**TEMA 4 – REALIZACIÓN DE CONSULTAS**

1. **INTRODUCCIÓN**

Las sentencias SQL pueden ser escritas y ejecutadas de dos formas:

* Desde el entorno gráfico con SQLDeveloper.
* Desde el entorno de SQL\*Plus

1. **LA SENTENCIA SELECT**

Se usa para recuperar o seleccionar los datos de una o varias tablas.

Consta de 4 partes básicas:

* 1. **Cláusula SELECT**

A continuación de la clausua SELECT se debe especifiacar cada una de las columnas que queremos seleccionar.

Hay que tener en cuenta:

* Se pueden nombrar a las columnas anteponiendo el nombre de la tabla de la que proceden, es opcional.
* Para incluir todas las columnas de una tabla se utiliza el asterisco \*.
* Se pueden poner alias a los nombre de las columnas.
* Tambien se pude sustituir el nombre de las columnas por constantes, expresiones o funciones SQL.
  1. **Cláusula FROM**
* Con esta se establece de dónde se obtienen las columnas que vamos a seleccionar.
* En esta clausula se definen los nombres de las tablas de las que proceden las columnas.
* **Consulta combinada** o **Join**, si se utiliza más de una tabla, se separan por comas.
* Se puede añadir el nombre del usuario propietario de esta tablas, de este modo se puede distinguir entre las tablas de un user y otro.
* Se puede asociar a las tablas un alias, no es necesario entrecomillarlo.
  1. **Cláusula WHERE**

Será donde pongamos la condición que han de cumplir las filas para salir como resultado de dicha columna.

* 1. **Cláusula ORDER BY**

Se utiliza para especificar el criterio de ordenación de la respuesta a nuestra consulta.

La ordenación por defecto es descendente, aunque, después de cada columna de ordenación se puede incluir el tipo de orenación :

* **ASC** -> ordenación ascendente.
* **DESC** -> ordenación descendente.

Se puede ordenar por más de una columna.

Se puede colocar el número de orden del campo por el que se quiere ordenar en lugar del nombre, es decir, referenciar a los campos por su posición en la lista de selección.

Solo es posible ordenar campos tipo carácter, número o fecha.

1. **OPERADORES**

Los operadores son símbolos que permiten realizar operaciones matemáticas, concatenar cadenas o hacer comparaciones.

Oracle reconoce 4 tipo se operadores:

* 1. **Operadores de Comparación**

Llamados **relacionales** en informática, son símbolos usados para comparar dos valores, si el resultado es correcto la expresión es True, en caso contrario es False.

| **OPERADOR** | **SIGNIFICADO** |
| --- | --- |
| **=** | Igualdad. |
| **!=, < >, ^=** | Desigualdad. Distinto |
| **<** | < |
| **>** | Mayor que. |
| **<=** | Menor o igual que. |
| **>=** | Mayor o igual que. |
| **IN** | Igual que cualquiera de los miembros entre paréntesis. |
| **NOT IN** | Distinto que cualquiera de los miembros entre paréntesis. |
| **BETWEEN** | Entre. Contenido dentro del rango. |
| **NOT BETWEEN** | Fuera del rango. |
| **LIKE '\_abc%'** | Se utiliza sobre todo con textos y permite obtener columnas cuyo valor en un campo cumpla una condición textual. Utiliza una cadena que puede contener los símbolos "%" que sustituye a un conjunto de caracteres o "\_" que sustituye a un carácter. |
| **IS NULL** | Devuelve verdadero si el valor del campo de la fila que examina es nulo. |

* 1. **Operadores Aritméticos y Concatenación**

Permiten realizar cálculos con valores numéricos:

| **OPERADOR** | **SIGNIFICADO** |
| --- | --- |
| **+** | Suma |
| **-** | Resta |
| **\*** | Multiplicación |
| **/** | División |

Usando expresiones con operadores es posible obtener salidas en las cuales una columna sea el resultado de un cálculo y no un capmpo.

Si una operación aritmética se calcula sobre un valor NULL, el resultado es NULL.

Para concatenar strings existe el operador concatenacion ||.

