recurso, lo insertamos en ambas tablas
        insert into recurso (codigo,falta) values (auxcodigo,sysdate);
        insert into recurso_gratuito(Codigo) values (auxcodigo);
   WHEN no_llega then
      dbms_output.put_line('No hay suficientes recursos de pago');
END;
```