TEMA 4. LENGUAJE DE CONSULTA ESTRUCTURADO (SQL).

1. QUÉ ES SQL
2. CÓMO SE USA SQL
3. PARA QUÉ SIRVE SQL
4. TIPOS DE SENTENCIAS
    - 4.1. LENGUAJE DE MANIPULACIÓN DE DATOS (LMD)
    - 4.2. LENGUAJE DE DEFINICIÓN DE DATOS (LDD)

1. QUÉ ES SQL.

- SQL (Structured Query Language) es un lenguaje que permite expresar operaciones diversas sobre datos almacenados en bases de datos relacionales.
- SQL es un lenguaje de cuarta generación.
- Fue desarrollado por IBM y la versión original se denominaba SEQUEL.
- ANSI-SQL es el nombre que damos a una estandarización de las diversas implementaciones que de la evolución del lenguaje original inventado por IBM se han ido desarrollando posteriormente.

2. CÓMO SE USA.

Las peticiones sobre los datos se expresan en SQL mediante *sentencias*, que deben escribirse de acuerdo con las reglas sintácticas y semánticas de este lenguaje.

- **SQL INTERACTIVO**: Las sentencias pueden escribirse directamente en la pantalla de un terminal interactivo, en el cual también se recibe el resultado.
- **SQL INCORPORADO Y DINÁMICO:** Esto quiere decir que se pueden incorporar instrucciones de SQL en lenguajes de programación como: C++, C, Java, Cobol, Pascal y Fortran.

3. PARA QUÉ SIRVE

Permite:

1. Principalmente a los Programadores y a los Usuarios:
    - Insertar, modificar, borrar y consultar los datos contenidos en las tablas de la Base de Datos. (Lenguaje de Manipulación de Datos: LMD).
2. Realizar tareas propias del Administrador de la Base de Datos, como son:
    - Definición, modificación y destrucción de objetos de la Base de Datos, gestión de las autorizaciones de acceso, etc. (Lenguaje de Definición de Datos: LDD y Lenguaje de Control de Datos: LCD).

4. TIPOS DE SENTENCIAS.

4.1 LENGUAJE DE MANIPULACIÓN DE DATOS (LMD)

Las sentencias de esta parte de SQL permiten realizar consultas y mantenimiento de datos. Son las siguientes:

SELECT, INSERT, UPDATE y DELETE

Veamos la sintaxis de estas sentencias:
(Ver "Convenciones de sintaxis" en enlace Manual de Referencia de Transact-SQL)

SENTENCIA SELECT

Permite realizar consultas sobre una o varias tablas de la base de datos. La sintaxis completa de la instrucción SELECT es compleja, aunque las cláusulas principales se pueden resumir del modo siguiente:

```
SELECT [ ALL | DISTINCT ]
[ TOP (número | porcentaje de las filas) [ PERCENT ] ] expresión
[,expresiones,…]
[FROM tabla1 [, tabla2,...] [JOIN…]]
[WHERE predicados ]
[GROUP BY columna/s]
[HAVING condición]
[ORDER BY columna/s] [DESC];
```

Si tenemos una sentencia SELECT formada por las cláusulas anteriores, el resultado de ejecutarla se puede obtener mediante los pasos siguientes:

1. Ejecutar la cláusula FROM. Lo obtenido de momento es la tabla resultante de la sentencia.
2. Ejecutar la cláusula WHERE. Se eliminan todas las filas que no satisfagan el predicado.
3. Ejecutar la cláusula GROUP BY. Formar grupos con las filas de la tabla resultante en el paso anterior que tengan iguales valores en las columnas de agrupamiento.
4. Ejecutar la cláusula HAVING. Descartar los grupos que no satisfagan la condición especificada.
5. Ejecutar la cláusula SELECT. Esto implica evaluar sus expresiones para cada grupo, produciendo por cada uno de ellos una fila de la tabla resultante final, con tantos valores como expresiones. Si se ha especificado DISTINCT se eliminan las filas repetidas.
6. Ejecutar ORDER BY. Es decir presentar la tabla resultante clasificada por las columnas especificadas. Por defecto ascendentemente.
7. Si se ha limitado el número de filas, ejecutar TOP.

Consideraciones:

1. Una **expresión** es una combinación de operandos, operadores aritméticos y paréntesis (no todos ellos obligatorios) que cuando el SGBD la ejecuta produce un único resultado. Los operandos pueden ser nombres de columnas, constantes, *funciones* u otras expresiones. Como expresión también podemos usar *.