* 1. **Operadores Lógicos**

Se usán para comprobar si se cumple una u otra condición o ninguna de ellas.

| **OPERADOR** | **SIGNIFICADO** |
| --- | --- |
| **AND** | Devuelve verdadero si sus expresiones a derecha e izquierda son ambas verdaderas. |
| **OR** | Devuelve verdadero si alguna de sus expresiones a derecha o izquierda son verdaderas. |
| **NOT** | Invierte la lógica de la expresión que le precede, si la expresión es verdadera devuelve falsa y si es falsa devuelve verdadera. |

* 1. **Precedencia**

Orden de precedencia en Oracle es el siguiente:

* + - 1. Se evalúa la multiplicación y la división al mismo nivel.
      2. Se evaluan las sumas y restas.
      3. Concatenación.
      4. Todas las comparaciones.
      5. Se evaluan los operadores IS (NOT) NULL, LIKE, BETWEEN.
      6. NOT.
      7. AND.
      8. OR.

En casos de igualdad se evalúa de izquierda a derecha.

Para variar su orden se utilizan paréntesis.

1. **CONSULTAS CALCULADAS**

Se obtienen a través de la sentencia **SELECT**  poniendo a continuación la expresión.

Esta consulta no modifica los valores originales de las columnas ni de la tabla a la que se está obteniendo dicha consulta, únicamente muestra una columna nueva con los valores calculados.

1. **FUNCIONES**

Son operaciones que se realizan sobre los datos y que realizan un determinado cálculo, para ello necesitna datos de entrada (parámetros o argumentos), en funcion de estos se realizará el cálculo de la función.

Las funciones se pueden incluir en las cláusualas SELECT, WHERE, ORDER BY.

* 1. **Funciones numéricas**
* **ABS (n)**, calcula el valor absoluto.
* **EXP (n)**, calcula el exponente en base e.
* **CEIL (n)**, calcula el valor entero inmediatamente superior o igual a n.
* **FLOOR (n)**, calcula el valor entero inmediatamente inferior o igual a n.
* **MOD (m,n)**, calcula el resto resultante de dividir m entre n.
* **POWER (valor, exponente)**, eleva el valor al exponente indicado.
* **ROUND (n, decimales)**, Redondea el número al siguiente con los dec. indicados
* **SQRT (n)**, calcula la raiz cuadrada.
* **TRUN (m,n)**, trunca un número a la cantidad de dicimales especificada n.
* **SING (n)**, si n es positivo retorna 1, si es negativo devuelve -1 y 0 si es 0.
  1. **Funciones de Cadena de Caracteres**
* **CHR (n)**, devuelve el carácter cuyo valor es n.
* **ASCII (n)**, devuelve el valor ASCII de n.
* **CONTACT (st1, st2)**, devuelve 2 strings contatenadas, equivale a ||.
* **LOWER (st)**, devuelve el string en minúsculas.
* **UPPER (st)**, devuelve el string en mayusculas.
* **INITCAP (st)**, devuelve el estring con su primer carácter en mayusculas.
* **LPAD (st1, n, st2)**, devuelve st1 con lenght n, ajustada a la derecha rellenado por la izquierda con st2
* **RPAD (st1, n st2)**, devuelve st1 con length n, ajustada a la izquierda rellenando por la derecha con st2.
* **REPLACE (st, old, new)**, devuelve st en el que cada ocurencia del string old ha sido sustituida por el string new.
* **SUBSTR (st, m, n)**, obtiene un substring de un string, devuelve el string st compuesto por n caracteres a partir de la posición m.
* **LENGTH (st)**, devuelve la longitud del st.
* **TRIM (st)**, elimina los espacios en blanco a la izquierda y derecha de st y los espacios dobles del interior.
* **LTRIM (st)**, elimina los espacios a la izquierda de st.
* **RTRIM (st)**, elimina los espacios a la derecha de st.
* **INSTR (st, stSearched[,posInt [,posAparicion]])**, obtiene la **posición** en la que se encuentra la cadena buscada en la cadena inicial.

Se puede comenzar a buscar desde una posición inicial concreta e incluso indicar el número de aparición de la cadena buscada.

Si no encuentra nada devuelve cero.

