# UD04: Realización de consultas

RUN SQL - RECUERDA

* CONNECT SYSTEM/14Febrero
* CREATE TABLESPACE ***EJERCICIO***1 DATAFILE ‘C:\ORACLEXE\APP\ORACLE\ORADATA\XE\***EJERCICIO1.DBF***’ SIZE 5M;
* CREATE USER ***EJERCICIO1*** IDENTIFIED BY ***PASS*** DEFAULT TABLESPACE ***EJERCICIO1***;
* GRANT CONNECT, RESOURCE TO ***EJERCICIO1***;

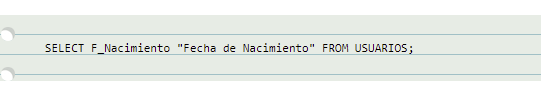
## 2. La sentencia SELECT

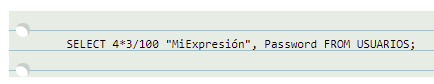
Cuatro partes básicas:

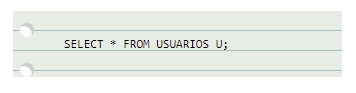
* Cláusula **SELECT** seguida de la descripción de lo que se desea ver, es decir, de los nombres de las columnas que quieres que se muestren separadas por comas simples (" , "). Esta parte es obligatoria.
* Cláusula **FROM** seguida del nombre de las tablas de las que proceden las columnas de arriba, es decir, de donde vas a extraer los datos. Esta parte también es obligatoria.
* Cláusula **WHERE** seguida de un criterio de selección o condición. Esta parte es opcional.
* Cláusula **ORDER BY** seguida por un criterio de ordenación. Esta parte también es opcional.

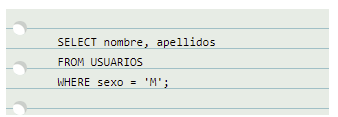
|  |
| --- |
| SELECT [ALL | DISTINCT] columna1, columna2, … FROM tabla1, tabla2, … WHERE condición1, condición2, … ORDER BY ordenación; |

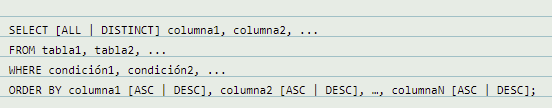
## 2.1 Cláusulas

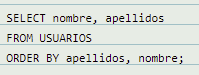


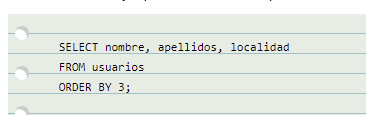


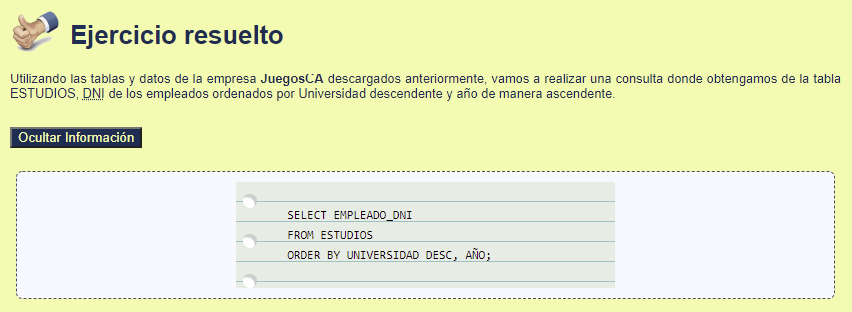






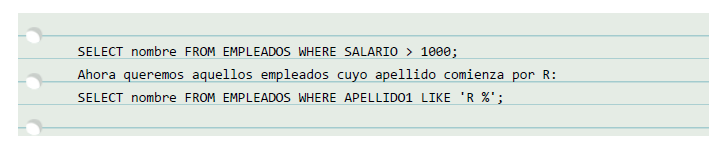






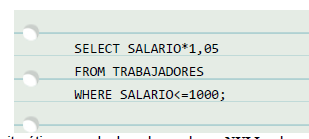
## 3. Operadores

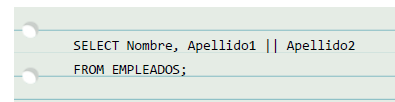
|  |  |
| --- | --- |
| Operador | Significado |
| = | Igualdad. |
| !=, < >, ^= | Desigualdad. |
| < | Menor que |
| > | Mayor que. |
| <= | Menor o igual que. |
| >= | Mayor o igual que. |
| IN | Igual que cualquiera de los miembros entre paréntesis. |
| NOT IN | Distinto que cualquiera de los miembros entre paréntesis. |
| BETWEEN | Entre. Contenido dentro del rango. |
| NOT BETWEEN | Fuera del rango. |
| LIKE '\_abc%' | 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. |

