Libro capítulo 4 funciones de caracteres

sábado, 22 de febrero de 2025 08:06 p. m.

LECTURA DEL DIAGRAMA DE SINTAXIS

TABLA 4.1 Símbolos de sintaxis

SÍMBOLO	USO
[ ]	Los corchetes encierran opciones de sintaxis.
{ }	Las llaves encierran elementos de los cuales solo se requiere uno.
|	Una barra vertical indica opciones.
. . .	Tres puntos indican que la expresión anterior puede repetirse.
Delimitadores	Los delimitadores que no sean corchetes, llaves, barras o los tres puntos deben ingresarse exactamente como se muestra en la sintaxis. Ejemplos de tales delimitadores son comas, paréntesis, etcétera.
CAPS	Las palabras en mayúsculas indican las palabras clave de Oracle que identifican los elementos individuales del comando SQL o el nombre de la función SQL. No importa si la palabra clave o el comando están en mayúsculas, para facilitar la lectura.
SUBRAYAR	Los valores predeterminados están subrayados.

FUNCIONES DE LOS CARACTERES

Todas las funciones de caracteres requieren parámetros de entrada alfanuméricos. La entrada puede ser un literal de texto o un literal de carácter, a veces denominado cadena, texto , constante o una columna de tipo de datos VARCHAR2, CHAR o CLOB. Los literales de texto siempre están rodeados por comillas simples.

FUNCION LOWER: La función LOWER transforma los datos a minúsculas. La siguiente consulta muestra cómo una columna y una constante de texto sirven como parámetros individuales para la función LOWER.
Sintaxis basica: UPPER(cadena)
Ejemplo de uso: SELECT state, LOWER(state), LOWER('State') FROM zipcode.

En los resultados de la consulta, los nombres de las columnas aparecen como LO Y LOWER porque SQL asigna nombres a las columnas basándose en las expresiones que se utilizan en la consulta. Veamos cómo se generan:

1. state → El nombre de la columna se mantiene igual porque se selecciona directamente de la tabla.
2. LOWER(state) → SQL usa el nombre de la función aplicada como encabezado de columna, es decir, LOWER(state).
3. LOWER('State') → Como LOWER('State') es una expresión sin un alias, SQL la muestra tal cual.

Para darles nombres más legibles, se puede usar AS para asignar alias:

```
SELECT state AS original_state,
       LOWER(state) AS state_lowercase,
       LOWER('State') AS static_lowercase
FROM zipcode;
```

FUNCIONES UPPER Y INICAT: La función UPPER es lo exactamente lo opuesto a la función LOWER: transforma los datos a mayúsculas. Y la función INITCAP pone en mayúscula la primera letra de una palabra y en minúscula el resto de la palabra.
Sintaxis basica: LOWER(cadena)
Sintaxis basica: INITCAP(cadena)
Ejemplo de uso: SELECT UPPER(city) as "Upper Case City", state, INITCAP(state) FROM zipcode WHERE zip = '10035'

Uso de funciones en la cláusula WHERE
A continuación, te explico cómo puedes usar estas funciones dentro de una cláusula WHERE para realizar filtrados específicos:

1. Usando UPPER en WHERE
La función UPPER se utiliza en la cláusula WHERE para hacer comparaciones sin tener en cuenta si el texto está en mayúsculas o minúsculas. Esto es útil cuando no sabes si los datos están almacenados en mayúsculas o minúsculas y quieres hacer una búsqueda sin importar la capitalización.
Ejemplo: Supongamos que tienes una tabla empleados con una columna nombre, y quieres buscar todos los empleados cuyo nombre sea "juan" sin importar cómo está capitalizado el texto.

SELECT * FROM empleados WHERE UPPER(nombre) = 'JUAN';
◇ Explicación: La función UPPER(nombre) convierte el valor de la columna nombre a mayúsculas para hacer la comparación, y la cadena 'JUAN' también está en mayúsculas, por lo que se ignorará el caso y buscará todos los registros cuyo nombre sea "juan", "Juan", "JUAN", etc.

2. Usando LOWER en WHERE
De forma similar, la función LOWER se puede utilizar para hacer comparaciones sin importar si las letras están en mayúsculas o minúsculas.
Ejemplo: Siguiendo con el ejemplo anterior, si quieres buscar todos los empleados cuyo nombre sea "juan" sin importar la capitalización, puedes usar LOWER.

SELECT * FROM empleados WHERE LOWER(nombre) = 'juan';
◇ Explicación: La función LOWER(nombre) convierte el valor de la columna nombre a minúsculas antes de hacer la comparación. Así, la consulta buscará nombres como "juan", "Juan", "JUAN", etc., sin importar cómo estén almacenados en la base de datos.

3. Usando INITCAP en WHERE
La función INITCAP convierte la primera letra de cada palabra a mayúsculas y el resto a minúsculas. Esto también puede ser útil si necesitas buscar registros con un formato estandarizado.
Ejemplo: Supongamos que tienes nombres de empleados en la columna nombre y quieres buscar un nombre específico, pero quieres asegurarte de que la búsqueda sea insensible al caso y también normalice la capitalización de las palabras.
SELECT * FROM empleados WHERE INITCAP(nombre) = 'Juan Perez';
◇ Explicación: La función INITCAP(nombre) convierte la primera letra de cada palabra de la columna nombre a mayúscula y el resto de las letras a minúsculas. Esto hace que "juan perez", "JUAN PEREZ", o "jUaN pErEz" coincidan con la búsqueda de 'Juan Perez'.