2. Las funciones pueden ser colectivas o escalares.

- Son ***funciones colectivas o de agregado***: SUM, MAX, MIN, AVG, COUNT. El resultado se obtiene a partir de una colección de valores. Representan un único valor que se obtiene aplicando una *operación a un conjunto de valores.* Un resultado por tabla o por grupo.
- *Son* ***funciones*** **escalares**: LEN, SUBSTRING, UPPER, LOWER. El resultado se obtiene a partir de un único valor (un resultado por fila). Al menos debéis saber localizar las de cadenas, matemáticas y las de fecha-hora.

(Ver "Operadores" en enlace Manual de Referencia de Transact-SQL: Elementos de lenguaje/Operadores/Visión general)

(Ver "Funciones" en el enlace Manual SQL w3schools: SQL Server Functions, y en enlace Manual de Referencia de Transact-SQL: Funciones)

Los siguientes ejemplos de tipos de expresiones se encuentran en "Ejemplos de tipos de expresiones.SQL". El archivo pertenece a EJEMPLOS_LMD.

```
--EJEMPLOS DE TIPOS DE EXPRESIONES QUE NOS PODEMOS ENCONTRAR:

USE empresa;

--SELECT expresión. Donde expresión es un operando (función).
SELECT GETDATE();

/*SELECT expresión. Donde expresión es un operando (función),
que tiene como parámetro
otra función */
SELECT FORMAT(GETDATE(),'d');

--SELECT * y usando la cláusula FROM.
SELECT *
FROM temple;

/*SELECT expresión1, expresión2 con cláusula FROM y cláusula
ORDER BY.
Donde expresión1 y expresión2 son operandos (nombres de
columnas)*/
SELECT  nomem, salar
FROM temple
ORDER BY nomem ASC;

--Igual si no ponemos ASC (valor por defecto).
SELECT  nomem, salar
FROM temple
ORDER BY nomem;

--Igual, pero ordenando descendentemente.
SELECT  nomem, salar
FROM temple
ORDER BY nomem DESC;

/*SELECT expresión1, expresión2 con cláusula FROM y cláusula TOP
para limitar el número de filas mostradas*/
--Del resultado de la consulta muestra las cinco primeras filas.
SELECT TOP(5) nomem, salar
FROM temple;
```

```sql
/*Dos ejemplos de SELECT con TOP y ORDER BY*/
--Del resultado de la consulta muestra las cinco primeras filas.
/*ORDER BY 1: 1 indica el lugar que ocupa la expresión del
SELECT por la que ordeno.*/
SELECT TOP(5) nomem, salar
FROM temple
ORDER BY 1;

--Del resultado de la consulta muestra el 50%.
SELECT TOP(50) PERCENT nomem, salar
FROM temple
ORDER BY 1;

--SELECT con la cláusula DISTINCT y ORDER BY
/*Así mostramos los salarios con las comisiones de los 14
empleados.*/
--ALL es el valor por defecto. Podemos no ponerlo.
SELECT ALL salar, comis
FROM temple
ORDER BY 1;
--Si solo queremos conocer los distintos salarios con sus
comisiones, me interesa que no se repitan las filas.
SELECT DISTINCT salar,comis
FROM temple
ORDER BY 1;

/*SELECT expresión1, expresión2, expresión3 con cláusula FROM.
Donde expresión1 y expresión2 son operandos (nombres de
columnas) y expresión3 está formada por operando (nombre de
columna), operador aritmético (producto) y operando (constante
numérica 2).*/
/*También usamos un alias de columna para darle nombre a una
columna.*/
SELECT nomem, salar, (salar * 2) AS 'DOBLE DEL SALARIO'
FROM temple;
```

```
/*SELECT expresión1, expresión2, expresión3 con cláusula FROM.
Donde expresión1 y expresión2 son operandos (nombres de
columnas) y expresión3 está formada por operando (nombre de
columna), operador aritmético (producto) y otra expresión
(10/100). A su vez, esta última expresión está formada
por operando (constante numérica 10), operador aritmético
(división) y operando (constante numérica 100).*/
/*También usamos un alias de columna para darle nombre a una
columna.*/
SELECT nomem, salar, (salar * 10/100) AS '10% DEL SALARIO'
FROM temple;

--FUNCIONES ESCALARES

/*SELECT expresión. Donde expresión es un operando (función
escalar).*/
/*También usamos un alias de columna para darle nombre a una
columna.*/
SELECT nomem, LEN(nomem) AS 'LONGITUD'
FROM temple;

SELECT fecna, YEAR(GETDATE()) - YEAR(fecna) AS 'AÑOS'
FROM temple;

--FUNCIONES COLECTIVAS O DE AGREGADO

/*SELECT expresión. Donde expresión es un operando (función de
agregado).*/
/*También usamos un alias de columna para darle nombre a una
columna.*/
SELECT AVG(salar) AS 'SALARIO MEDIO'
FROM temple;

--FUNCIÓN DE AGREGADO COUNT
/*
COUNT(nombre_columna) cuenta el número de veces que
el atributo tiene valor distinto de null en la tabla

COUNT(*) cuenta el número de filas de la tabla
*/
SELECT COUNT(extel)
FROM temple;

SELECT COUNT(DISTINCT extel)
FROM temple;
```

```sql
SELECT COUNT(*)
FROM temple;

SELECT COUNT(comis)
FROM temple;

SELECT COUNT(DISTINCT comis)
FROM temple;

SELECT COUNT(*)
FROM temple;
```

