# Oracle básico (II): Creación y manejo de tablas

Con el artículo anterior iniciamos una entrega de Oracle Básico comenzando con el tema de creación y manejo de tablas. Ahora pasaremos a estudiar la consulta y selección de registros con el lenguaje estándar para bases de datos relacionales *SQL* (*Structured Query Languague* = *Lenguaje de Consulta estructurado*).

La ventaja principal del SQL, desde mi punto de vista, es su capacidad de combinar sencillez y facilidad con potencia y eficiencia, conteniendo un conjunto de herramientas que optimizan las consultas.

Vale la pena destacar que, aunque los conceptos a estudiar son específico de *ORACLE*, también son útiles para cualquier programador que esté trabajando con algún software que contenga SQL.

## Sentencia *SELECT*

Como ya sabemos, la herramienta fundamental de SQL es la sentencia *SELECT*, que permite seleccionar registros desde las tablas de la Base de Datos, devolviendo aquellos que cumplan las condiciones establecidas y pudiendo presentar el resultado en el orden deseado.

Primeramente estudiaremos la forma básica de la sentencia SELECT, que esta formado por:

```
SELECT   Lista...
FROM     Tabla, Tabla...
WHERE    Condiciones...
      ORDER BY Expresión,Expresión,...
      ;      Fin de la sentencia.
```

Donde:

La orden SELECT puede contener:

- Columnas: nombre, telefono
- Expresiones y funciones: *SYSDATE-fecha,UPPER(direccion)*
- Pseudo-Columnas del Sistema: *SYSDATE*,*USER*.
- Asterisco: Todas las columnas.

La orden *FROM* identifica la lista de tablas a consultar. Si alguna de las tablas a consultar no es propiedad del usuario, debe especificarse el nombre del propietario antes que el nombre de la tabla en la forma *nombre_propietario.nombre_tabla*.

La orden *WHERE* decide los registros a seleccionar según las condiciones establecidas, limitando el número de registros que se muestran.

La orden *ORDER BY* indica el orden en que aparece el resultado de la consulta.

e

Ilustremos lo explicado hasta el momento con el ejemplo del fuente 1, donde consultaremos las ventas realizadas en los últimos 10 días, mostrando el nombre del cliente, artículo vendido y su valor.

```
Fuente 1

SELECT nombre,articulo,valor    Lista nombre del cliente,
                                nombre del artículo y
                                el valor de la venta.
    FROM                        clientes,ventas
                                Tablas con la información de clientes y ventas.
    WHERE clientes.codigo=ventas.codigo
                                Establece la relación, según código de cliente,
                                entre las tablas clientes y ventas.
          and sysdate-ventas.fecha>=10
                                Consulta las ventas de los últimos 10 días.
    ORDER BY nombre             Ordenar el listado por nombre del cliente.
;                               Fin de la sentencia.
```

El resultado de esta sentencia SELECT sería el de la tabla 1:

| NOMBRE | ARTICULO | VALOR |
|---|---|---|
| CASA AUGE DEPORTES | PAPEL | 330.0 |
| CASA AUGE DEPORTES | DISKETTE | 33.0 |
| CLUB DE TENNIS | PAPEL | 500.5 |
| ... | | |
| CLUB DE TENNIS | PAPEL | 100.5 |
| FERIA DEL LIBRO | PAPEL | 310.0 |
| PINTURERIAS PROPIOS | PAPEL | 220.5 |
| PINTURERIAS PROPIOS | DISKETTE | 20.5 |

Tabla 1: Resultados de la sentencia SELECT del fuente 1

Obsérvese que las columnas que tienen el mismo nombre en ambas tablas se diferencian escribiendo el nombre de la tabla antes que el nombre de la columna, como en el caso de *ventas.fecha*, *ventas.codigo* y *clientes.codigo*.

## Operadores lógicos

Para construir la condición de la consulta necesitamos conocer los operadores lógicos, por eso a continuación damos una lista de los operadores más usados, agrupados en cuatro grupos:

1. Valor único:Comprueban un valor simple.
2. Lista de valores:Comprueban más de un valor.
3. Combinaciones lógicas:Combinan expresiones lógicas.
4. Negación:Invierte el resultado de la expresión con operadores de valor único o de lista de valores.