FUNCIONES LPAD Y RPAD
Las funciones LPAD y RPAD en Oracle se utilizan para rellenar cadenas de texto a la izquierda o a la derecha, respectivamente, con un carácter específico hasta alcanzar una longitud deseada. Ambas funciones son muy útiles cuando necesitamos asegurarnos de que las cadenas de texto tengan un tamaño uniforme o si necesitamos agregar caracteres para alinear datos en reportes.

1. Función LPAD (Left PAD): La función LPAD se utiliza para agregar caracteres al principio (izquierda) de una cadena hasta que alcanza una longitud específica.
Sintaxis: LPAD(cadena, longitud_deseada, [carácter])
- cadena: Es la cadena original a la que se le agregará el relleno.
- longitud_deseada: Es la longitud final que debe tener la cadena después de agregar los caracteres. Si la longitud de la cadena original es mayor que la longitud deseada, la cadena no se recorta.

- carácter (opcional): Es el carácter que se utilizará para rellenar. Si no se especifica, el carácter predeterminado es un espacio en blanco.

2. Función RPAD (Right PAD)
La función RPAD se utiliza para agregar caracteres al final (derecha) de una cadena hasta alcanzar una longitud específica.
Sintaxis: RPAD(cadena, longitud_deseada, [carácter])

- cadena: Es la cadena original a la que se le agregará el relleno.
- longitud_deseada: Es la longitud final que debe tener la cadena después de agregar los caracteres. Al igual que con LPAD, si la longitud de la cadena original es mayor que la longitud deseada, la cadena no se recorta.
- carácter (opcional): Es el carácter que se utilizará para rellenar. Si no se especifica, el carácter predeterminado es un espacio en blanco.

Consideraciones:

- Ajuste de longitudes de cadenas: Las funciones LPAD y RPAD se utilizan comúnmente para asegurar que las cadenas de texto tengan una longitud uniforme, especialmente cuando trabajamos con reportes o formatos específicos.
- Relleno con caracteres personalizados: Si deseas rellenar con caracteres diferentes a los espacios en blanco, puedes especificar el carácter que desees. Por ejemplo, se pueden rellenar números con ceros para un formato específico, como en los números de factura o código de productos.
- Longitud mayor que la cadena original: Si la longitud deseada es menor o igual a la longitud de la cadena original, no se recortará la cadena, solo se completará con los caracteres especificados hasta alcanzar la longitud deseada.
- Limitación en el uso de caracteres: El carácter de relleno puede ser cualquier letra, número o símbolo que desees, pero en Oracle, este puede ser un solo carácter. Si deseas rellenar con múltiples caracteres, debes usar subcadenas.

Uso de LPAD y RPAD en WHERE
Al igual que con las funciones de capitalización como UPPER y LOWER, puedes utilizar LPAD y RPAD para modificar el formato de las cadenas antes de hacer una comparación o filtrado. Esto es útil cuando, por ejemplo, necesitas hacer una búsqueda de cadenas con longitudes específicas o asegurarte de que las cadenas tengan un formato de relleno antes de comparar.

1. Usando LPAD en WHERE
La función LPAD agrega caracteres al inicio (izquierda) de una cadena hasta que alcanza una longitud especificada. Esto puede ser útil si los datos en la base de datos tienen una longitud variable y quieres compararlos con una longitud estándar.
Ejemplo: Supongamos que tienes una tabla productos con una columna codigo_producto, y los códigos tienen diferentes longitudes. Quieres buscar todos los productos cuyo código, cuando se le agregan ceros a la izquierda hasta alcanzar una longitud de 6 caracteres, sea igual a "000123".

```
SELECT *  FROM productos  WHERE LPAD(codigo_producto, 6, '0') = '000123';
```

◇ Explicación: LPAD(codigo_producto, 6, '0'): Esta función agrega ceros a la izquierda del valor en la columna codigo_producto hasta que la longitud total sea 6 caracteres.
La consulta buscará todos los productos cuyo codigo_producto sea igual a '000123', independientemente de la longitud original del código en la base de datos, siempre que se rellene con ceros hasta alcanzar 6 caracteres.

2. Usando RPAD en WHERE
De forma similar, RPAD agrega caracteres al final (derecha) de una cadena hasta alcanzar una longitud específica. Esto puede ser útil cuando quieres asegurarte de que los datos tengan un formato consistente con relleno al final de la cadena.
Ejemplo: Siguiendo con el ejemplo anterior, supongamos que quieres buscar todos los productos cuyo código, cuando se le agregan espacios al final hasta alcanzar una longitud de 6 caracteres, sea igual a "123 ".

```
SELECT *  FROM productos  WHERE RPAD(codigo_producto, 6, ' ') = '123   ';
```

◇ Explicación: RPAD(codigo_producto, 6, ' '): Esta función agrega espacios al final del valor de la columna codigo_producto hasta que la longitud total sea 6 caracteres.
La consulta buscará todos los productos cuyo codigo_producto sea igual a '123 ', agregando espacios al final para completar la longitud de 6.

LA TABLA DUAL
La tabla DUAL en Oracle es una tabla especial que tiene una sola fila y una sola columna. Esta tabla se utiliza comúnmente en consultas cuando necesitas realizar operaciones que no dependen de ninguna tabla en particular, como cálculos matemáticos, funciones, o expresiones que no requieren una fuente de datos específica. Es una tabla virtual creada automáticamente en todas las bases de datos Oracle, y se encuentra en el esquema SYS. Su propósito principal es simplificar la escritura de consultas sin necesidad de utilizar una tabla real.