3. Cuando en la **cláusula FROM** se especifica una sola tabla se buscan los valores en ella. Cuando se pone más de una tabla separadas por coma se realiza el producto cartesiano de estas y sobre ese resultado se buscan los datos. También puede aparecer la **cláusula JOIN**, en ese caso se realiza una concatenación de las tablas sobre el campo que se le indique.

Los siguientes ejemplos de la cláusula FROM se encuentran en “Ejemplos FROM1.SQL”. El archivo pertenece a EJEMPLOS_LMD.

```
/*En estos ejemplos vemos que en la cláusula FROM puede aparecer:
- Una sola tabla.
- Varias tablas separadas por "coma". Entonces, tendremos el producto
cartesiano de dichas tablas.
- Varias tablas usando la cláusula JOIN. Entonces, tendremos la concatenación
  de dichas tablas mediante el campo indicado.
*/
USE ejemploempresa;

SELECT * FROM departamento;
SELECT * FROM empleado;

--PRODUCTO CARTESIANO.
SELECT *
FROM departamento, empleado;
SELECT *
FROM empleado, departamento;

--PRODUCTO CARTESIANO Y SELECCIÓN DE FILAS.
--Uso de alias de tablas.
SELECT *
FROM departamento d, empleado e
WHERE d.numde=e.numde;

SELECT e.nomem,d.nomde
FROM departamento d, empleado e
WHERE d.numde=e.numde;

SELECT *
FROM empleado e, departamento d
WHERE e.numde=d.numde;

SELECT e.nomem,d.nomde
FROM empleado e, departamento d
WHERE e.numde=d.numde;

--CONCATENACIÓN: USO DE LA CLÁUSULA JOIN.
SELECT *
FROM departamento d JOIN empleado e ON (d.numde=e.numde);

SELECT *
FROM empleado e JOIN departamento d ON (e.numde=d.numde);
```

```sql
SELECT e.nomem,d.nomde
FROM departamento d JOIN empleado e ON (d.numde=e.numde);

SELECT e.nomem,d.nomde
FROM empleado e JOIN departamento d ON (e.numde=d.numde);
```

4. Un **predicado** expresa una condición entre valores, y según estos pueden resultar Verdadero o Falso. Los predicados pueden ser simples o compuestos.

PREDICADOS SIMPLES

Predicados Básicos

Expresan condiciones de comparación entre dos valores. Tienen esta forma:

"ELEMENTO DE COMPARACIÓN_1" Operador_Relacional "ELEMENTO DE COMPARCIÓN_2"

Operador_Relacional (operadores de comparación) puede ser: =, <>, >, <, >=, <=

Los "ELEMENTOS DE COMPACIÓN" pueden ser expresiones. El "ELEMENTO DE COMPARCIÓN_2" puede ser a su vez el resultado de una sentencia SELECT, que **debe ir entre paréntesis** y producir un único valor. A las sentencias SELECT incluidas dentro de otra sentencia SELECT se le llaman ***sentencias subordinadas o subselects***.