## Valor único

```
>  <   >=  <=  =
```

Operadores clásicos de comparación:
mayor, menor, mayor e igual, menor e igual, igual a.

```
!=  <> ^=
```

Operador "Distinto de" en sus tres formas.

```
IS NULL
```

Comprueba la ausencia de datos (valor nulo).
No se puede usar la comparación = NULL.

```
LIKE
```

Selecciona registros según el reconocimiento de un patrón de consulta.

## Lista de valores

```
BETWEEN valor AND valor
```

Comprueba que el valor se encuentre en el rango de valores.

```
IN (valor,...,valor)
```

Verifica si el valor pertenece a la lista de valores.

Combinaciones lógicas.

```
AND
```

Retorna Verdadero si todas las condiciones son verdaderas.

```
OR
```

Retorna Verdadero si alguna de las condiciones es verdadera.

## Negación

```
NOT
```

Invierte el resultado de una expresión lógica, por ejemplo.

```
IS NOT NULL

NOT BETWEEN valor AND valor

NOT IN  (valor,...,valor)

NOT LIKE
```

A continuación mostramos algunas consultas con el uso de diferentes operadores lógicos:

- Clientes a los que no se les ha registrado su dirección.

```
SELECT nombre,telefono
FROM    clientes
WHERE   direccion IS NULL;
```

- Clientes dados de alta en los últimos 10 días.

```
SELECT nombre,direccion,telefono
FROM   clientes
WHERE  fecha BETWEEN sysdate-10
       AND sysdate;
```

- Datos de los Clientes que pertenecen a una lista de clientes.

```
SELECT nombre,direccion,telefono
FROM   clientes
WHERE  nombre
       IN ('PINTURAS','CASA DE DEPORTES')
;
```

- Clientes dados de altas en lo que va del mes y cuyo nombre comience con la letra P u otra letra mayor o su teléfono contenga el código (0567). Ver fuente 2

```
Fuente 2

SELECT nombre,direccion,telefono,fecha
FROM   clientes
```

```
      WHERE
      fecha BETWEEN
            to_date('01/'||to_char(sysdate,'MM/YY'),'DD/MM/YY')
            AND
            sysdate
            AND (nombre  >= 'P' OR  telefono LIKE '%(0567)%');
```

El manejo de fecha es una de las capacidades de mayor variedad e interés en ORACLE por las posibilidades que presenta en el almacenamiento, cálculo y presentación de fechas. Por eso, en el último ejemplo damos un vistazo a algunas funciones útiles en el uso de fechas como son:

```
to_char(sysdate,'MM/YY')
```

Devuelve una cadena de caracteres de la forma mes/año de la fecha actual.

```
'01/'||to_char(sysdate,'MM/YY')
```

Forma la cadena de caracteres *01/mes/año* que representa el primer día del mes. El operador || se usa para unir o concatenar cadenas de caracteres.

```
. . .
to_date('01/'||to_char(sysdate,
                       'MM/YY'
                      ),
        'DD/MM/YY')
. . .
```

Convierte la cadena de caracteres *01/mes/año* al tipo fecha.

## Patrón de consulta

Una de las herramientas lógicas más poderosas de SQL es el reconocimiento de un patrón de consulta, instrumento éste que permite la búsqueda por nombre, dirección u otro dato parcialmente recordado. Los patrones de consulta juegan un papel importante en el momento de realizar consultas, ya que es común que necesitemos encontrar un texto y no recordemos exactamente cómo fue ingresado. Con el uso del operador *LIKE* podemos comparar patrones y ubicar un texto, independientemente de la posición en que se encuentre.

Para la definición del patrón de consulta existen dos tipos de caracteres especiales:

% (signo de porcentaje) llamado *comodín*, representa cualquier cantidad de espacios o caracteres en esa posición. Significa que se admite cualquier cosa en su lugar: un caracter, cien caracteres o ningún caracter.

_ (signo de subrayado) llamado *marcador de posición*, representa exactamente una posición e indica que puede existir cualquier caracter en esa posición.