Características de la tabla DUAL:
- Solo una fila y una columna: La tabla DUAL tiene una única fila y una única columna llamada DUMMY, que contiene un valor único, típicamente 'X'.
- Usos comunes: Se utiliza para:
    + Realizar cálculos y expresiones sin necesidad de una tabla real.
    + Obtener la fecha y hora actuales usando funciones como SYSDATE.
    + Ejecutar funciones como UID, USER, CURRVAL, entre otras.

Pertenece al usuario SYS, que es el propietario de todas las tablas del sistema Oracle. Esta tabla, como muchas otras tablas del sistema Oracle, es accesible para todos los usuarios a través de un sinónimo público

FUNCIONES LTRIM, RTIM Y TRIM
Las funciones LTRIM, RTRIM y TRIM en Oracle se utilizan para eliminar espacios en blanco o caracteres específicos de las cadenas de texto. Estas funciones son muy útiles cuando tienes datos que contienen espacios o caracteres no deseados al principio o al final de las cadenas y quieres limpiarlas antes de realizar una comparación o análisis.

1. LTRIM (Left Trim)
La función LTRIM elimina los espacios o caracteres especificados desde el inicio (izquierda) de una cadena de texto. Sintaxis: LTRIM(cadena [, caracteres_a_eliminar])
- cadena: La cadena de texto de la que quieres eliminar los caracteres al principio.
- caracteres_a_eliminar (opcional): El conjunto de caracteres que deseas eliminar. Si no se especifica, se eliminarán los espacios en blanco por defecto.

Ejemplo:
Supongamos que tienes la siguiente cadena: " Hola Mundo" (con espacios antes de "Hola Mundo").
SELECT LTRIM(' Hola Mundo') FROM DUAL;
◇ Resultado: 'Hola Mundo'
Los espacios iniciales han sido eliminados.

Ejemplo con caracteres específicos:
Si quieres eliminar un carácter específico, por ejemplo, el carácter 'x':
SELECT LTRIM('xxxHola Mundo', 'x') FROM DUAL;
◇ Resultado: 'Hola Mundo'
Los caracteres 'x' al principio de la cadena han sido eliminados.

2. RTRIM (Right Trim)
La función RTRIM elimina los espacios o caracteres especificados desde el final (derecha) de una cadena de texto.
Sintaxis: RTRIM(cadena [, caracteres_a_eliminar])
- cadena: La cadena de texto de la que quieres eliminar los caracteres al final.
- caracteres_a_eliminar (opcional): El conjunto de caracteres que deseas eliminar. Si no se especifica, se eliminarán los espacios en blanco por defecto.

Ejemplo:
Supongamos que tienes la siguiente cadena: "Hola Mundo " (con espacios al final de "Hola Mundo").

SELECT RTRIM('Hola Mundo   ') FROM DUAL;

- Resultado: 'Hola Mundo'

Los espacios finales han sido eliminados.

Ejemplo con caracteres específicos:
Si quieres eliminar un carácter específico, por ejemplo, el carácter 'x':
SELECT RTRIM('Hola Mundoxxx', 'x') FROM DUAL;

- Resultado: 'Hola Mundo'

Los caracteres 'x' al final de la cadena han sido eliminados.

3. TRIM (Both Sides Trim)
La función TRIM elimina los espacios o caracteres especificados tanto al principio como al final de una cadena de texto.
Sintaxis: TRIM([LEADING|TRAILING|BOTH] [caracteres_a_eliminar] FROM cadena)

- Si desea que la función actúe como LTRIM, especifique LEADING como el primer parámetro; para RTRIM, utilice la opción TRAILING; Para ambos, especifique la palabra clave BOTH u omítala por completo.
- caracteres_a_eliminar (opcional): El conjunto de caracteres que deseas eliminar. Si no se especifica, se eliminarán los espacios en blanco por defecto.
- cadena: La cadena de texto de la que quieres eliminar los caracteres al principio y al final.

Ejemplo:
Supongamos que tienes la siguiente cadena: " Hola Mundo " (con espacios al principio y al final de "Hola Mundo").
SELECT TRIM('   ' FROM '   Hola Mundo   ') FROM DUAL;

- Resultado: 'Hola Mundo'

Los espacios al principio y al final han sido eliminados.

Ejemplo con caracteres específicos:
Si quieres eliminar un carácter específico, por ejemplo, el carácter 'x':
SELECT TRIM('x' FROM 'xxxHola Mundoxxx') FROM DUAL;

- Resultado: 'Hola Mundo'

Los caracteres 'x' al principio y al final de la cadena han sido eliminados.

Consideraciones:

- Si no se especifica un conjunto de caracteres, las funciones **LTRIM**, **RTRIM** y **TRIM** eliminarán los espacios en blanco por defecto.
- Estas funciones son útiles cuando deseas **limpiar datos** o **normalizar cadenas** de texto eliminando espacios innecesarios que pueden haber sido ingresados por los usuarios o por otros procesos de la base de datos.
- Si quieres quitar caracteres diferentes a los espacios, puedes especificar qué caracteres deseas eliminar usando el segundo parámetro (por ejemplo, '0', 'x', etc.).

FUNCION SUBSTR
La función SUBSTR en Oracle se utiliza para extraer una subcadena de una cadena de texto. Es muy útil cuando necesitas obtener una parte específica de una cadena a partir de una posición determinada.