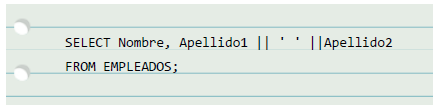


### 3.2. Operadores aritméticos

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

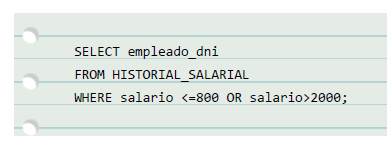


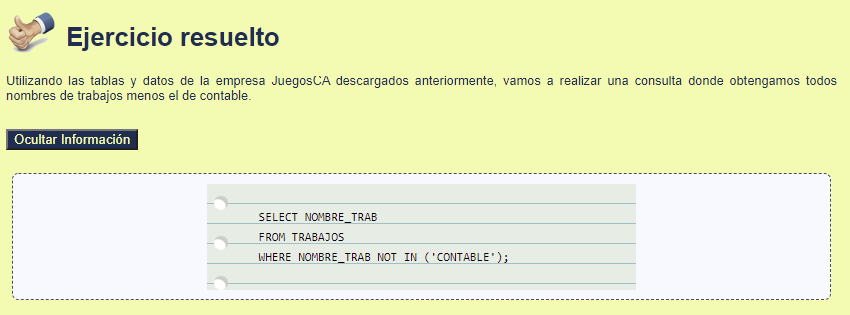




### 3.3. Operadores lógicos

|  |  |
| --- | --- |
| 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. |





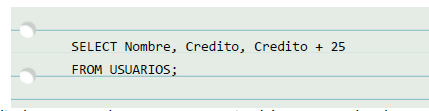
### 3.4. Precedencia

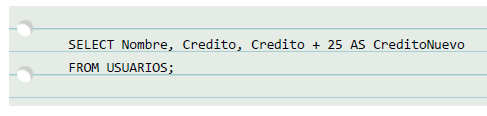
Con frecuencia utilizaremos la sentencia SELECT acompañada de expresiones muy extensas y resultará difícil saber que parte de dicha expresión se evaluará primero, por ello es conveniente conocer el orden de precedencia que tiene Oracle:

1. Se evalúa la multiplicación (\*) y la división (/) al mismo nivel.
2. A continuación sumas (+) y restas (-).
3. Concatenación (| |).
4. Todas las comparaciones (<, >, …).
5. Después evaluaremos los operadores IS NULL, IN NOT NULL, LIKE, BETWEEN.
6. NOT.
7. AND.
8. OR.

Si quisiéramos variar este orden necesitaríamos utilizar paréntesis.

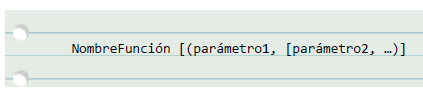
## 4. Consultas calculadas





## 5. Funciones

Las funciones se especifican de la siguiente manera:



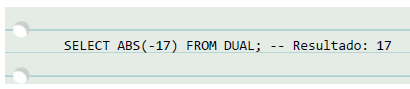
Puedes anidar funciones dentro de funciones.

Existe una gran variedad para cada tipo de datos:

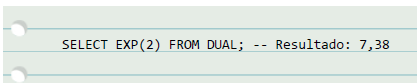
* numéricas,
* de cadena de caracteres,
* de manejo de fechas,
* de conversión,
* otras

### 5.1. Funciones numéricas

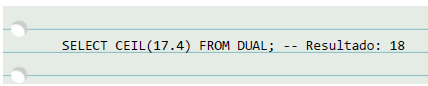
ABS(n): Calcula el valor absoluto de un número n.



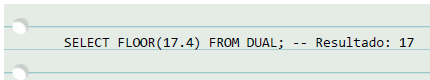
EXP(n): Calcula еn , es decir, el exponente en base e del número n.



CEIL(n): Calcula el valor entero inmediatamente superior o igual al argumento n.



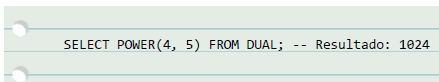
FLOOR(n): Calcula el valor entero inmediatamente inferior o igual al parámetro n.



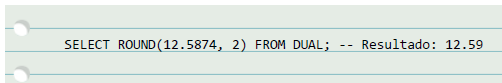
MOD(m,n): Calcula el resto resultante de dividir m entre n.



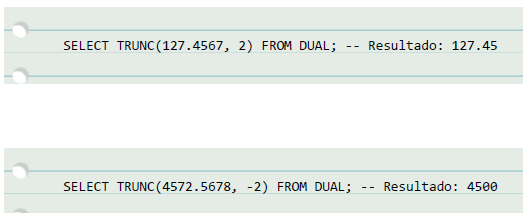
POWER(valor, exponente): Eleva el valor al exponente indicado.

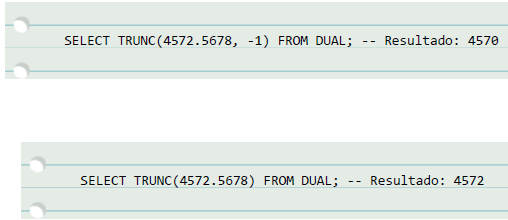


ROUND(n, decimales): Redondea el número n al siguiente número con el número de decimales que se indican.

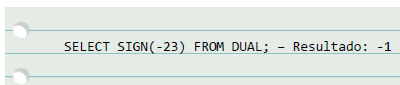


TRUNC(m,n): Trunca un número a la cantidad de decimales especificada por el segundo argumento. Si se omite el segundo argumento, se truncan todos los decimales. Si "n" es negativo, el número es truncado desde la parte entera.



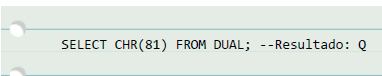


SIGN(n): Si el argumento "n"es un valor positivo, retorna 1, si es negativo, devuelve -1 y 0 si es 0.