Los siguientes ejemplos de predicados básicos se encuentran en "Ejemplos predicados básicos.SQL". El archivo pertenece a EJEMPLOS_LMD.

```
USE ejemploempresa;

/*Ejemplo de predicado básico donde los dos elementos de
comparación son expresiones.
numde es operando (nombre de columna) y 2 es operando (constante
numérica)*/
SELECT nomem
FROM empleado
WHERE numde > 2;

/*Ejemplo de predicado básico donde el segundo elemento de
comparación es un SELECT: Queremos saber el nombre de los
empleados que están en un departamento con un código mayor que
el código del departamento de Dep2*/
SELECT * FROM empleado;
SELECT numde FROM departamento WHERE nomde='Dep2';

--SOLUCIÓN:
SELECT nomem
FROM empleado
WHERE numde > (SELECT numde FROM departamento WHERE
nomde='Dep2');
```

```sql
--También se podría haber hecho con producto cartesiano.
SELECT *
FROM empleado e,departamento d;

--SOLUCIÓN:
SELECT nomem
FROM empleado e,departamento d
WHERE d.nomde='Dep2' AND e.numde>d.numde;

--También se podría haber hecho con concatenación o JOIN.
SELECT *
FROM empleado e JOIN departamento d ON (e.numde>d.numde);

--SOLUCIÓN:
SELECT nomem
FROM empleado e JOIN departamento d ON (e.numde>d.numde)
WHERE d.nomde='Dep2';
```

Predicado NULL

Su formato es: nombre_columna IS [NOT] NULL

Los siguientes ejemplos de predicados NULL se encuentran en “Ejemplos predicado NULL.SQL”. El archivo pertenece a EJEMPLOS_LMD.

```
USE empresa;

SELECT * FROM temple;

/*Obtiene los nombres de los empleados cuya comisión es nula.
El campo comis tiene el valor NULL.
*/
SELECT nomem
FROM temple
WHERE comis IS NULL;

/*Obtiene los nombres de los empleados cuya comisión no es nula.
El campo comis tiene un valor distinto de NULL.
*/
SELECT nomem
FROM temple
WHERE comis IS NOT NULL;
```

Predicados Cuantificados

Cuando se usa una sentencia subordinada en un predicado de comparación, su resultado debe ser un valor único, sin embargo se admite que el resultado tenga varios valores si la sentencia subordinada va precedida de algunas de las palabras cuantificadoras: ANY o ALL.

Si se usa ANY . El predicado cuantificado es verdadero si la comparación es verdadera para alguno de los valores resultantes en la sentencia subordinada.

Los siguientes ejemplos de predicados cuantificados (ANY) se encuentran en "predicado ANY.SQL". El archivo pertenece a EJEMPLOS_LMD.

```
USE EMPRESA

/*Obtener el número de los empleados (su identificador único)
que ganen más que alguno de los empleados del departamento 110*/

--Vemos todos los empleados.
SELECT * FROM temple;

--Vemos los datos que nos interesa de los empleados del
departamento 110.
SELECT numem, salar, numde
FROM temple
WHERE numde=110;

--Estos son los empleados cuyo salario es mayor que 1500 o mayor
que 1800.
SELECT *
FROM temple
WHERE salar > ANY(SELECT salar
                  FROM temple
                  WHERE numde=110);

/*Si no queremos que salgan los empleados del departamento 110
vamos a poner un predicado compuesto utilizando el operador
lógico AND*/
SELECT *
FROM temple
WHERE (salar > ANY(SELECT salar
               FROM temple
               WHERE numde=110) ) AND numde <> 110;
```

```sql
--SOLUCIÓN:
SELECT numem
FROM temple
WHERE (salar > ANY(SELECT salar
                FROM temple
                WHERE numde=110) ) AND numde <> 110;




--OTRAS FORMAS DE HACER EL EJEMPLO:

/*Si el salario es mayor que el más pequeño, entonces lo será
para los dos*/
SELECT numem
FROM temple
WHERE (salar > (SELECT MIN (salar) FROM temple WHERE numde=110)
) AND numde <> 110;

/*SELF JOIN: Se llama SELF JOIN cuando hacemos un producto
cartesiano o una concatenación de una tabla consigo misma.*/

--Con producto cartesiano.
SELECT E1.numem ,E1.numde,E1.salar,E2.numem,E2.numde,E2.salar
FROM temple E1, temple E2
WHERE (E1.salar>E2.salar) AND E2.numde=110 AND E1.numde<>110;

/*Tenemos que poner DISTINCT para que no salgan dos veces los
empleados
como el 150 cuyo salario es 2200 y cumple que es mayor que
1500 y mayor que 1800*/
SELECT DISTINCT E1.numem
FROM temple E1, temple E2
WHERE (E1.salar>E2.salar) AND E2.numde=110 AND E1.numde<>110;

--Con JOIN
SELECT DISTINCT E1.numem
FROM temple E1 JOIN temple E2 ON (E1.salar>E2.salar)
WHERE  E2.numde=110 AND E1.numde<>110;
```

