TEMA 4 SQL - DML

Eduardo Martín-Sonseca

Contenido

[1. INTRODUCCIÓN 2](#_Toc128496592)

[2. MODOS DE EJECUCIÓN SQL 2](#_Toc128496593)

[3. ELEMENTOS DEL LENGUAJE SQL: SENTENCIAS SQL 2](#_Toc128496594)

[3.1. ELEMENTOS DEL LENGUAJE SQL: NORMAS DE ESCRITURA 3](#_Toc128496595)

[4. SENTENCIA SELECT 3](#_Toc128496596)

[4.1. SENTENCIA SELECT FROM 3](#_Toc128496597)

[4.2. SENTENCIA SELECT WHERE 3](#_Toc128496598)

[4.3. SENTENCIA SELECT ORDER BY 4](#_Toc128496599)

[4.4. SENTENCIA SELECT ALL | DISTINCT 4](#_Toc128496600)

[4.5. SENTENCIA SELECT ALIAS DE COLUMNAS 4](#_Toc128496601)

[5. OPERADORES ARITMÉTICOS 4](#_Toc128496602)

[6. LA CONCATENACIÓN DE TEXTOS 4](#_Toc128496603)

[7. CONDICIONES 4](#_Toc128496604)

[7.1. OPERADORES DE COMPARACIÓN 4](#_Toc128496605)

[7.2. OPERADORES LOGICOS 5](#_Toc128496606)

[7.3. BETWEEN 5](#_Toc128496607)

[7.4. IN 5](#_Toc128496608)

[7.5. LIKE 5](#_Toc128496609)

[7.6. NULL 5](#_Toc128496610)

[7.7. Precedencia 6](#_Toc128496611)

[8. Funciones 6](#_Toc128496612)

[9. CONSULTAS MULTITABLA: JOIN 6](#_Toc128496613)

[10. CONSULTAS AGRUPADAS: GROUP BY 6](#_Toc128496614)

[11. OUTER JOIN 7](#_Toc128496615)

[12. COMBINACIONES ESPECIALES 7](#_Toc128496616)

[13. SUBCONSULTAS 7](#_Toc128496617)

[13.1. TEST EN SUBCONSULTAS 8](#_Toc128496618)

[14. SENTENCIA INSERT 8](#_Toc128496619)

# INTRODUCCIÓN

**SQL** es un lenguaje estándar para trabajar en SGBD relacional, es un lenguaje para todo tipo de usuarios, (DML es el lenguaje de usuarios), las características son

* El un **leguaje declarativ**o, lo importante es definir que se desea hacer, no como hacerlo
* Las instrucciones SQL utilizan un lenguaje natural
* Es un lenguaje para todo tipo de usuarios

# MODOS DE EJECUCIÓN SQL

**Ejecución directa. SQL interactivo:** Las instrucciones, se introduce a través de comandos

**Ejecución incrustada o embebida**: Se incluye en el código Java, las instrucciones SQL

**Ejecución a través de clientes gráficos**: Un programa software que permite conectar a la BBDD a través de un cliente, el software permite manejar de forma grafica la BD y las acciones realizadas son traducidas a SQL y enviadas al servidor

**Procesamiento de una instrucción SQL**

1. Se analiza la instrucción, para comprobar su sintaxis.
2. Si es correcta se valora si los metadatos son correctos. Se comprueba esto con la  
   información del diccionario de datos.
3. Si es correcta, se optimiza, a fin de consumir los mínimos recursos posibles.
4. Se ejecuta la sentencia y se muestra el resultado

# ELEMENTOS DEL LENGUAJE SQL: SENTENCIAS SQL

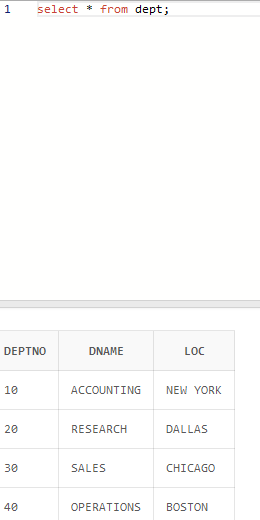
* **DML:** Se utilizan los siguientes comandos
  + **INSERT**
  + **UPDATE**
  + **DELETE**
  + **SELECT.**
* **DDL:** Se utiliza para crear y modificar la estructura de la BD, se utilizan los siguientes comandos:
  + CREATE
  + ALTER
  + DROP
  + RENAME
  + TRUNCATE.
* **DCL** Sentencias que permiten administran los derechos y restricciones de los usuarios de la base de datos, se utilizan los siguientes comandos:
  + **GRANT**
  + **REVOKE**