En los fuentes 3, 4 y 5 detallamos tres ejemplos de consulta con el Operador *LIKE*:

Listar los clientes cuya dirección contengan la palabra *URUGUAY* independientemente de su ubicación:

```
Fuente 3

SELECT nombre,direccion,telefono
FROM   clientes
WHERE  direccion LIKE '%URUGUAY%';

Listar los clientes  cuyos teléfonos tienen comienzan con el código de área el
0722

SELECT nombre,direccion,telefono
FROM   clientes
WHERE  telefono LIKE '(0722)%';
```

Listar los clientes cuyo nombre terminan con la palabra *LIBRO*:

```
Fuente 4

SELECT nombre,direccion,telefono
FROM   clientes
WHERE  nombre LIKE '%LIBRO';
```

Listar los clientes que tengan la palabra *LIBRO* a partir de la 5ª posición en el nombre.

```
Fuente 5

SELECT nombre,direccion,telefono
FROM   clientes
WHERE  nombre LIKE '____LIBRO%';
```

## Agrupamiento de datos

SQL proporciona una forma eficiente para manejar la información con el agrupamiento de datos a través de la formación de grupos y las funciones correspondientes, dando la posibilidad de procesar no solo registros individuales como hemos hecho hasta ahora. También podemos agrupar registros por un criterio determinado, como por ejemplo, agrupar por clientes las ventas realizadas.

Cada grupo tendrá como resultado de la consulta una fila resumen que contiene la información del grupo.

Para la formación de grupos adicionamos, a la forma básica de la sentencia *SELECT* vista

anteriormente, la orden *GROUP BY* ubicada antes de *ORDER BY*, como se muestra a continuación:

```
SELECT     Lista...
FROM       Tabla, Tabla...
WHERE      Condiciones
GROUP BY   Expresión, Expresión,...
           ORDER BY  Expresión, Expresión,...
;          Terminador
```

Las funciones para el procesamiento de grupos son:

COUNT(columna) Cantidad de registros en que la columna tiene valores no nulos.

COUNT(*) Cantidad de registros que hay en la tabla, incluyendo los valores nulos.

MIN(columna) Valor mínimo del grupo.

MAX(columna) Valor máximo del grupo.

SUM(columna) Suma los valores del grupo.

AVG(columna) Calcula valor medio del grupo, sin considerar los valores nulos.

La lista de columnas a mostrar en la consulta puede contener las funciones de grupo, así como la columna o expresión usada para formar los grupos en la orden *GROUP BY*. En una misma consulta no se pueden mezclar funciones de grupo con columnas o funciones que trabajan con registros individuales.

Las ventas por cliente es un buen ejemplo para mostrar el uso de los grupos. En el siguiente caso se hace un resumen de ventas por cliente, con la cantidad de ventas, valor mínimo, medio y máximo, así como la suma total de ventas. La formación del grupo será por el nombre del cliente y la columna a cuantificar para cada grupo será el valor de las ventas.

```
SELECT nombre "CLIENTE",
     COUNT(valor) "VENTAS",
     MIN(valor) "MINIMA",
     AVG(valor) "MEDIA",
     MAX(valor) "MAXIMA",
     SUM(VALOR) "TOTAL"
   FROM clientes,ventas
   WHERE clientes.codigo=ventas.codigo
   GROUP BY nombre
;
```

| CLIENTE | VENTAS | MINIMA | MEDIA | MAXIMA | TOTAL |
|---|---|---|---|---|---|
| CASA AUGE DEPORTES | 3 | 33.0 | 138.667 | 330.0 | 416.0 |
| CLUB DE TENNIS | 3 | 100.5 | 237.167 | 500.5 | 711.5 |
| FERIA DEL LIBRO | 3 | 110.0 | 203.333 | 310.0 | 610.0 |

| | | | | | |
|---|---|---|---|---|---|
| PINTURERIAS PROPIOS | 3 | 20.5 | 90.500 | 220.5 | 271.5 |

Tabla 2: Estadística de la base de datos

Como se puede observar en la tabla 2, a cada columna se le asignó un título de cabecera, que se escribe entre comillas dobles a la derecha de la columna, con el objetivo de mejorar la presentación de la consulta.