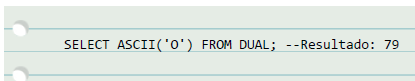


## 5.2. Funcioones de caracteres

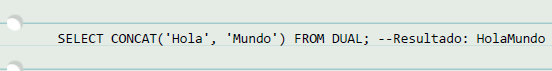
CHR(n): Devuelve el carácter cuyo valor codificado es n.



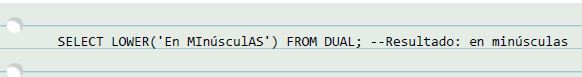
ASCII(n): Devuelve el valor ASCII de n.



CONCAT(cad1, cad2): Devuelve las dos cadenas unidas. Es equivalente al operador ||



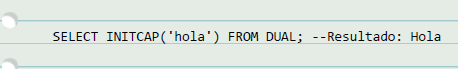
LOWER(cad): Devuelve la cadena cad con todos sus caracteres en minúsculas.



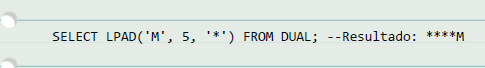
UPPER(cad): Devuelve la cadena cad con todos sus caracteres en mayúsculas.



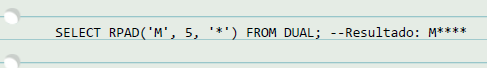
INITCAP(cad): Devuelve la cadena cad con su primer carácter en mayúscula.



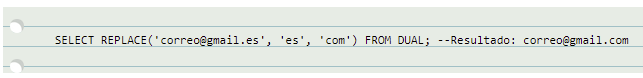
LPAD(cad1, n, cad2): Devuelve cad1 con longitud n, ajustada a la derecha, rellenando por la izquierda con cad2.



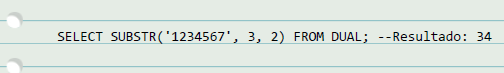
RPAD(cad1, n, cad2): Devuelve cad1 con longitud n, ajustada a la izquierda, rellenando por la derecha con cad2.



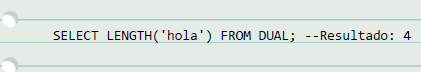
REPLACE(cad, ant, nue): Devuelve cad en la que cada ocurrencia de la cadena ant ha sido sustituida por la cadena nue.



SUBSTR(cad, m, n): Devuelve la cadena cad compuesta por n caracteres a partir de la posición m.



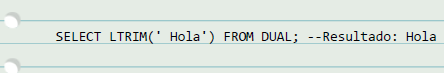
LENGTH(cad): Devuelve la longitud de cad.



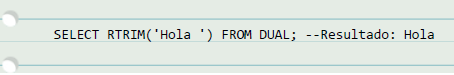
TRIM(cad): Elimina los espacios en blanco a la izquierda y la derecha de cad y los espacios dobles del interior.



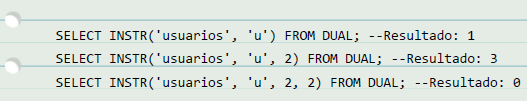
LTRIM(cad): Elimina los espacios a la izquierda que posea cad.



RTRIM(cad): Elimina los espacios a la derecha que posea cad.



INSTR(cad, cadBuscada [, posInicial [, nAparición]]): Obtiene la posición en la que se encuentra la cadena buscada en la cadena inicial cad. 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.



## 5.3. Funciones de manejo de fechas

En los SGBD se utilizan mucho las fechas. Oracle tiene dos tipos de datos para manejar fechas, son DATE y TIMESTAMP.

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

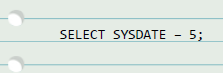
Podemos realizar operaciones numéricas con las fechas:

* Le podemos sumar números y esto se entiende como sumarles días, si ese número tiene decimales se suman días, horas, minutos y segundos.
* La diferencia entre dos fechas también nos dará un número de días.

En Oracle tenemos las siguientes funciones más comunes:

* SYSDATE Devuelve la fecha y hora actuales.
  + Ejemplo: SELECT SYSDATE FROM DUAL;
    - Resultado: 26/07/11
* SYSTIMESTAMP Devuelve la fecha y hora actuales en formato TIMESTAMP.
  + Ejemplo: SELECT SYSTIMESTAMP FROM DUAL;
    - Resultado: 26-JUL-11 08.32.59,609000 PM +02:00
* ADD\_MONTHS(fecha, n) Añade a la fecha el número de meses indicado con n.
  + Ejemplo: SELECT ADD\_MONTHS('27/07/11', 5) FROM DUAL;
    - Resultado: 27/12/11
* MONTHS\_BETWEEN(fecha1, fecha2) Devuelve el número de meses que hay entre fecha1 y fecha2.
  + Ejemplo: SELECT MONTHS\_BETWEEN('12/07/11','12/03/11') FROM DUAL;
    - Resultado: 4
* LAST\_DAY(fecha) Devuelve el último día del mes al que pertenece la fecha. El valor devuelto es tipo DATE.
  + Ejemplo: SELECT LAST\_DAY('27/07/11') FROM DUAL;
    - Resultado: 31/07/11
* NEXT\_DAY(fecha, d) Indica el día que corresponde si añadimos a la fecha el día d. El día devuelto puede ser texto ('Lunes', Martes', ..) o el número del día de la semana (1=lunes, 2=martes, ..) dependiendo de la configuración.
  + Ejemplo: SELECT NEXT\_DAY('31/12/11','LUNES') FROM DUAL; --
    - Resultado: 02/01/12
* EXTRACT(valor FROM fecha) Extrae un valor de una fecha concreta. El valor puede ser day, month, year, hours, etc.
  + Ejemplo: SELECT EXTRACT(MONTH FROM SYSDATE) FROM DUAL;
    - Resultado: 7

En Oracle: Los operadores aritméticos "+" (más) y "-" (menos) pueden emplearse para las fechas. Por ejemplo:



Devolvería la fecha correspondiente a 5 días antes de la fecha actual.