# ELEMENTOS DEL LENGUAJE SQL: NORMAS DE ESCRITURA

* En SQL no se distingue entre mayúsculas y minúsculas.
* Las instrucciones finalizan con el signo de punto y coma.
* Cualquier comando SQL (SELECT, INSERT,) puede ser partido  
  utilizando espacios o saltos de línea antes de finalizar la instrucción.
* Se pueden tabular líneas para facilitar la lectura si fuera necesario.
* Los comentarios en el código SQL comienzan por /\* y terminan por \*/, para hacer comentarios en una línea es **--**  
  (excepto en algunos SGBD)

# SENTENCIA SELECT

Sirve para recuperar información de la BD



# SENTENCIA SELECT FROM

El alias es cambiar el nombre las columnas de la tabla

# SENTENCIA SELECT WHERE

Esta cláusula permite obtener las filas que cumplen la condición expresada.  
**where job like 'CLERK';**  
Las expresiones pueden ser

* una constante
* una expresión aritmética
* un valor nulo o un nombre de columna.

Los operadores de comparación pueden ser los siguientes:  
Se pueden construir condiciones múltiples utilizando los operadores lógicos AND, OR y NOT  
Se pueden usar paréntesis para forzar el orden de evaluación

# SENTENCIA SELECT ORDER BY

Para ordenar el order by, puede ser ASC o DESC, y si no se pone nada, es de forma ASC

# SENTENCIA SELECT ALL | DISTINCT

* **ALL** muestra todos incluso los duplicados, Si no se pone nada, es **ALL**
* **DISTICT** Elimina duplicados, teniendo en cuenta de la tabla
  + **select distinct emp.job from scott.emp;**

# SENTENCIA SELECT ALIAS DE COLUMNAS

Los nombres de las columnas se pueden cambiar en la misma sentencia select creando alias (se pone entre comillas dobles a continuación del nombre de la columna)

* + **SELECT nom\_dpto “Departamento”**

# OPERADORES ARITMÉTICOS

La prioridad de los operadores aritméticos es la normal, tienen más prioridad la multiplicación y división, después la suma y la resta, se puede modificar la prioridad de una consulta utilizando paréntesis.

# LA CONCATENACIÓN DE TEXTOS

Para la concatenación de textos se utiliza **||**

# CONDICIONES

Se utiliza la cláusula where para poner una condición en la consulta que deben cumplir todas las filas, si no lo cumplen, no aparece en el resultado

# OPERADORES DE COMPARACIÓN

Se utilizan para comparar números

|  |  |
| --- | --- |
| Operador | Significado |
| **>** | **Mayor que** |
| **<** | **Menor que** |
| **>=** | **Mayor o igual** |
| **<=** | **Menor o igual** |
| **=** | **Igual** |
| **<>** | **Distinto** |
| **!=** | **Distinto** |

# OPERADORES LOGICOS

|  |  |
| --- | --- |
| Operador | Significado |
| **AND** | **Devuelve true si las dos condiciones son verdaderas** |
| **OR** | **Devuelve true si una las dos condiciones es verdadera** |
| **NOT** | **Si era verdadera pasa a ser falsa, y viceversa** |

# BETWEEN

Obtiene datos que se encuentran dentro de un **rango** su formato es:

**[NOT] BETWEEN valor inicial AND valor\_final**

# IN

Comprueba si una expresión **pertenece** o no a una lista de valores, su formato es

**[NOT] IN (lista de valores separados por comas)**

# LIKE

Sirve para comparar cadenas de caracteres

|  |  |
| --- | --- |
| Operador | Significado |
| **%** | **Representa cualquier cadena de 0 o más caracteres** |
| **\_** | **Representa cualquier carácter** |

# NULL

Es null, si está completamente vacía la fila para comprobarlo se utiliza los siguiente:

**columna IS [NOT] NULL**

# Precedencia

|  |  |
| --- | --- |
| Operador | Significado |
| **1** | **\* /** |
| **2** | **+ -** |
| **3** | **||** |
| **4** | **Comparaciones** |
| **5** | **IS [NOT] NULL, [NOT] LIKE, IN** |
| **6** | **NOT** |
| **7** | **AND** |
| **8** | **OR** |

# Funciones

Estas funciones se usan dentro de expresiones y actúan con los valores de las columnas, variables o constantes.

Todas las funciones devuelven un resultado que procede de un determinado cálculo.

La mayoría de las funciones precisan que se les envíe datos de entrada (parámetros o argumentos) que son necesarios para realizar el cálculo de la función.

Oracle tiene una tabla DUAL, que se permite hacer pruebas, tiene un solo campo **DUMMY**

<https://gregoriofer.com/moodle/pluginfile.php/70966/mod_resource/content/3/Funciones%20ORACLE.pdf>

# CONSULTAS MULTITABLA: JOIN

En el apartado FROM se pueden indicar varias tablas separadas por comas.

