**Declaración del cursor:** CURSOR <nombrecursor> IS SELECT <sentencia select>;
**Apertura del cursor:** OPEN <nombrecursor>;
**Recogida de información:** FETCH <nombrecursor> INTO {<variable> | <listavariables>}; **la variable puede ser declarada:**
<variable> <nombrecursor>%ROWTYPE
**Cierre del cursor:** CLOSE <nombrecursor>;

**Atributos del cursor:**
**%FOUND** Devuelve **verdadero** si el último FETCH ha recuperado algún valor. Si el cursor no estaba abierto devuelve **error**, y si estaba abierto pero no se había ejecutado aún ningún FETCH, devuelve **NULL**.
**%NOTFOUND.** Hace lo contrario que el atributo anterior.
**%ROWCOUNT** Devuelve el número de filas recuperadas hasta el momento por el cursor (número de FETCH realizados satisfactoriamente).
**%ISOPEN** Devuelve verdadero si el cursor está abierto.

**Variables de acoplamiento en el manejo de cursores.** Son las que se incluyen en la cláusula WHERE y se tienen que declarar previamente.

**Cursor FOR..LOOP.** Esta estructura realiza las siguientes tareas de manera implícita:
- Declarar una variable que recogerá los datos del cursor.
- Abrir el cursor.
- Recuperar con FETCH una a una las filas extraídas, introduciendo los datos en la variable, procesándolos, y comprobando también si se han recuperado datos o no.
- Cerrar el cursor.

El formato y el uso de esta estructura es:
1. Se declara la información cursor en la sección correspondiente (como cualquier otro cursor)
2. Se procesa el cursor utilizando el siguiente formato:

```
FOR nombrevareg IN nombrecursor LOOP
    ….
END LOOP;
```

**Uso de alias en las columnas de selección del cursor.** Cuando en una consulta hay expresiones, debemos colocar alias en las columnas.
**Cursores con parámetros.** Se pueden usar parámetros en lugar de variables de acoplamiento. Todos los parámetros formales de un cursor son parámetros de entrada y su ámbito es local al cursor, por eso solamente pueden ser referenciados dentro de la consulta:

```
CURSOR nombrecursor [(parámetro1, parámetro2,…)] IS
    SELECT <sentencia select en la que intervendrán los parámetros>;
```

Los parámetros tienen la siguiente sintaxis:

```
Nombredevariable [IN] tipodedato [{:= |DEFAULT} valor]
```

Para abrir un cursor con parámetros:

```
OPEN nombrecursor [(parámetro1, parámetro2, …)];
```

En el caso de la instrucción FOR..LOOP, el paso de parámetros se hará a continuación del identificador del cursor:

```
….
        FOR  reg_emple  IN  cur1(20,'DIRECTOR')  LOOP
….
```

**Atributos en cursores implícitos.** Oracle abre implícitamente un cursor cuando procesa un comando SQL que no esté asociado a un cursor explícito. El cursor implícito **se llama SQL** y dispone también de los cuatro atributos que pueden facilitarnos información sobre la ejecución de los comandos SELECT INTO, INSERT, UPDATE y DELETE.
El comportamiento de los atributos en los cursores implícitos es distinto al de los cursores explícitos

1. Devolverán un valor relativo a la última orden, aunque el cursor esté cerrado
2. SELECT.. INTO ha de devolver una fila y sólo una, pues de lo contrario se producirá un error y se levantará automáticamente una excepción:
   - NO_DATA_FOUND
   - TOO_MANY_ROWS

   Se detendrá la ejecución normal del programa y bifurcará a la sección EXCEPTION
3. Lo anterior **no es aplicable** a las órdenes **INSERT, UPDATE, DELETE,** ya que en estos casos no se levantan las excepciones correspondientes

Cuando un SELECT INTO hace referencia a una **función de grupo** nunca se levantará la excepción NO_DATA_FOUND y **SQL%FOUND siempre será verdadero** porque siempre retornan algún valor (aunque sea NULL).