Se pueden emplear estas funciones enviando como argumento el nombre de un campo de tipo fecha.

## 5.4. Funciones de conversión

* TO\_NUMBER(cad, formato) Convierte textos en números. Se suele utilizar para dar un formato concreto a los números. Los formatos que podemos utilizar son los siguientes:

|  |  |
| --- | --- |
| 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. |

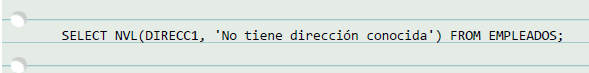
* TO\_CHAR(d, formato) Convierte un número o fecha d a cadena de caracteres, se utiliza normalmente para fechas ya que de número a texto se hace de forma implícita como hemos visto antes.
* TO\_DATE( cad, formato) Convierte textos a fechas. Podemos indicar el formato con el que queremos que aparezca.

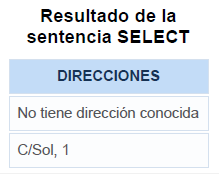
|  |  |
| --- | --- |
| Símbolo | Significado |
| YY | Año en formato de dos cifras |
| YYYY | Año en formato de cuatro cifras |
| MM | Mes en formato de dos cifras |
| MON | Las tres primeras letras del mes |
| MONTH | Nombre completo del mes |
| DY | Día de la semana en tres letras |
| DAY | Día completo 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 |

## 5.5. Otras funciones: NVL y DECODE

Las funciones con nulos nos permitirán hacer algo en caso de que aparezca un valor nulo.

* NVL(valor, expr1) Si valor es NULL, entonces devuelve expr1. Ten en cuenta que expr1 debe ser del mismo tipo que valor.
* DECODE(expr1, cond1, valor1 [, cond2, valor2, ...], default ) Esta función evalúa una expresión expr1, si se cumple la primera condición (cond1) devuelve el valor1, en caso contrario evalúa la siguiente condición y así hasta que una de las condiciones se cumpla. Si no se cumple ninguna condición se devuelve el valor por defecto que hemos llamado default.





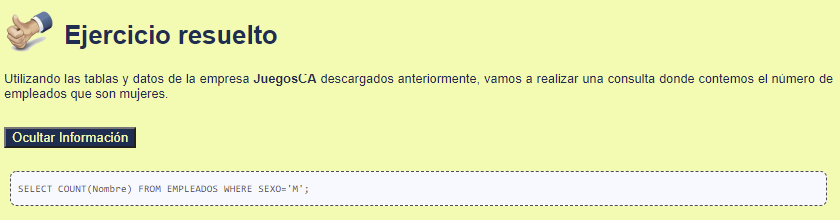
# 6. Consultas de resumen

Todas las funciones de agregado tienen una estructura muy parecida: FUNCIÓN ([ALL| DISTINCT] Expresión) y debemos tener en cuenta que:

* La palabra ALL indica que se tienen que tomar todos los valores de la columna. Es el valor por defecto.
* La palabra DISTINCT indica que se considerarán todas las repeticiones del mismo valor como uno solo (considera valores distintos).
* El grupo de valores sobre el 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. Por tanto, en la expresión nunca puede aparecer ni una función de agregado ni una subconsulta.
* Todas las funciones se aplican a las filas del origen de datos una vez ejecutada la cláusula WHERE (si la tuviéramos).
* Todas las funciones (excepto COUNT) ignoran los valores NULL.
* Podemos encontrar una función de agrupamiento dentro de una lista de selección en cualquier sitio donde pudiera aparecer el nombre de una columna. Es por eso que puede formar parte de una expresión pero no se pueden anidar funciones de este tipo.
* No se pueden mezclar funciones de columna con nombres de columna ordinarios, aunque hay excepciones que veremos más adelante.

## 6.1. Funciones de agregado: SUM y COUNT

* La función SUM:
  + SUM([ALL|DISTINCT] expresión)
  + Devuelve la suma de los valores de la expresión.
  + Sólo puede utilizarse con columnas cuyo tipo de dato sea número. El resultado será del mismo tipo aunque puede tener una precisión mayor.
    - Por ejemplo, SELECT SUM( credito) FROM Usuarios;
* La función COUNT:
  + COUNT([ALL|DISTINCT] expresión)
  + 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 constante o una función.
  + Puede contar cualquier tipo de datos incluido texto.
  + COUNT simplemente cuenta el número de registros sin tener en cuenta qué valores se almacenan.
  + La función COUNT no cuenta los registros que tienen campos NULL a menos que expresión sea el carácter comodín asterisco (\*).
  + Si utilizamos COUNT(\*), calcularemos el total de filas, incluyendo aquellas que contienen valores NULL.
  + Por ejemplo,
    - SELECT COUNT(nombre) FROM Usuarios;
    - SELECT COUNT(\*) FROM Usuarios;