Si se usa ALL . El predicado cuantificado es verdadero si la comparación es verdadera para todos y cada uno de los valores resultantes en la sentencia subordinada.

Los siguientes ejemplos de predicados cuantificados (ALL) se encuentran en "predicado ALL.SQL". El archivo pertenece a EJEMPLOS_LMD.

```
USE EMPRESA

/*Obtener el número de los empleados (su identificador único)
que ganen más que cualquiera de los empleados del departamento
110*/

--Vemos todos los empleados.
SELECT * FROM temple;

/*Vemos los datos que nos interesa de los empleados del
departamento 110.*/
SELECT numem, salar, numde
FROM temple
WHERE numde=110;

/*Estos son los empleados cuyo salario es mayor que 1500 y mayor
que 1800.*/
SELECT *
FROM temple
WHERE salar > ALL (SELECT salar
                        FROM temple
                    WHERE numde=110);

/*Si no queremos que salgan los empleados del departamento 110
vamos a poner un predicado compuesto utilizando el operador
lógico AND*/
SELECT *
FROM temple
WHERE (salar > ALL (SELECT salar
                        FROM temple
                    WHERE numde=110) ) AND numde <> 110;

--SOLUCIÓN:
SELECT numem
FROM temple
WHERE (salar > ALL (SELECT salar
                        FROM temple
                    WHERE numde=110) ) AND numde <> 110;
```

```sql
--OTRAS FORMAS DE HACER EL EJEMPLO:

/*Si es mayor que el más grande, entonces lo será para todos*/
SELECT numem
FROM temple
WHERE salar > (SELECT MAX(salar) FROM temple WHERE numde=110)
AND numde <> 110;

/*SELF JOIN:*/
--Producto cartesiano.
SELECT E1.numem,E1.salar,E2.numem,E2.salar, E2.numde
FROM temple E1, temple E2
WHERE  E2.numde=110 AND E1.numde<>110 AND (E1.salar> (SELECT
MAX(salar) FROM temple WHERE numde=110) );

/*Eliminamos las repeticiones*/
SELECT DISTINCT E1.numem
FROM temple E1, temple E2
WHERE  E2.numde=110 AND E1.numde<>110 AND (E1.salar> (SELECT
MAX(salar) FROM temple WHERE numde=110) );
```

Predicado BETWEEN

Su formato es: Expresion1 [NOT] BETWEEN Expresion2 AND Expresion3

Sirve para determinar si un valor está comprendido entre otros dos.

Los siguientes ejemplos de predicados BETWEEN se encuentran en “predicado BETWEEN.SQL”. El archivo pertenece a EJEMPLOS_LMD.

```
USE EMPRESA

/*Obtener los empleados cuyo salario estén comprendido entre
1000 y 1600 ambos inclusive.*/
--Lo que nos pide es que: salario >= 1000 AND salario <= 1600.

--Vemos todos los empleados.
SELECT * FROM temple;

SELECT *
FROM temple
WHERE salar BETWEEN 1000 AND 1600;



--Si no usamos el predicado BETWEEN, se podría hacer así.
SELECT *
FROM temple
WHERE salar >= 1000 AND salar<=1600;

/*Obtener los empleados cuyo salario sea menor que 1000 o bien
mayor que 1600. */
--Lo que nos pide es que: salario < 1000 OR salario > 1600.
--Al ser lo contrario de BETWEEN, podemos usar el predicado NOT
BETWEEN.

SELECT *
FROM temple
WHERE salar NOT BETWEEN 1000 AND 1600;

--Si no usamos el predicado BETWEEN, se podría hacer así.
SELECT *
FROM temple
WHERE salar < 1000 OR salar> 1600;
```

Predicado LIKE

Su formato es: Nombre_columna [NOT] LIKE constante alfanumérica.