El orden en las consultas por grupos, cuando no esta presente la orden *ORDER BY*, está dado por la columna que forma los grupos. Si deseamos cambiar ese orden, como es el caso de ordenar por el valor total de ventas, se debe adicionar al final la orden *ORDER BY SUM(VALOR).*

En el ejemplo del fuente 6 se forman grupos por artículo para obtener la cantidad de ventas, valor mínimo, medio y máximo, así como la suma total de ventas para cada artículo, ordenado de mayor a menor por la venta total.

```
Fuente 6

SELECT articulo, COUNT(valor) "VENTAS",MIN(valor) "MINIMA",
    AVG(valor) "MEDIA",MAX(valor) "MAXIMA",SUM(VALOR) "TOTAL"
    FROM ventas
    GROUP BY articulo
    ORDER BY SUM(valor) DESC
;
```

| ARTICULO | VENTAS | MINIMA | MEDIA | MAXIMA | TOTAL |
|---|---|---|---|---|---|
| PAPEL | 8 | 100.5 | 234.00 | 500.5 | 1872 |
| DISKETTE | 4 | 20.5 | 34.25 | 53.0 | 137 |

Tabla 3: Otra estadística de la base de datos

## Subconsultas

Otro aspecto de fácil diseño y uso que muestra una vez más las posibilidades de SQL son las subconsultas.

Subconsulta es aquella consulta de cuyo resultado depende otra consulta, llamada principal, y se define como una sentencia *SELECT* que esta incluida en la orden *WHERE* de la consulta principal. Una subconsulta, a su vez, puede contener otra subconsulta y así hasta un máximo de 16 niveles.

Las particularidades de las subconsultas son:

1. Su resultado no se visualiza, sino que se pasa a la consulta principal para su comprobación.

2. Puede devolver un valor único o una lista de valores y en dependencia de esto se debe usar el operador del tipo correspondiente.

3. No puede usar el operador *BETWEEN*, ni contener la orden *ORDER BY*.

4. Puede contener una sola columna, que es lo más común, o varias columnas. Este último caso se llama subconsulta con columnas múltiples. Cuando dos o más columnas serán comprobadas al mismo tiempo, deben encerrarse entre paréntesis.

Expliquemos como se construye una subconsulta con el siguiente ejemplo, donde necesitamos saber ¿cuál fue la mayor venta realizada?. Para ello, diseñemos una subconsulta que busque el valor máximo de venta con el uso de la función *MAX(valor)* y una consulta principal que muestre las ventas iguales al máximo valor encontrado por la subconsulta. Veamos el fuente 7.

```
Fuente 7

SELECT nombre,articulo,valor
FROM clientes,ventas
WHERE valor = ( Subconsulta para buscar
                SELECT MAX(valor)          el valor máximo de venta.
                FROM ventas
              )
      AND clientes.codigo=ventas.codigo
;
```

NOMBRE: CLUB DE TENNIS
ARTICULO: PAPEL
VALOR: 500.5

Otra aplicación clásica de la subconsulta es cuando deseamos saber las ventas de un artículo realizadas por encima de su venta promedio. En este caso, es necesario realizar los pasos mostrados en el fuente 8:

```
Fuente 8

SELECT   nombre,valor "PAPEL"
FROM     clientes,ventas
WHERE    clientes.codigo=ventas.codigo
    AND articulo='PAPEL'
    AND valor >
                (                              Subconsulta de
                 SELECT AVG(valor)             venta promedio de
                 FROM   ventas                 papel.
                 WHERE  articulo='PAPEL'
                )
ORDER BY valor DESC            Ordenado por valor de venta
;                              en forma descendente
```

| NOMBRE | PAPEL |
|---|---|
| CLUB DE TENNIS | 500.5 |
| CASA AUGE DEPORTES | 330.0 |
| FERIA DEL LIBRO | 310.0 |

Tabla 4: ventas por encima de su venta promedio

## Grupos con subconsulta

Hasta el momento estudiamos por separado un conjunto de herramientas de SQL, viendo en cada caso sus posibilidades. Ahora pasaremos a ver la combinación de grupos y subconsultas, lo que multiplica las posibilidades de SQL en cuanto al rendimiento en el diseño de consultas complejas se refiere, las cuales se pueden realizar en forma sencilla y con pocas líneas de código.