Sintaxis de la función SUBSTR: SUBSTR(cadena, inicio [, longitud])

- cadena: La cadena de texto de la cual se extraerá la subcadena.
- inicio: La posición de inicio desde donde se comenzará a extraer la subcadena. La posición de inicio es 1 para el primer carácter de la cadena.
  Si el valor de inicio es negativo, la función comienza a contar desde el final de la cadena (por ejemplo, -1 es el último carácter, -2 es el penúltimo, etc.).
- longitud (opcional): La cantidad de caracteres que se extraerán a partir de la posición de inicio. Si no se especifica, se extraerá desde la posición de inicio indicada hasta el final de la cadena.

Ejemplos de uso de SUBSTR:
1. Extraer una subcadena desde una posición específica:
Supongamos que tienes la cadena 'Hola Mundo' y deseas extraer los primeros 4 caracteres:

SELECT SUBSTR('Hola Mundo', 1, 4) FROM DUAL;

- Resultado: 'Hola'

La subcadena comienza en la posición 1 (el primer carácter de la cadena) y extrae los siguientes 4 caracteres.

2. Extraer una subcadena desde una posición hasta el final:
Si no especificas la longitud, se extraerá desde la posición indicada hasta el final de la cadena.
SELECT SUBSTR('Hola Mundo', 6) FROM DUAL;

- Resultado: 'Mundo'

La subcadena comienza desde la posición 6 (el carácter 'M') y extrae el resto de la cadena hasta el final.

3. Extraer una subcadena con una posición negativa (contando desde el final):
Cuando usas un número negativo para el parámetro inicio, Oracle cuenta desde el final de la cadena.
Por ejemplo:
SELECT SUBSTR('Hola Mundo', -5, 5) FROM DUAL;

- Resultado: 'Mundo'

Comienza a contar desde el carácter 'M' (que es el quinto carácter desde el final de la cadena) y extrae los siguientes 5 caracteres.

4. Extraer una subcadena con solo una longitud mayor que el número de caracteres disponibles:
Si la longitud solicitada es mayor que el número de caracteres disponibles a partir de la posición de inicio, Oracle simplemente devolverá los caracteres que haya disponibles.
SELECT SUBSTR('Hola Mundo', 7, 20) FROM DUAL;

- Resultado: 'Mundo'

Aunque se pidieron 20 caracteres, solo hay 5 caracteres disponibles a partir de la posición 7, por lo que solo se devuelve 'Mundo'.

5. Extraer caracteres al final de una cadena con un valor negativo para inicio:
Si deseas obtener los últimos 3 caracteres de una cadena, puedes hacerlo de la siguiente manera:
SELECT SUBSTR('Hola Mundo', -3) FROM DUAL;

- Resultado: 'ndo'

Empieza desde el tercer carácter desde el final y extrae el resto de la cadena.

6. Extraer subcadenas de una cadena con números y texto:
Supongamos que tienes la cadena '12345-67890' y deseas extraer la parte antes del guion:
SELECT SUBSTR('12345-67890', 1, INSTR('12345-67890', '-') - 1) FROM DUAL;

- Resultado: '12345'

INSTR es otra función que devuelve la posición del primer carácter de una subcadena dentro de una cadena, y aquí se usa para encontrar el guion ('-'), y con SUBSTR se extrae la parte de la cadena que está antes del guion.

FUNCION INSTR
La función INSTR en Oracle se utiliza para buscar la posición de una subcadena dentro de una cadena de texto. Devuelve la posición inicial (índice) de la primera aparición de una subcadena dentro de la cadena de texto. Es muy útil cuando necesitas encontrar dónde empieza una subcadena dentro de una cadena más grande.

Sintaxis de la función INSTR: INSTR(cadena, subcadena [, inicio [, ocurrencia [, modo]]])

- cadena: La cadena de texto donde se va a buscar la subcadena.
- subcadena: La subcadena que se quiere buscar dentro de la cadena.
- inicio (opcional): La posición desde la cual empezar a buscar. El valor predeterminado es 1, que significa que la búsqueda comienza desde el principio de la cadena. Si es un valor negativo, la búsqueda se hace desde el final de la cadena.
- ocurrencia (opcional): El número de la ocurrencia que deseas encontrar. El valor predeterminado es 1, lo que significa que buscará la primera aparición de la subcadena.
- modo (opcional): Especifica si la búsqueda debe ser sensible a mayúsculas/minúsculas o no. Puedes usar 0 para no distinguir entre mayúsculas y minúsculas, o 1 para que sea sensible a

mayúsculas y minúsculas.

Ejemplos de uso de la función INSTR:
1. Buscar la posición de una subcadena:
Supongamos que tienes la cadena 'Hola Mundo' y deseas encontrar la posición de la subcadena 'Mundo':

SELECT INSTR('Hola Mundo', 'Mundo') FROM DUAL;

- Resultado: 6

La función devuelve 6, ya que 'Mundo' comienza en la posición 6 de la cadena 'Hola Mundo'.

2. Buscar la segunda ocurrencia de una subcadena:
Supongamos que tienes la cadena 'ABABABAB' y deseas encontrar la segunda aparición de la subcadena 'AB':
SELECT INSTR('ABABABAB', 'AB', 1, 2) FROM DUAL;

- Resultado: 3

La función devuelve 3, ya que la segunda ocurrencia de 'AB' comienza en la posición 3 de la cadena 'ABABABAB'.