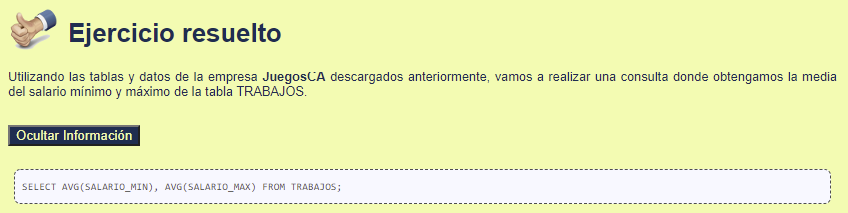


## 6.2. Funciones de agregado: MIN y MAX

* Función MIN:
  + MIN ([ALL| DISTINCT] expresión)
  + Devuelve el valor mínimo de la expresión sin considerar los nulos (NULL).
  + En expresión podemos incluir el nombre de un campo de una tabla, una constante o una función (pero no otras funciones agregadas de SQL).
    - Un ejemplo sería: SELECT MIN(credito) FROM Usuarios;
* Función MAX:
  + MAX ([ALL| DISTINCT] expresión)
  + Devuelve el valor máximo de la expresión sin considerar los nulos (NULL).
  + En expresión podemos incluir el nombre de un campo de una tabla, una constante o una función (pero no otras funciones agregadas de SQL).
    - Un ejemplo, SELECT MAX (credito) FROM Usuarios;

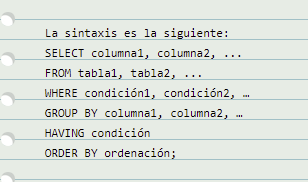
## 6.3. Funciones de agregado AVG, VAR, STDEV y STDEVP

* Función AVG
  + AVG ([ALL| DISTINCT] expresión)
  + Devuelve el promedio de los valores de un grupo, para ello se omiten los valores nulos (NULL).
  + El grupo de 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 y el tipo de dato del resultado puede variar según las necesidades del sistema para representar el valor.
* Función VAR
  + VAR ([ALL| DISTINCT] expresión)
  + Devuelve la varianza estadística de todos los valores de la expresión.
  + Como tipo de dato admite únicamente columnas numéricas. Los valores nulos (NULL) se omiten.
* Función STDEV
  + STDEV ([ALL| DISTINCT] expresión)
  + Devuelve la desviación típica estadística de todos los valores de la expresión.
  + Como tipo de dato admite únicamente columnas numéricas. Los valores nulos (NULL) se omiten.



# 7. Agrupamiento de registros

Podemos obtener estos subtotales utilizando la cláusula GROUP BY.



En la cláusula GROUP BY se colocan las columnas por las que vamos a agrupar. En la cláusula HAVING se especifica la condición que han de cumplir los grupos para que se realice la consulta.

Es muy importante que te fijes bien en el orden en el que se ejecutan las cláusulas:

Caso práctico

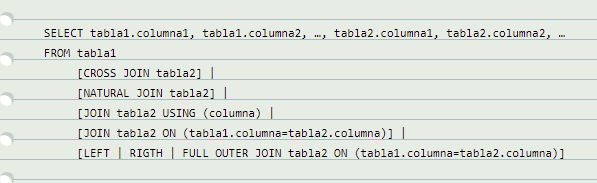
1. WHERE que filtra las filas según las condiciones que pongamos.
2. GROUP BY que crea una tabla de grupos nueva.
3. HAVING filtra los grupos.
4. ORDER BY que ordena o clasifica la salida.

Las columnas que aparecen en el SELECT y que no aparezcan en la cláusula GROUP BY deben tener una función de agrupamiento. Si esto no se hace así producirá un error. Otra opción es poner en la claúsula GROUP BY las mismas columnas que aparecen en SELECT.





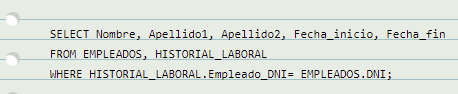
# 8. Consultas multitablas

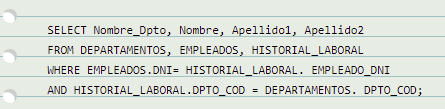


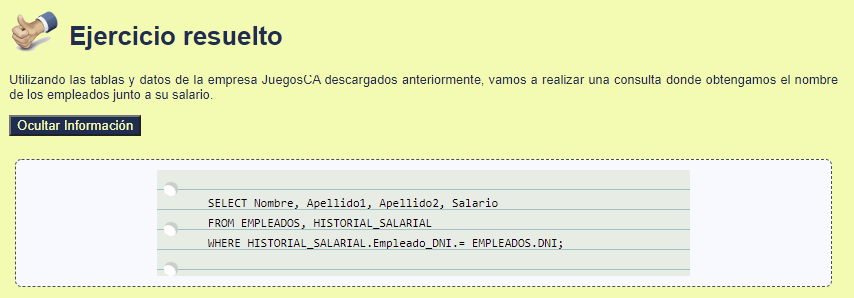
## 8.1.Composiciones internas

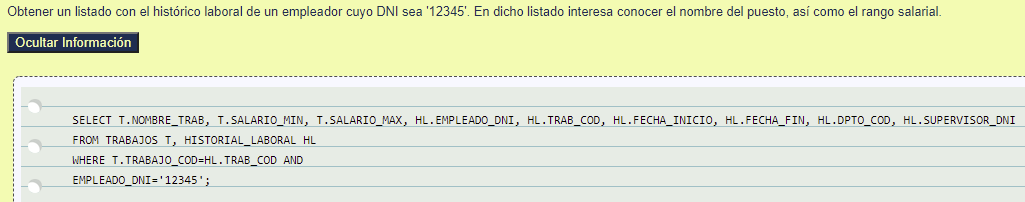
Las reglas para las composiciones son:

* Pueden combinarse tantas tablas como se desee.
* El criterio de combinación puede estar 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 las distintas tablas, deben identificarse especificando la tabla de procedencia o utilizando un alias de tabla.