**Excepciones:** sirven para tratar errores en tiempo de ejecución, así como errores y situaciones definidas por el usuario.

```
EXCEPTION
        WHEN <nombredeExcepción1> THEN
                <instrucciones1>;
        WHEN <nombredeExcepción2> THEN
                <instrucciones2>;
        …

        [WHEN OTHERS THEN
                <instrucciones>;]
END<nombre de programa>;
```

Cuando se produce un error, PL/SQL levanta una excepción y pasa el control a la sección EXCEPTION correspondiente del bloque PL, que actuará según lo establecido y dará por finalizada la ejecución del bloque actual

Hay tres tipos de excepciones:

- Excepciones internas predefinidas.
- Excepciones definidas por el usuario.
- Otras excepciones

**Excepciones internas predefinidas:** Están predefinidas por Oracle. Se disparan automáticamente al producirse determinados errores. No hay que declararlas en la sección DECLARE.

**Excepciones definidas por el usuario:**
Se deben declarar en la sección DECLARE de la forma siguiente: **<nombreexcepción> EXCEPTION;**
Se disparan o levantan en la sección ejecutable del programa con la orden RAISE: **RAISE <nombreexcepción>;**

**Otras excepciones:** otros errores internos de Oracle no tienen asignada una excepción, sino un código de error y un mensaje de error, a los que se accede mediante las funciones SQLCODE y SQLERRM ( no son accesibles desde órdenes SQL*PLUS).

```
EXCEPTION
…..
  WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: '||SQLCODE ||SQLERRM);
….
END;
```

También podemos asociar una excepción a alguno de estos errores internos que no tienen excepciones predefinidas asociadas. Siguiendo los siguientes pasos:
Definimos una excepción en la sección de declaraciones como si fuese una excepción definida por el usuario: <nombreexcepción> EXCEPTION;
Asociamos esa excepción a un determinado código de error mediante la directiva del compilador PRAGMA EXCEPTION_INIT, según el formato siguiente:

```
PRAGMA EXCEPTION_INIT (<nombre_excepción>,<número_de_error_Oracle>);
```

Indicamos el tratamiento que recibirá la excepción en la sección EXCEPTION como si se tratase de cualquier otra excepción definida o predefinida:

```
DECLARE
….
Err_externo EXCEPTION;     ---- Se define la excepción de usuario
    ….
    PRAGMA EXCEPTION_INIT (err_externo, -1547);  /* Se asocia con un error de                                                    Oracle     */
    ….
BEGIN
    ….
    /*   no hay que levantar la excepción, ya que llegado el caso Oracle lo hará  */
    ….
EXCEPTION
    …
    WHEN err_externo  THEN                -- Se trata como cualquier otra
      <tratamiento>;
END;
```

**Programación y ámbito de las excepciones**

- Cuando se levanta una excepción, el programa bifurca a la sección EXCEPTION del bloque actual. Si no está definida en ella, la excepción se propaga al bloque que llamó al actual, pasando el control a la sección EXCEPTION de dicho bloque y así hasta encontrar tratamiento para la excepción o devolver el control al programa Host.
- Una vez tratada la excepción en un bloque, se devuelve el control al bloque que llamó al que trató la excepción, con independencia del que la disparó.
- Si, después de tratar una excepción, queremos volver a la línea siguiente a la que se produjo, podemos diseñar el programa para que funcione así. Esto se consigue encerrando el comando o comandos que pueden levantar la excepción en un subbloque junto al tratamiento para la excepción.
- La cláusula WHEN OTHERS tratará cualquier excepción que no aparezca en las cláusulas WHEN correspondientes, con independencia del tipo de excepción. De este modo se evita que la excepción se propague a los bloques de nivel superior.
- Si la excepción se levanta en la sección declarativa automáticamente se propagará al bloque que llamó al actual
- Se puede levantar una excepción en la sección EXCEPTION de forma voluntaria o por un error que se produzca al tratar una excepción. La excepción original (la que se estaba tratando cuando se produjo nueva excepción) se perderá, ya que solamente puede estar activa una excepción.

```
EXCEPTION
   WHEN  INVALID_NUMBER  THEN
         INSERT  INTO ….          -- podría levantar DUP_VAL_ON_INDEX
   WHEN  DUP_VAL_ON_INDEX  THEN              -- no atrapará la excepción
         …..
```