Para combinar grupos con subconsulta debemos incluir en la sentencia *SELECT* la orden *HAVING*, que tiene las siguientes características:

1. Funciona como la orden *WHERE*, pero sobre los resultados de las funciones de grupo, en oposición a las columnas o funciones para registros individuales que se seleccionan mediante la orden *WHERE*. O sea, trabaja como si fuera una orden *WHERE*, pero sobre grupos de registros.

2. Se ubica después de la orden GROUP BY.

3. Puede usar una función de grupo diferente a la de la orden SELECT.

El ejemplo a diseñar para nuestra aplicación es la consulta ¿cuál fue el artículo más vendido y en qué cantidad?. En este caso, la orden *HAVING* de la consulta principal selecciona aquellos artículos (*GROUP BY*) que tienen una venta total (*SUM(valor)*) igual a la mayor venta realizada por artículo (*MAX(SUM(valor))*) que devuelve la subconsulta.

La sentencia *SELECT* y el resultado de nuestra consulta sería la del fuente 9 y la tabla 5:

```
Fuente 9

SELECT articulo "ARTICULO MAS VENDIDO",SUM(valor) "VENTA"
    FROM ventas
    GROUP BY articulo
    HAVING SUM(valor) =
    (                              Subconsulta para buscar
     SELECT MAX(SUM(valor))        el artículo más vendido
     FROM ventas                   con la formación de grupos
     GROUP BY articulo             por artículo.
    )
;
```

| ARTICULO MÁS VENDIDO | VENTA |
|---|---|
| PAPEL | 1872 |

Tabla 4: ventas por encima de su venta promedio

## Indices

El índice es un instrumento que aumenta la velocidad de respuesta de la consulta, mejorando

su rendimiento y optimizando su resultado. El manejo de los índices en *ORACLE* se realiza de forma *inteligente*, donde el programador sólo crea los índices sin tener que especificar, explícitamente, cuál es el índice que va a usar. Es el propio sistema, al analizar la condición de la consulta, quien decide qué índice se necesita. Por ejemplo cuando en una consulta se relacionan dos tablas por una columna, si ésta tiene definido un índice se activa, como en el caso cuando relacionamos la tabla de clientes y ventas por la columna código para identificar al cliente (*WHERE clientes.codigo=ventas.codigo*)

La identificación del índice a usar está relacionada con las columnas que participan en las condiciones de la orden *WHERE*. Si la columna que forma el índice está presente en alguna de las condiciones éste se activa. No obstante, existen casos en que la presencia de la columna no garantiza el uso de su índice, ya que éstos son ignorados, como en las siguientes situaciones cuando la columna indexada es:

Evaluada con el uso de los operadores *IS NULL* o *IS NOT NULL*.

```
SELECT nombre,articulo,valor
FROM    clientes,ventas
WHERE  nombre IS NOT NULL;
```

Modificada por alguna función, excepto por las funciones *MAX(columna)* o *MIN(columna)*.

```
SELECT nombre,articulo,valor
FROM    clientes,ventas
WHERE UPPER(nombre)>' '
;
```

Usada en una comparación con el operador *LIKE* a un patrón de consulta que comienza con alguno de los signos especiales (*% _*).

```
SELECT nombre,articulo,valor
       FROM    clientes,ventas
       WHERE  nombre
       LIKE '%DEPORTE%'
;
```

Finalmente debemos aclarar que los registros cuyo valor es *NULL* para la columna indexada, no forman parte del índice. Por lo tanto, cuando el índice se activa estos registros no se muestran como resultado de la consulta.

Con lo estudiado en nuestros dos primeros artículos sobre *Oracle Básico*, aprendimos a crear nuestras tablas e índices y a construir las consultas. Por lo tanto, ya tenemos los elementos necesario para analizar las particularidades del *Diseño de Pantallas* con el módulo *SQLFORMS*, con el objetivo de ingresar, modificar, eliminar y consultar las tablas en forma amigable y en la medida de las necesidades del usuario. A este tema dedicaremos nuestro próximo artículo.

## Bibliografía

ORACLE 7 Manual de Referencia

**Koch, George.**
*Osborne/McGraw-Hill*, 1994.

ORACLE Manual de Referencia.
**Koch, George.**
*Osborne/McGraw-Hill*, 1992.

Mastering Oracle.
**Cronin, Daniel.**
*Hayden Books*, 1990.