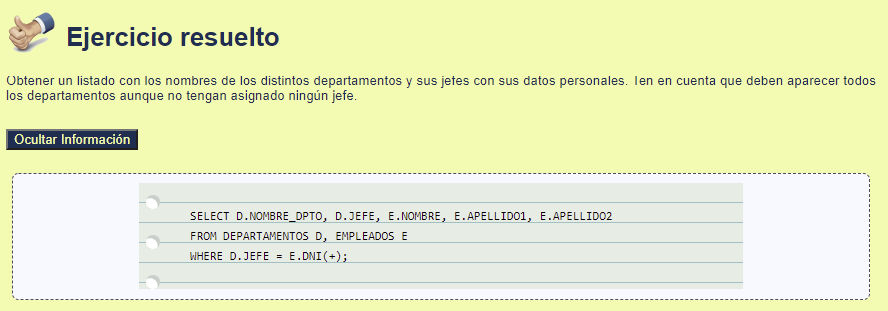




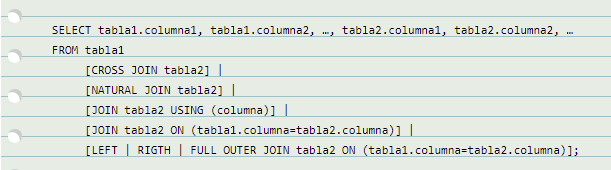




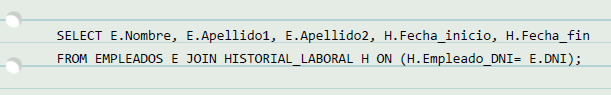
## 8.2. Composiciones externas

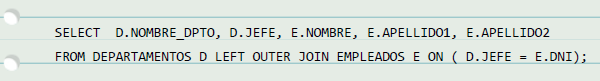


## 8.3. Composiciones en la versión SQL99



* CROSS JOIN: creará un producto cartesiano de las filas de ambas tablas por lo que podemos olvidarnos de la cláusula WHERE.
* NATURAL JOIN: detecta automáticamente las claves de unión, basándose en el nombre de la columna que coincide en ambas tablas. Por supuesto, se requerirá que las columnas de unión tengan el mismo nombre en cada tabla. Además, esta característica funcionará incluso si no están definidas las claves primarias o ajenas.
* JOIN USING: las tablas pueden tener más de un campo para relacionar y no siempre queremos que se relacionen por todos los campos. Esta cláusula permite establecer relaciones indicando qué campo o campos comunes se quieren utilizar para ello.
* JOIN ON: se utiliza para unir tablas en la que los nombres de columna no coinciden en ambas tablas o se necesita establecer asociaciones más complicadas.
* OUTER JOIN: se puede eliminar el uso del signo (+) para composiciones externas utilizando un OUTER
* JOIN, de este modo resultará más fácil de entender.
* LEFT OUTER JOIN: es una 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: es una 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: es una composición externa en la que se devolverán todas las filas de los campos no relacionados de ambas tablas.

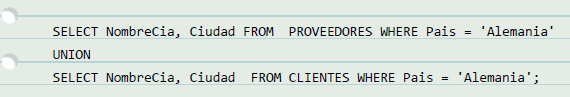




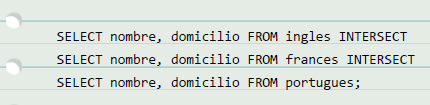
# 9. Otras consultas multitablas: Unión, Intersección y diferencia de consultas

* UNION: combina las filas de un primer SELECT con las filas de otro SELECT, desapareciendo las filas duplicadas.
* INTERSECT: examina las filas de dos SELECT y devolverá aquellas que aparezcan en ambos conjuntos. Las filas 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 reducirán a una antes de comenzar la comparación.

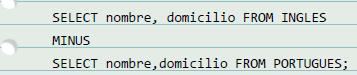
UNIÓN: Obtener los nombres y ciudades de todos los proveedores y clientes de Alemania.



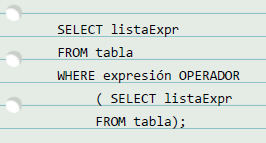
INTERSECCIÓN: Una academia de idiomas da clases de inglés, frances y portugues; almacena los datos de los alumnos en tres tablas distintas una llamada "ingles", en una tabla denominada "frances" y los que aprenden portugues en la tabla "portugues". La academia necesita el nombre y domicilio de todos los alumnos que cursan los tres idiomas para enviarles información sobre los exámenes.



DIFERENCIA: Ahora la academia necesita el nombre y domicilio solo de todos los alumnos que cursan inglés (no quiere a los que ya cursan portugués pues va a enviar publicidad referente al curso de portugués).