3. Buscar desde una posición específica:
Si deseas buscar la subcadena 'AB' pero comenzando desde la posición 4, puedes hacerlo de la siguiente manera:
SELECT INSTR('ABABABAB', 'AB', 4) FROM DUAL;

- Resultado: 5

La búsqueda comienza desde la posición 4 y la subcadena 'AB' se encuentra nuevamente en la posición 5.

4. Búsqueda desde el final de la cadena (posición negativa):
Si deseas buscar desde el final de la cadena (en lugar de desde el principio), puedes usar un valor negativo en el parámetro inicio.
SELECT INSTR('ABABABAB', 'AB', -3) FROM DUAL;

- Resultado: 5

Aquí, la búsqueda comienza desde el tercer carácter desde el final de la cadena, y la primera ocurrencia de 'AB' se encuentra en la posición 5. Se lee de derecha a izquierda.

5. Hacer una búsqueda que no distinga entre mayúsculas y minúsculas:
Si deseas que la búsqueda no distinga entre mayúsculas y minúsculas, puedes usar 0 en el parámetro modo. Por ejemplo, buscar 'hOLA' en la cadena 'Hola Mundo':
SELECT INSTR('Hola Mundo', 'hOLA', 1, 1, 0) FROM DUAL;

- Resultado: 1

La función devuelve 1, ya que la búsqueda no distingue entre mayúsculas y minúsculas, por lo que 'hOLA' se encuentra en la posición 1.

NOTA:
Si la subcadena no se encuentra, la función INSTR devuelve 0.

FUNCION LENGTH
La función LENGTH en Oracle se utiliza para obtener la longitud de una cadena de texto, es decir, contar el número de caracteres que contiene. Esta función es útil para determinar el tamaño de una cadena, ya sea para validaciones, comparaciones o manipulación de datos.

Sintaxis de la función LENGTH: LENGTH(cadena)

- cadena: La cadena de texto de la cual deseas obtener la longitud.

Cómo funciona:
La función LENGTH devuelve un número entero que representa el número de caracteres en la cadena proporcionada, incluyendo cualquier espacio o carácter especial.

Ejemplos de uso de la función LENGTH:

1. Obtener la longitud de una cadena:
Supongamos que tienes la cadena 'Hola Mundo' y deseas conocer cuántos caracteres tiene:
SELECT LENGTH('Hola Mundo') FROM DUAL;

- Resultado: 10

La cadena 'Hola Mundo' tiene 10 caracteres, incluyendo el espacio entre "Hola" y "Mundo".

2. Obtener la longitud de una cadena con espacios:
Si tienes una cadena que contiene espacios al principio o al final, esos espacios también se cuentan. Por ejemplo, para la cadena ' Hola Mundo ' (con espacios antes y después):
SELECT LENGTH(' Hola Mundo ') FROM DUAL;

- Resultado: 14

Los espacios al principio y al final también se cuentan, por lo que la longitud total es 14.

3. Longitud de una cadena vacía:
Si pasas una cadena vacía, la función LENGTH devuelve 0:
SELECT LENGTH('') FROM DUAL;

- Resultado: 0

La cadena vacía tiene 0 caracteres.

4. Longitud de una cadena con caracteres especiales:
La función LENGTH también cuenta caracteres especiales, como saltos de línea o caracteres no alfabéticos.
SELECT LENGTH('Hola\nMundo') FROM DUAL;

- Resultado: 11

Aquí, '\n' (un salto de línea) se cuenta como un carácter, por lo que la longitud total es 11.

FUNCIONES EN WHERE Y ORDER BY
Las funciones en las cláusulas WHERE y ORDER BY son muy útiles para realizar filtrados o clasificaciones específicas de los datos en una consulta SQL. A continuación, te explico cómo funcionan y cómo puedes usarlas.

1. Funciones en la cláusula WHERE:
La cláusula WHERE se usa para filtrar los resultados de una consulta basándote en ciertas condiciones. Dentro de esta cláusula, puedes usar varias funciones de Oracle (como UPPER, LOWER, LENGTH, TRIM, entre otras) para aplicar condiciones más complejas.

Ejemplo: Función UPPER en la cláusula WHERE:
Si quieres filtrar los registros de una tabla de manera que no distinga entre mayúsculas y minúsculas, puedes usar la función UPPER para convertir los valores a mayúsculas antes de hacer la comparación.
SELECT * FROM empleados WHERE UPPER(nombre) = 'JUAN';

- Explicación: En este caso, UPPER(nombre) convierte el valor de la columna nombre a mayúsculas y lo compara con 'JUAN' en mayúsculas, sin importar cómo esté almacenado en la base de datos (por ejemplo, 'juan', 'Juan', 'JUAN', etc.).

Ejemplo: Función LENGTH en la cláusula WHERE:
Si deseas filtrar los registros de una tabla en base a la longitud de un texto, puedes usar la función LENGTH.
SELECT *
FROM empleados WHERE LENGTH(nombre) > 5;

- Explicación: Esta consulta selecciona a todos los empleados cuyo nombre tenga más de 5 caracteres.

Ejemplo: Función TRIM en la cláusula WHERE:
Si quieres eliminar los espacios en blanco de una cadena antes de hacer la comparación, puedes usar la función TRIM.
SELECT * FROM clientes WHERE TRIM(direccion) = 'Calle Falsa 123';

- Explicación: TRIM(direccion) elimina los espacios en blanco al principio y al final de la cadena

almacenada en la columna direccion antes de compararlo con 'Calle Falsa 123'.