* 1. **Funciones de Manejo de Fechas**

Oracle tiene 2 tipos de datos para mejar fechas:

* **DATE**, almacena fechas concretas incluyendo a veces la hora.
* **TIMESTAMP**, almacena un instante de tiempo concreto, puede incluir hasta fracciones de segundo.

Podemos realizar operaciones numéricas con las fechas:

* Sumarle números, se entiende como sumarle días, h, min y seg.
* Restarle números, se entiende como restarle días, h, min y seg.
* Calcular diferencias entre 2 fechas, dara el número de días entre ellas.

Las funciones más comunes en Oracle:

* **SYSDATE**, devuele la fecha y hora actuales.
* **SYSTIMESTAMP**, devuelve la fecha y hora actuales en formato TIMESTAMP.
* **ADD\_MONTH (date, n)**, añade a la fecha el número de meses indicado.
* **MONTHS\_BETWEEN (d1, d2)**, devuelve el número de meses entre las fechas.
* **LAST\_DAY (date)**, devuelve el último día del mes al que pertenece la fecha.
* **NEXT\_DAY (date, d)**, indica el día que corresponde si añadimos al date el dia d
* **EXTRACT (valor FROM date)**, extrae un valor de una fecha concreta.
* Tambien se pueden emplear los operadores **+** y **-** .
  1. **Funciones de Conversión**

Funciones que pueden pasar de un tipo de dato a otro.

* **TO\_NUMBER(**st, format**)**, convierte en números, se suele usar para dar un formato concreto a los números.

| **Símbolo** | **Significado** |
| --- | --- |
| **9** | Posiciones numéricas. Si el número que se quiere visualizar contiene menos dígitos de los que se especifican en el formato, se rellena con blancos. |
| **0** | Visualiza ceros por la izquierda hasta completar la longitud del formato especificado. |
| **$** | Antepone el signo de dólar al número. |
| **L** | Coloca en la posición donde se incluya, el símbolo de la moneda local (se puede configurar en la base de datos mediante el parámetro **NSL\_CURRENCY**) |
| **S** | Aparecerá el símbolo del signo. |
| **D** | Posición del símbolo decimal, que en español es la coma. |
| **G** | Posición del separador de grupo, que en español es el punto. |

* **TO\_CHAR (d, format)**, convierte un número o fecha en una cadena de caracteres, se usa normalmente para fechas ya que de número a texto se hace de forma implicita como hemos visto antes.
* **TO\_DATE (st, format)**, Convierte textos a fechas, podemos indicar el formato con el que queremos que aparezca.

Para estas 2 últimas funciones en el caso de las fechas, indicamos el formato incluyendo los siguientes símbolos:

| **Símbolo** | **Significado** |
| --- | --- |
| **YY** | Año en formato de dos cifras |
| **YYYY** | Año en formato de cuatro cifras |
| **MM** | Mes en formato de dos cifras |
| **MON** | Las tres primeras letras del mes |
| **MONTH** | Nombre completo del mes |
| **DY** | Día de la semana en tres letras |
| **DAY** | Día completo de la semana |
| **DD** | Día en formato de dos cifras |
| **D** | Día de la semana del 1 al 7 |
| **Q** | Semestre |
| **WW** | Semana del año |
| **AM** **PM** | Indicador a.m.  Indicador p.m. |
| **HH12** **HH24** | Hora de 1 a 12  Hora de 0 a 23 |
| **MI** | Minutos de 0 a 59 |
| **SS** **SSSS** | Segundos dentro del minuto  Segundos dentro desde las 0 horas |

* 1. **Otras Funciones – NVL y DECODE**

Son funciones con NULLs que nos permiten hacer algo en caso de que aparezca un valor nulo:

* **NVL (valor, expr1)**, si el valor es NULL, devuelve expr1.
* **DECODE (expr1, cond1, valor1 [, cond2, valor2, …], default)**, evalúa una expresión exprl, si se cumple la primera cond1 devuelve valor1, en caso contrario evalúa la siguiente condición.

Si no se cumple ninguna condición se devuelve el valor default.

1. **CONSULTAS DE RESUMEN**

La sentencia SELECT nos permite obtener resúmenes de los datos de modo vertical -> consta de una serie de cláusulas específicas -> GROUP BY, HAVING.