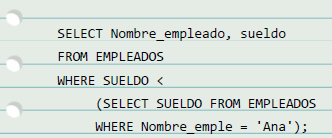
# 10. Subconsultas



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

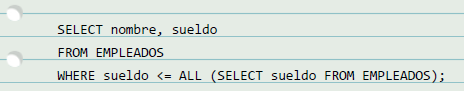
El OPERADOR puede ser >, <, >=, <=, !=, = o IN. Las subconsultas que se utilizan con estos operadores devuelven un único valor, si la subconsulta devolviera más de un valor devolvería un error.

Como puedes ver en la sintaxis, las subconsultas deben ir entre paréntesis y a la derecha del operador.



Cuando el resultado de la subconsulta es más de una fila, SQL utiliza instrucciones especiales entre el operador y la consulta. Estas instrucciones son:

* ANY. Compara con cualquier fila de la consulta. La instrucción es válida si hay un registro en la subconsulta que permite que la comparación sea cierta.
* ALL. Compara con todas las filas de la consulta. La instrucción resultará cierta si es cierta toda la comparación con los registros de la subconsulta.
* IN. No utiliza comparador, lo que hace es comprobar si el valor se encuentra en el resultado de la subconsulta.
* NOT IN. Comprueba si un valor no se encuentra en una subconsulta.



# Tests

1. Si examinas las filas del SELECT de una tabla A y de otra tabla B y obtienes aquellas filas que aparecen en ambas tablas, eliminando duplicados, estás haciendo uso de:

* a.JOIN.
* b.MINUS.
* c.UNION.
* d.INTERSECT.

2. Para obtener datos de dos tablas, Comerciales y Anuncios, la consulta debe ser:

* a.SELECT COMERCIALES FROM ANUNCIOS;.
* b.SELECT \* FROM COMERCIALES, ANUNCIOS WHERE COMERCIALES.DNI = ANUNCIOS.DNI\_COMERCIAL;.
* c.SELECT \* FROM COMERCIALES GROUP BY ANUNCIOS;.
* d.SELECT \* FROM COMERCIALES, ANUNCIOS;.

3. Quiero obtener el siguiente formato para la columna sueldo: 12.012,12€ ¿qué función de las siguientes debería utilizar?

* a.TO\_CHAR(sueldo, '99G999D99N').
* b.TO\_CHAR(sueldo, '99G999D99L').
* c.TO\_NUMBER(sueldo, '999G999D99L').
* d.TO\_CHAR(sueldo, '99D999G99M').

4. HAVING y WHERE:

* a.HAVING valida condiciones de grupos.
* b.WHERE pregunta por características de grupos.
* c.HAVING pregunta por características de filas concretas.
* d.Son cláusulas equivalentes, se usan de la misma forma.

5. ¿Cuáles de estas afirmaciones sobre la sentencia SELECT son verdaderas?

* a.Forma parte de las sentencias de Lenguaje de Definición de Datos.
* b.Podemos obtener cualquier fila, todas las filas, cualquier subconjunto de filas de una tabla.
* c.Se usa para recuperar información.
* d.Se utiliza para hacer consultas a la Base de Datos.

6. ¿Qué es una composición externa?

* a.Se denota con el símbolo (+).
* b.OUTER JOIN.
* c.Seleccionar filas de una tabla aunque no tengan correspondencia con las filas de la otra tabla.
* d.MINUS JOIN.

7. Tenemos la siguiente consulta: SELECT DEPT\_NO FROM DEPART WHERE LOC=('MADRID', 'BARCELONA'); ¿cuál de las siguientes respuestas es verdadera?

* a.Es correcta.
* b.No es correcta debe dejar un espacio entre LOC y el símbolo de igual.
* c.No es correcta pues MADRID y BARCELONA hay que escribirlos con comillas dobles.
* d.No es correcta pues hay que usar el operador 'IN' en lugar del '
* e. '.

8. ¿Qué devuelve la siguiente consulta: SELECT TRUNC(187.98,-2)FROM DUAL;?

* a.100.
* b.188.
* c.187.1.
* d.187.100.

9. ¿Qué palabra clave se utiliza para ordenar el resultado?

* a.ORDER.
* b.SORT.
* c.ORDER BY.
* d.ASC .

10. La sintaxis de las subconsultas es la siguiente: SELECT listaExpr FROM tabla WHERE expresión OPERADOR ( SELECT listaExpr FROM tabla); ¿Qué operadores de los siguientes podemos utilizar como OPERADOR?

* a.OR.
* b.=.
* c.IN.
* d.>.

11. Quiero extraer el último carácter de una cadena, para ello utilizaría:

* a.INSTR('cadena', length('cadena')).
* b.SUBSTR('cadena', length('cadena')).
* c.INSTR('cadena',-1).
* d.SUBSTR('cadena', 1,-1).

12. Una sentencia SELECT ¿cuando altera el contenido de una tabla?

* a.Solo si intervienen cálculos (SUM, AVG).
* b.Solo cuando afecta a todos los registros.
* c.Siempre.
* d.Nunca, SELECT no altera el contenido de una tabla.

13. ¿Qué devolvería la siguiente consulta SELECT NEXT\_DAY('24/11/11', 'JUEVES') FROM DUAL;?

* 2/12/11.
* 24/11/11.
* c.30/11/11.
* 1/12/11.

14. Oracle tiene dos tipos de datos para manejar fechas, son:

* a.TIME, SYSDATE.
* b.DATE, SYSDATE.
* c.DATE, TIMESTAMP.
* d.DATE, TIME.