- En ocasiones, puede resultar conveniente, después de tratar una excepción, volver a levantar la misma excepción para que se propague al bloque de nivel superior. Esto puede hacerse indicando al final de los comandos de manejo de la excepción el comando RAISE sin parámetros:

```
WHEN  TOO_MANY_ROWS  THEN
      ….;          -- instrucciones de manejo de error
      RAISE;       -- levanta de nuevo y la propaga
```

- Las excepciones declaradas en un bloque son locales al bloque, por tanto, no son conocidas en bloques de nivel superior. No se puede declarar dos veces la misma excepción en el mismo bloque, pero sí en distintos bloques. En este caso, la excepción del subbloque prevalecerá sobre la del bloque, aunque esta última se puede levantar desde el subbloque utilizando la notación de punto ( RAISE nombrebloque.nombreexcepción).
- Las variables locales, las globales, los atributos de un cursor, etc. Se pueden referenciar desde la sección EXCEPTION según las reglas de ámbito que rigen para los objetos del bloque. Pero si la excepción se ha disparado dentro de un bucle cursor FOR … LOOP no se podrá acceder a los atributos del cursor, ya que Oracle cierra este cursor entes de disparar la excepción.
- Cuando no se encuentra tratamiento para una excepción en ninguno de los bloques o programas, Oracle da por finalizado con fallos el programa y ejecuta automáticamente un ROLLBACK.

**Utilización de RAISE_APPLICATION_ERROR**

En el paquete DBMS_STANDARD se incluye un procedimiento muy útil llamado RAISE_APPLICATION_ERROR que sirve para levantar errores y definir y enviar mensajes de error. Su formato es el siguiente:

**RAISE_APPLICATION_ERROR (número_de_error, mensaje_de_error)**

*Número_de_error* es un número comprendido entre –20000 y –20999, y *mensaje_de_error* es una cadena de hasta 512 bytes
Cuando un subprograma hace esta llamada, se levanta la excepción y se deshacen los cambios realizados por el subprograma.

**Control de transacciones**
**Transacción**: conjunto de operaciones dependientes unas de otras que se realizan en la base de datos. Es el usuario (el programador, en este caso) quien decide cuáles serán las operaciones que compondrán la transacción. Oracle garantiza la consistencia de los datos en una transacción en términos de VALE TODO o NO VALE NADA.
**Comandos de control de transacciones**
**COMMIT**. Concluye la transacción actual. Hace definitivos los cambios efectuados, liberando las filas bloqueadas.
**ROLLBACK**. Concluye la transacción actual. Deshace los cambios efectuados y libera las filas bloqueadas. Se usa cuando no se puede terminar una transacción porque se levanta una excepción.
**ROLLBACK implícitos**. Cuando un subprograma almacenado falla y no se controla la excepción que produjo el fallo, ORACLE ejecuta automáticamente un ROLLBACK.
**SAVEPOINT <punto_de_salvaguarda>**. Se usa para poner marcas al procesar transacciones para poder deshacer parte de ella.
**ROLLBACK TO <punto_de_salvaguarda>**. Deshace el trabajo realizado sobre la base de datos después del punto indicado, incluyendo posibles bloqueos. No confirma el trabajo hecho hasta <punto_de_salvaguarda>.

Oracle establece un punto de salvaguarda implícito cada vez que se ejecuta una sentencia de manipulación de datos. En el caso de que la sentencia falle, Oracle restaurará automáticamente los datos a sus valores iniciales.

Por omisión, el número de SAVEPOINT está limitado a cinco por sesión, pero se puede cambiar en el parámetro SAVEPOINT del fichero de inicialización de Oracle hasta 255
Cuando se ejecuta un ROLLBACK TO <marca> todas las marcas después del punto indicado desaparecen (la indicada no desaparece). También desaparecen todas las marcas cuando se ejecuta un COMMIT.
Los nombres de las marcas son identificadores no declarados y se pueden reutilizar