Tenemos también unas funciones de agrupamiento o agregado que son las que nos dirán qué cálculos queremos realizar sobre los datos, toman un grupo de datos y producen un único dato que resume el grupo.

Estructura de una función de agregado:

***FUNCTION ([ALL | DISTINCT] Expresión)***

* **ALL**, indica que se tienen que tomar todos los valores de la columna, es el valor por defecto.
* **DISTINCT**, indica que se consideran todas las repeticiones del mismo valor como uno solo (considera valores distintos).
* El grupo de valores sobre los que actúa la función lo determina el resultado de la **expresión** que será el nombre de una columna o una expresión basada en una o varias columnas.
* Todas las funciones se aplican a la filas del origen de datos una vez ejecutada la cláusula **WHERE** (si la tuviéramos).
* Todas las funciones (execept COUNT) ignoran los valores **NULL**.
* Se puede encontrar una función de agrupamiento dentro de una lista de selección en cualquier sitio donde pudiera aparecer el nombre de una columna.
* No se pueden mezclar funciones de columna con nombres de columna ordinarios, aunque hay excepciones.
  1. **Funciones de Agregado – SUM y COUNT**

***SUM ([ALL | DISTINCT] Expression)***

Devuelve la suma de los valores de la expresión.

Solo puede usarse con columna cuyo tipo de dato sea número.

***COUNT ([ALL | DISTINCT] Expression)***

Cuenta los elementos de un campo.

Expresión contiene el nombre del campo que deseamos contar.

Los operandos de expresión pueden incluir el nombre del campo, una cte., una función o el carácter \* en cuyo caso contaría el número de filas que cumplen la condición especificada, si la hay.

Puede contar cualquier tipo de datos. Simplemente cuenta el número de registros sin tener en cuenta qué valores se almacenan.

No cuenta registros NULL a menos que expresión sea el carácter comodín asterisco \*, así contaremos el total de filas incluyendo las NULL.

* 1. **Funciones de Agregado – MIN y MAX**

***MIN ([ALL | DISTINCT] Expression)***

Devuelve el valor mínimo de la expresión sin considerar los NULL.

En la expresión podemos incluir el nombre de un campo de una tabla, una cte. o una función.

***MAX ([ALL | DISTINCT] Expression)***

Devuelve el valor máximo de la expresión sin considerar los NULL.

En la expresión podemos incluir el nombre de un campo de una tabla, una cte. o una función.

* 1. **Funciones de Agregado – AVG, VAR y STDEV**

***AVG ([ALL | DISTINCT] Expression)***

Devuelve el promedio o media de los valores de un grupo, se omiten los valores NULL.

El grupo de los valores será el que se obtenga como resultado de la expresión y ésta puede ser un nombre de columna o una expresión basada en una columna o varias de la tabla.

Se aplica a campos tipo número.

***VAR ([ALL | DISTINCT] Expression)***

Devuelve la varianza estadística de todos los valores de la expresión.

Solo admite únicamente columnas numéricas.

Los valores NULL se omiten.

***STDEV ([ALL | DISTINCT] Expression)***

Devuelve la desviación típica estadística de todos los valores de la expresión.

Solo admite columnas numéricas.

Los NULL se omiten.

1. **AGRUPAMIENTO DE REGISTROS**

Agrupados según un determinado campo -> Calculo de totales parciales al utilizar consultas de resumen.

En la cláusula GROUP BY se colocan las columnas a agrupar.

En la cláusula HAVING se especifica la condición que han de cumplir los grupos para que se realice la consulta.

Orden de ejecución de las cláusulas:

* + - 1. WHERE, filtra las filas según las condiciones propuesta.
      2. GROUP BY, crea una tabla de grupos nueva.
      3. HAVING, filtra los grupos.
      4. ORDER BY, ordena o clasifica la salida.

1. **CONSULTAS MULTITABLAS**

Es posible realizar consultar usando varias tablas en la misma sentencia SELECT lo que permitirá realizar distintas operaciones como:

* 1. **Composiciones Internas**

Se indica poniendo la clausula FROM en las tablas a componer separadas por comas, pudiendo obtener el producto cartesiano de las tablas.