15. En una sentencia SELECT las cláusulas obligatorias son:

* a.SELECT, FROM, WHERE.
* b.SELECT, FROM, WHERE GROUP BY.
* c.SELECT, FROM, WHERE, ORDER BY.
* d.SELECT, FROM.

16. Las subconsultas sólo aparecen como parte de la condición de búsqueda de una cláusula WHERE, nunca de las cláusulas HAVING, que es inherente a GROUP BY. ¿Verdadero o falso?

* Verdadero
* Falso

17. ¿Para qué sirve la cláusula GROUP BY?

* a.Agrupar las columnas de una tabla.
* b.Clasificar de manera ordenada los campos seleccionados.
* c.Especificar la condición que debe cumplir un grupo de filas.
* d.Organizar registros en grupos y obtener un resumen de dichos grupos.

18. ¿Qué valor devuelve la función: INSTR ('García Pérez, Francisco', 'a',3,2)?

* a.10.
* b.2.
* c.17.
* d.6.

19. En la sintaxis de SELECT, la cláusula ALL:

* a.Recupera todos las columnas de la tabla especificada.
* b.Recupera todas las filas que no estén repetidas.
* c.Recupera todas las filas aunque estén repetidas.
* d.Recupera todas las columnas aunque estén repetidas.

20. Tenemos la siguiente consulta SELECT COUNT(\*) FROM EMPLEADOS GROUP BY DEPARTAMENTOS; ¿Qué resultado obtendríamos?

* a.Número de empleados en cada departamento.
* b.Numero de empleados totales.
* c.Número de filas totales de la tabla.
* d.Número de columnas de la tabla EMPLEADOS agrupados por departamentos.

21. ¿Qué devolvería la siguiente consulta SELECT LAST\_DAY('24/11/11') FROM DUAL;?

* a.Jueves.
* b.30/11/11.
* c.Miércoles.
* d.27/11/05.

22. Quiero rellenar la columna Localidad con puntos ("…") por la derecha, y con una longitud total de 30 caracteres, para ello usaría:

* a.LPAD(Localidad, 30, '.').
* b.RPAD(Localidad, length(localidad)+30, '.').
* c.RPAD(Localidad, length(localidad)+12, '.').
* d.RPAD(Localidad, 30, '.').

23. Si utilizando operadores de comparación de cadenas quiero obtener cualquier cadena cuyo segundo carácter sea una A, usaría:

* a.'\_ \_ A%'.
* b.'%A%'.
* c.'\_A%'.
* d.\_A'.

24. Elige cuáles de las siguientes reglas para las composiciones son verdaderas:

* a.Si hay columnas con el mismo nombre en las distintas tablas, no es necesario identificarlas especificando la tabla de procedencia, ya que deberían tener nombres distintos.
* b.En la cláusula SELECT pueden citarse columnas de ambas tablas, condicionen o no la combinación.
* c.Pueden combinarse tantas tablas como se desee.
* d.El criterio de combinación puede estar formado por más de una pareja de columnas.

25. Con SQL, ¿cómo seleccionarías todos los registros de una tabla denominada "Personas", donde el "Nombre" es "Pedro" y "Apellido" es "Gil"?

* a.SELECT \* WHERE FirstName = 'Pedro', LastName = 'Gil' FROM Personas;.
* b.SELECT \* FROM Personas WHERE Nombre LIKE 'Peter' AND LastName LIKE 'Jackson';.
* c.SELECT \* FROM Personas WHERE Nombre LIKE 'Pedro' AND LIKE 'Gil';.
* d.SELECT \* FROM Personas WHERE FirstName = 'Pedro' OR LastName = 'Gil';.

26. ¿Qué palabra clave se usa para devolver solamente valores diferentes?

* a.COUNT.
* b.AS.
* c.DISTINCT.
* d.ORDER BY.

27. La función MAX(expresión) devuelve el valor máximo de la expresión considerando los nulos. ¿Verdadero o falso?

* Verdadero
* Falso

28. ¿Cuál de estas afirmaciones es correcta sobre la siguiente consulta? SELECT DNOMBRE 'DEPARTAMENTOS', DEPT\_NO 'NUMERO DEPARTAMENTO' FROM DEPART;

* a.Los alias de columnas se especifican delante del nombre de columna.
* b.No están permitidos alias en las columnas.
* c.Es correcta se permiten los alias en las columnas.
* d.Los alias de columnas se especifican con comillas dobles.

29. La función: REPLACE ('QUE COMPLEJO ES SQL', 'SQL', 'sql'). ¿Qué cadena devolvería?

* a.que complejo es SQL.
* b.QUE COMPLEJO ES sQl.
* c.qUE COMPlEJo Es sql.
* d.QUE COMPLEJO ES sql.

30. La combinación externa (OUTER JOIN) nos permite seleccionar algunas filas de una tabla aunque éstas no tengan correspondencia con las filas de la otra tabla con la que se combina. ¿Verdadero o falso?

* Verdadero
* Falso

RESPUESTAS

1. D
2. B
3. C
4. A
5. Bcd
6. Abc
7. d
8. a
9. c
10. bcd
11. b
12. d
13. d
14. c
15. d
16. b
17. d
18. c
19. c
20. a
21. b
22. d
23. c
24. bcd
25. b
26. c
27. b
28. d
29. d
30. a