Si no ponemos la condición de emparejamiento (Join) entre las tablas, se produce un producto cruzado o producto cartesiano, en el que aparecerán todos los registros de tabla1 relacionados con todos los registros de tabla2, es decir, el producto cartesiano empareja todas las filas de una tabla con cada fila de la otra tabla., la sintaxis es:

**SELECT columnas de las tablas que aparecen en la cláusula FROM tabla1, tabla2 WHERE tabla1.columna = tabla2.columna;**

# CONSULTAS AGRUPADAS: GROUP BY

La sentencia SELECT posibilita agrupar uno o más conjuntos de filas. El agrupamiento se lleva a cabo mediante la cláusula GROUP BY por las columnas especificadas y en el orden especificado, las **clausulas se ejecutan en tiempo de ejecución :**

|  |  |
| --- | --- |
| **1 WHERE** | **Selecciona las filas** |
| **2 GROUP BY** | **agrupa las filas** |
| **3 HAVING** | **Filtra grupos** |
| **4 ORDER BY** | **Ordena los grupos** |

# OUTER JOIN

Este nos permite seleccionar filas de una tabla aunque no tengan correspondencia con las filas de la otra tabla que se combina, su sintaxis es :

**SELECT tabla1.columa1, tabla1.columa2, tabla2.columa1**

**FROM tabla1, tabla2**

**WHERE tabla1.columa1 = tabla2. columna1 (+)**

# COMBINACIONES ESPECIALES

Los operadores relacionales **UNION, INTERSECT y MINUS** son operadores de conjuntos, son filas resultantes de cualquier select, que permiten combinar los resultados de varias select, para obtener un único resultado, el formato es:

**SELECT … FROM … WHERE … Operador de conjuntos SELECT … FROM … WHERE …**

* **Operador unión combina** los resultados de ambas consultas, y unión, no elimina duplicados
* **Operador intersect devuelve** las filas que son **comunes** en ambas consultas, las filas duplicadas son eliminadas
* **Operador minus devuelve** las filas que están en la **primera** select y no en la segunda

# SUBCONSULTAS

Una subconsulta es una sentencia SELECT dentro de otra, consiste en incluir una SELECT **como parte de la clausula where,** la subconsulta se ejecuta primero, y el valor extraído de la subconsulta es “introducido” en la consulta principal, este es el formato

**SELECT ...  
FROM ...  
WHERE columna operador (SELECT ...  
FROM ...  
WHERE ...)**

## TEST EN SUBCONSULTAS

* **Test de comparación** **(>,< <>, <=, >=,=),** compara el valor de una expresión con un valor único producido por una subconsulta

**Obtener aquellos apellidos de empleados cuyo oficio es  
igual al oficio de 'GIL':  
SELECT apellido FROM Empleados WHERE oficio =  
(SELECT oficio FROM Empleados WHERE apellido ='GIL' );**

* **Test de pertenencia a un conjunto devuelto por una subconsulta** (**IN**), comprueba si el valor de una expresión coincide con uno del conjunto de valores producidos por una subconsulta

**Obtener aquellos apellidos de empleados cuyo oficio sea  
alguno de los oficios que hay en el departamento 20:  
SELECT apellido FROM Empleados WHERE oficio IN  
(SELECT oficio FROM Empleados WHERE num\_dpto=20);**

* **Test de existencia** (**EXISTS**, **NOT EXISTS**) examina si una subconsulta produce alguna fila de resultados, si devuelve filas es TRUE si no FALSE, (Las consultas que no tengan empleados se usa el operador **NOT EXISTS**)

**Listar los departamentos que tengan empleados:  
SELECT nombre\_dpto, num\_dpto FROM Departamentos WHERE EXISTS  
(SELECT \* FROM Empleados WHERE  
Empleados.num\_dpto=Departamentos.num\_dpto);**

* **Test de comparación cuantificada (ANY ALL)**
  + **ANY** compara el valor de una expresión con cada uno del conjunto de valores
  + **ALL** compara el valor de una expresión con cada uno del conjunto de valores

# SENTENCIA INSERT

Esta sentencia permite añadir filas de datos en una tabla su sintaxis es

**INSERT INTO tabla [(columna1 [, columna2] …)] VALUES (valor1 [, valor2] …);**

- tabla: es la tabla en la que se van a insertar las filas.

- [(columna1 [, columna2] …)]: son las columnas donde se van a introducir los valores. Si no se especifican, se consideran, por defecto, todas las columnas de la tabla.

- (valor1 [, valor2] …): son los valores que se van a dar a las columnas. Éstos se deben corresponder con cada una de las columnas que aparecen. Además, deben coincidir con el tipo de dato definido para cada columna.

- Cualquier columna que no se encuentre en la lista de columnas recibirá el valor NULL, siempre y cuando no esté definida como NOT NULL, en cuyo caso INSERT fallará.

- Si no se da la lista de columnas, se han de introducir valores en todas las columnas.

# SENTENCIA INSERT MULTIFILA

La sentencia ins