2. Funciones en la cláusula ORDER BY:
La cláusula ORDER BY se utiliza para ordenar los resultados de una consulta según una o más columnas, ya sea en orden ascendente (ASC) o descendente (DESC). Al igual que con la cláusula WHERE, puedes usar funciones en la cláusula ORDER BY para controlar cómo se ordenan los datos.

Ejemplo: Función UPPER en la cláusula ORDER BY:
Si deseas ordenar los resultados sin importar las mayúsculas o minúsculas, puedes usar la función UPPER en la cláusula ORDER BY:
SELECT nombre FROM empleados ORDER BY UPPER(nombre);
- ◇ Explicación: La función UPPER(nombre) convierte los valores de la columna nombre a mayúsculas antes de ordenarlos, de manera que el orden no se vea afectado por las diferencias de mayúsculas y minúsculas.

Ejemplo: Función LENGTH en la cláusula ORDER BY:
Si deseas ordenar los resultados según la longitud de una cadena (por ejemplo, el nombre de los empleados según su longitud), puedes usar la función LENGTH en la cláusula ORDER BY:
SELECT nombre FROM empleados ORDER BY LENGTH(nombre) DESC;
- ◇ Explicación: Esta consulta ordena los nombres de los empleados de mayor a menor longitud (por el valor calculado con la función LENGTH).

Ejemplo: Función TRIM en la cláusula ORDER BY:
Si deseas ordenar los resultados eliminando los espacios en blanco al principio y al final de las cadenas, puedes usar la función TRIM en la cláusula ORDER BY:
SELECT direccion FROM clientes ORDER BY TRIM(direccion);
- ◇ Explicación: TRIM(direccion) elimina los espacios en blanco antes de ordenar las direcciones de los clientes.

Consideraciones sobre el uso de funciones en WHERE y ORDER BY:
WHERE: Las funciones en WHERE se usan principalmente para aplicar condiciones específicas sobre los valores de las columnas.
Se pueden usar para transformar los datos antes de compararlos (por ejemplo, UPPER, LOWER, LENGTH). Ten en cuenta que el uso de funciones puede afectar el rendimiento si estás trabajando con grandes volúmenes de datos, ya que las funciones se aplican a cada fila de la tabla.

ORDER BY: Las funciones en ORDER BY permiten ordenar los resultados de manera más flexible. Por ejemplo, puedes ordenar según la longitud de las cadenas, sin importar las mayúsculas o minúsculas, etc.
Al igual que con WHERE, el uso de funciones en ORDER BY puede afectar el rendimiento si la consulta tiene que procesar grandes volúmenes de datos.

FUNCIONES ANIDADAS
Las funciones anidadas en SQL se refieren a la práctica de usar una función dentro de otra. Esto te permite realizar operaciones más complejas y realizar transformaciones adicionales en los resultados sin necesidad de escribir múltiples consultas. Las funciones anidadas pueden ser muy útiles cuando deseas aplicar múltiples transformaciones o cálculos a una misma columna de manera concisa.

Sintaxis general de las funciones anidadas: funcion1(funcion2(cadena o valor))
- funcion1 es la función externa.
- funcion2 es la función interna, que se ejecuta primero.

Las funciones se ejecutan de adentro hacia afuera, es decir, primero se evalúa la función interna, y luego la función externa.

Ejemplos de funciones anidadas:
1. Usando UPPER y LENGTH para contar el número de caracteres de una cadena en mayúsculas:
SELECT LENGTH(UPPER(nombre)) FROM empleados;
- ◇ Explicación: La función interna UPPER(nombre) convierte el valor de la columna nombre a

mayúsculas. La función externa LENGTH luego calcula la longitud de la cadena resultante. Este resultado nos dará el número de caracteres de cada nombre, pero primero transformado a mayúsculas.

2. Usando TRIM y LOWER para eliminar espacios y convertir a minúsculas:
SELECT LOWER(TRIM(direccion)) FROM clientes;

◇ Explicación: La función interna TRIM(direccion) elimina los espacios en blanco al principio y al final de la cadena de texto en la columna direccion. La función externa LOWER convierte el resultado de TRIM a minúsculas. El resultado será una dirección sin espacios extra y en minúsculas.

CONCATENACION
La concatenación en SQL se refiere a la acción de unir dos o más cadenas de texto en una sola. Oracle SQL proporciona diversas maneras de concatenar cadenas, y es una de las operaciones más comunes al trabajar con datos de texto.

Métodos para Concatenar Cadenas en Oracle:
1. Usando el operador ||
El operador || (doble barra vertical) es la forma estándar en Oracle SQL para concatenar cadenas de texto. Puedes usarlo para unir columnas, valores literales, o incluso una mezcla de ambos.

Sintaxis: SELECT columna1 || columna2 || ' texto adicional' AS resultado FROM nombre_tabla;
Ejemplo: Concatenar nombre y apellido:
SELECT nombre || ' ' || apellido AS nombre_completo FROM empleados;

◇ Explicación: En este caso, estamos concatenando la columna nombre con un espacio ' ' y luego con la columna apellido, resultando en una nueva columna nombre_completo.

Ejemplo: Concatenar texto con valor numérico
SELECT 'El salario de ' || nombre || ' es ' || salario || ' USD' AS mensaje FROM empleados;

◇ Explicación: Este ejemplo crea una cadena que concatena el texto 'El salario de ', seguido del nombre del empleado, el texto ' es ', el valor de la columna salario y ' USD'.