Hay que tener en cuenta que se obtienen todas las combinaciones posibles entre tablas, así que, puede ser una operación muy costosa.

**JOIN – Asociar Tablas** -> Discriminar para que únicamente aparezcan filas de una tabla que estén relacionadas con la otra tabla.

Reglas para las composiciones:

* Pueden combinarse tantas tablas como se quiera.
* El criterio de combinación puede ser formado por más de una pareja de columnas.
* En la cláusula SELECT pueden citarse columnas de ambas tablas, condicionen o no, la combinación.
* Si hay columnas con el mismo nombre en distintas tablas, deben clasificarse o identificarse especificando la tabla de procedencia seguida de un punto o utilizando un alias de tabla.

**Columnas de Join o emparejamiento**, las relacionadas que aparecen en la cláusula WHERE, no tienen por que estar incluidas en la lista de selección.

Se emparejarán tablas relacionadas entre sí siendo usualmente las columnas de emparejamiento la clave principal y la clave ajena.

**NombreTabla1.Camporelacionado1 = NombreTalba2.Camporelacionado2**

* 1. **Composiciones Externas**

“Si queremos que aparezcan registros de dos tablas, aunque en uno de ellos aparezcan valores NULL”.

Se añade un signo + entre paréntesis (+) en la igualdad entre campos que ponemos en la cláusula WHERE.

(+) irá detrás del nombre de la tabla en la que queremos aceptar valores nulos.

* 1. **Composiciones en la Versión SQL99**

En esta versión se incluyen mejoras de sintaxis:

* **CROSS JOIN**, crea un producto cartesiano de las filas de ambas tablas por lo que no es necesario WHERE.
* **NATURAL JOIN**, detecta las claves de unión, basándose en el nombre de la columna que coincide en ambas tablas, funciona incluso si no están definidas las claves primarias o ajenas.
* **JOIN USING**, permite establecer relaciones indicando qué campo o campos comunes se quieren utilizar para ello.
* **JOIN ON**, une tablas en las que los nombres de columna no coinciden en ambas tablas o se necesita establecer asociaciones más complicadas.
* **OUTER JOIN**, composiciones externas sin utilizar (+).
* **LEFT OUTER JOIN**, composición externa izquierda, todas las filas de la tabla de la izquierda se devuelven, aunque no haya ninguna columna correspondiente en las tablas combinadas.
* **RIGTH OUTER JOIN**, composición externa derecha, todas las filas de la tabla de la derecha se devuelven, aunque no haya ninguna columna correspondiente en las tablas combinadas.
* **FULL OUTER JOIN**, composición externa en la que se devolverán todas las filas de los campos no relacionados de ambas tablas.

1. **OTRAS CONSULTAS MULTITABLAS**

* **UNION**, combina las filas de un primer SELECT con las de otro, desapareciendo las duplicadas.
* **INTERSECT**, examina las filas de dos SELECT y devuelve las que aparezcan en ambos conjuntos, las duplicadas se eliminarán.
* **MINUS**, devuelve aquellas filas que están en el primer SELECT pero no en el segundo, las filas duplicadas del primer SELECT se reducen a una antes de comenzar la operación.

1. **SUBCONSULTAS**

La subconsulta o subselect, puede ir dentro de las clausulas WHERE, HAVING o FROM.

Dependiendo de los operadores usados pueden devolver 1 o varias filas:

**Operadores**

>, <, >=, <=, ¡=, = o IN.

Las subconsultas que utilizan estos devuelven un único valor, si devolviera más de un valor devolvería un error.

El tipo de dato que devuelve la subconsulta y la columna con la que se compara ha de ser el mismo.

Cuando el resultado de la subconsulta es **más de una fila**, SQL utiliza palabras reservadas entre el operador y la consulta:

* **ANY**, compara con cualquier fila de la consulta.

Es válida si hay un registro en la subconsulta que permite que la comparación sea cierta.

* **ALL**, compara todas las filas de la consulta.

La instrucción es cierta si es cierta la comparación con todas las filas devueltas por la consulta.

* **IN**, comprueba si el valor se encuentra en el resultado de la subconsulta.
* **NOT IN**, comprueba si un valor no se encuentra en una subconsulta.