2. Usando la función CONCAT
Oracle también proporciona la función CONCAT para concatenar dos cadenas. Sin embargo, hay una limitación: CONCAT solo puede unir dos cadenas a la vez, por lo que si necesitas concatenar más de dos, deberás usarla varias veces.
Sintaxis: CONCAT(cadena1, cadena2)

Ejemplo:
SELECT CONCAT(nombre, apellido) AS nombre_completo FROM empleados;

◇ Explicación: Este ejemplo concatena el nombre y el apellido de los empleados sin espacio entre ellos. Si deseas agregar un espacio, tendrías que usar otra concatenación:
SELECT CONCAT(CONCAT(nombre, ' '), apellido) AS nombre_completo FROM empleados;

3. Concatenación con múltiples columnas y literales
Puedes combinar columnas con literales de texto y otros valores. Por ejemplo, si quieres generar una dirección completa concatenando una calle, ciudad y código postal:
SELECT direccion || ', ' || ciudad || ', ' || codigo_postal AS direccion_completa FROM clientes;

◇ Explicación: Aquí concatenamos la direccion, ciudad y codigo_postal para crear una dirección completa.

FUNCION REPLACE
La función REPLACE en SQL se utiliza para reemplazar una subcadena dentro de una cadena de texto por otra subcadena. Esta función es muy útil cuando deseas hacer cambios en los datos, como reemplazar ciertos caracteres o palabras dentro de una columna o cadena.

Sintaxis básica de la función REPLACE: REPLACE(cadena_original, subcadena_a_reemplazar, subcadena_reemplazo)

- cadena_original: La cadena de texto en la que deseas realizar la sustitución.

- subcadena_a_reemplazar: La subcadena que quieres reemplazar dentro de la cadena_original.
- subcadena_reemplazo: La subcadena con la que deseas reemplazar la subcadena_a_reemplazar.

Ejemplos de la función REPLACE:
1. Reemplazar una palabra en una cadena:
Imagina que tenemos una tabla productos con una columna descripcion, y queremos reemplazar la palabra 'rojo' por 'azul'.
SELECT REPLACE(descripcion, 'rojo', 'azul') AS descripcion_modificada FROM productos;
◇ Explicación: La función REPLACE busca todas las apariciones de la palabra 'rojo' en la columna descripcion y las reemplaza por 'azul'.

2. Reemplazar un carácter en una cadena:
Supongamos que tenemos una tabla clientes y queremos reemplazar todos los guiones - en la columna telefono por espacios.
SELECT REPLACE(telefono, '-', ' ') AS telefono_modificado FROM clientes;
◇ Explicación: Aquí, REPLACE busca los guiones - en la columna telefono y los reemplaza por un espacio ' '.

3. Eliminar una subcadena:
Si deseas eliminar una parte de la cadena, puedes usar REPLACE para reemplazar la subcadena con una cadena vacía ''.
SELECT REPLACE(nombre, 'Dr. ', '') AS nombre_sin_prefijo FROM empleados;
◇ Explicación:En este caso, estamos eliminando el prefijo 'Dr. ' de los nombres de los empleados. Al reemplazar 'Dr. ' por una cadena vacía '', se eliminan todas las apariciones de este prefijo.

4. Reemplazar caracteres en una cadena numérica:
Supongamos que tenemos una tabla ventas con una columna precio (almacenada como texto) y queremos reemplazar las comas , por puntos . en los precios:
SELECT REPLACE(precio, ',', '.') AS precio_modificado FROM ventas;
◇ Explicación: Aquí, REPLACE reemplaza todas las comas , por puntos ., lo cual es útil cuando trabajamos con valores numéricos representados como texto y necesitamos ajustar el formato.

Consideraciones:
- REPLACE es sensible a mayúsculas y minúsculas: REPLACE distingue entre mayúsculas y minúsculas. Por ejemplo, REPLACE('Hola Mundo', 'mundo', 'amigo') no realizará cambios, ya que la palabra 'mundo' está en minúsculas y la función busca una coincidencia exacta.
- Reemplazo global: La función REPLACE reemplaza todas las apariciones de la subcadena en la cadena original. No tienes que especificar cuántas veces quieres que ocurra el reemplazo, ya que se aplica a cada ocurrencia de la subcadena.
- No cambia la cadena original: La función REPLACE no modifica la columna original de la tabla, sino que devuelve una nueva cadena con los cambios. Si deseas guardar los cambios permanentemente, necesitarás usar una instrucción UPDATE.

FUNCION TRADUCIR
En Oracle SQL, la función TRANSLATE se utiliza para reemplazar caracteres individuales dentro de una cadena de texto. A diferencia de la función REPLACE, que reemplaza una subcadena por otra subcadena completa, TRANSLATE permite reemplazar cada carácter individualmente de acuerdo con las posiciones de las cadenas que se le proporcionan.

Sintaxis de la función TRANSLATE:
TRANSLATE(cadena_original, caracteres_a_reemplazar, caracteres_reemplazo)
- cadena_original: La cadena de texto donde deseas realizar los reemplazos de caracteres.
- caracteres_a_reemplazar: Una cadena de caracteres que se deben buscar y reemplazar en la cadena original.
- caracteres_reemplazo: Una cadena de caracteres que contendrá los nuevos caracteres que reemplazarán a los caracteres de caracteres_a_reemplazar.

Explicación de cómo funciona TRANSLATE: TRANSLATE busca los caracteres en la cadena

caracteres_a_reemplazar y los reemplaza por los caracteres correspondientes en caracteres_reemplazo. Los caracteres de la cadena caracteres_a_reemplazar se reemplazarán por los caracteres en las mismas posiciones de la cadena caracteres_reemplazo. Si no hay un carácter correspondiente en caracteres_reemplazo, el carácter de la cadena original cadena_original no se modifica.

Ejemplos de uso de TRANSLATE:

1. Reemplazar caracteres individuales:
Supongamos que tenemos la siguiente cadena 'ABCD' y queremos reemplazar:
'A' por '1'
'B' por '2'
'C' por '3'
'D' por '4'
SELECT TRANSLATE('ABCD', 'ABCD', '1234') AS resultado FROM DUAL;

- Explicación:

TRANSLATE('ABCD', 'ABCD', '1234') reemplaza: 'A' por '1', 'B' por '2', 'C' por '3', 'D' por '4'.
Resultado: '1234'

. Reemplazar múltiples caracteres de una vez:
Imagina que tenemos una cadena con 'abc123', y queremos reemplazar:
'a' por 'z'
'b' por 'y'
'c' por 'x'
SELECT TRANSLATE('abc123', 'abc', 'zyx') AS resultado FROM DUAL;

- Explicación:

TRANSLATE('abc123', 'abc', 'zyx') reemplaza: 'a' por 'z', 'b' por 'y', 'c' por 'x'
Resultado: 'zyx123'

4. Caso sin reemplazo (solo eliminación):
Si no proporcionamos la cadena caracteres_reemplazo, la función solo eliminará los caracteres especificados en caracteres_a_reemplazar.
SELECT TRANSLATE('abracadabra', 'abc', '') AS resultado FROM DUAL;

- Explicación: TRANSLATE('abracadabra', 'abc', '') elimina los caracteres 'a', 'b' y 'c' de la cadena 'abracadabra'.

Resultado: 'rd'

Consideraciones:

- El orden importa: El orden de los caracteres en las cadenas caracteres_a_reemplazar y caracteres_reemplazo es importante. Los caracteres en caracteres_a_reemplazar se reemplazarán por los correspondientes en la misma posición en caracteres_reemplazo.
- Reemplazo individual: TRANSLATE reemplaza caracteres individuales, no cadenas completas. Si quieres reemplazar una subcadena, deberías usar REPLACE.
- Manejo de valores NULL: Si alguna de las cadenas es NULL, el resultado será NULL.
- Manejo de caracteres sin reemplazo:Si en caracteres_a_reemplazar hay caracteres que no existen en cadena_original, esos caracteres no se modificarán ni se eliminarán.
- Si caracteres_reemplazo tiene menos caracteres que caracteres_a_reemplazar, los caracteres sobrantes en caracteres_a_reemplazar se eliminan.

¿Por qué la consulta devuelve una cadena vacía en Oracle?
Oracle tiene una peculiaridad importante con TRANSLATE: Si la lista de reemplazo está completamente vacía (''), Oracle devuelve NULL en lugar de la cadena modificada. En este caso, cuando TRANSLATE intenta devolver la cadena restante, Oracle lo interpreta como una cadena vacía y la convierte en NULL.

Solución: Usar un espacio en la lista de reemplazo Si quieres eliminar los caracteres 'A', 'B' y 'C', pero asegurarte de que la cadena no sea NULL, usa un espacio en la lista de reemplazo:

SELECT NVL(TRANSLATE('ABCDEFG', 'ABC', ' '), 'SIN DATOS') AS resultado FROM DUAL;

◇ Explicación: 'A', 'B', y 'C' se reemplazan por un espacio ' '.
Sino forzamente se tendrá que elegir porque valores reemplazarlos.

FUNCION SOUNDEX
La función SOUNDEX en Oracle se usa para comparar palabras por su sonido en inglés en lugar de hacerlo por su escritura exacta. Pues genera un código basado en cómo suena una palabra en inglés. Este código se construye según reglas fonéticas del idioma inglés, lo que permite buscar palabras que suenan similar aunque se escriban diferente.
Esto es útil para búsquedas fonéticas, es decir, cuando dos palabras suenan igual pero están escritas de manera diferente.
¿Cómo funciona SOUNDEX?: SOUNDEX convierte una palabra en un código de 4 caracteres, donde:

- El primer carácter es la letra inicial de la palabra.
- Los siguientes 3 caracteres son números que representan los sonidos principales de la palabra.
- Si la palabra tiene menos sonidos, se rellena con ceros (0).

Sintaxis: SELECT SOUNDEX('palabra') FROM DUAL;

Ejemplos básicos
SELECT SOUNDEX('Smith'), SOUNDEX('Smyth') FROM DUAL;
S530 S530
Aunque "Smith" y "Smyth" tienen letras diferentes, suenan igual y SOUNDEX devuelve el mismo código.

SELECT SOUNDEX('Robert'), SOUNDEX('Rupert') FROM DUAL;
R163 R163
"Robert" y "Rupert" suenan similar, por lo que SOUNDEX genera el mismo código.

Comparaciones con SOUNDEX
Podemos usar SOUNDEX en un WHERE para encontrar palabras que suenan igual.
SELECT nombre FROM empleados
WHERE SOUNDEX(nombre) = SOUNDEX('John');
Esto buscará nombres en la base de datos que suenen como "John", como: "John", "Jon", "Jhon", etc.