Sirve para buscar combinaciones de caracteres, se pueden usar comodines.

(Ver SQL Like en el enlace Manual SQL w3schools, y ver "LIKE" en enlace Manual de Referencia de Transact-SQL)

Carácter comodín	Descripción	Ejemplo
`%`	Cualquier cadena de cero o más caracteres.	`WHERE title LIKE '%computer%'` busca todos los títulos de libros que tengan la palabra `computer` en cualquier parte del título.
`_` (subrayado)	Cualquier carácter individual.	`WHERE au_fname LIKE '_ean'` busca todos los nombres de cuatro letras que finalicen con `ean` (`Dean`, `Sean` y así sucesivamente).
`[ ]`	Cualquier carácter individual que se encuentre en el intervalo `[a-f]` o el conjunto `[abcdef]` que se haya especificado.	`WHERE au_lname LIKE '[C-P]arsen'` busca apellidos de autores que terminen por `arsen` y que empiecen por cualquier carácter individual entre `C` y `P`, por ejemplo, `Carsen`, `Larsen`, `Karsen` y así sucesivamente. En las búsquedas de intervalos, los caracteres incluidos en el intervalo pueden variar, dependiendo de las reglas de ordenación de la intercalación.
`[^]`	Cualquier carácter individual que no se encuentre en el intervalo `[^a-f]` o el conjunto `[^abcdef]` que se haya especificado.	`WHERE au_lname LIKE 'de[^l]%'` busca todos los apellidos de autores que empiecen por `de` y en los que la letra siguiente no sea `l`.

Los siguientes ejemplos de predicados LIKE se encuentran en “predicado LIKE.SQL”. El archivo pertenece a EJEMPLOS_LMD.

```
USE EMPRESA

/*Obtener el número del departamento (su identificador único)
Sector Servicios*/
--Vemos todos los empleados.
SELECT * FROM tdepto;

--SOLUCIÓN: Usando un predicado básico.
SELECT numde
FROM tdepto
WHERE nomde = 'Sector Servicios';

/*Obtener el número de los departamentos (su identificador
único) cuyo nombre de departamento empiece por la palabra
Sector*/
SELECT numde
FROM tdepto
WHERE nomde LIKE 'Sector %';
```

Predicado IN

Su formato es:

Expresión [NOT] IN (constante1, constante2, ...)

Sirve para preguntar si el resultado de una expresión está incluido en esa lista. En vez de una lista podemos poner una sentencia SELECT subordinada. Tendrá esta forma:

Expresión [NOT] IN (Subselect) equivale a Expresión = ANY (Subselect)

Los siguientes ejemplos de predicados IN se encuentran en "predicado IN.SQL". El archivo pertenece a EJEMPLOS_LMD.

```
USE EMPRESA

/*Obtener los datos de los empleados que ganen igual que
alguno de los empleados del departamento 111*/

--Vemos todos los empleados.
SELECT * FROM temple;

/*Vemos los datos que nos interesa de los empleados del
departamento 111.*/
SELECT numem,numde,salar
FROM temple
WHERE numde=111
ORDER BY 2;

--SOLUCIÓN:
--Si conocemos la información de la lista, la podemos poner.
SELECT *
FROM temple
WHERE salar IN (2200,1800) AND numde <> 111;

--SOLUCIÓN:
/*Si no conocemos lo que ganan los empleados del departamento
111, tendremos que usar IN con un subselect*/
SELECT *
FROM temple
WHERE (salar IN (SELECT salar
                 FROM temple
                 WHERE numde=111) ) AND numde <> 111;
```

```sql
/*El resultado es equivalente a utilizar un predicado ANY
con el operador relacional = .*/
SELECT *
FROM temple
WHERE (salar = ANY(SELECT salar
                   FROM temple
                   WHERE numde=111) ) AND numde <> 111;

--OTRAS FORMAS DE HACER EL EJEMPLO:

/*SELF JOIN:*/
--Producto cartesiano.
SELECT E1.*
FROM temple E1, temple E2
WHERE E2.numde=111 AND E1.numde<>111 AND E2.salar=E1.salar;

--JOIN.
SELECT E1.*
FROM temple E1 JOIN temple E2 ON (E2.salar=E1.salar)
WHERE E2.numde=111 AND E1.numde<>111;
```

Predicado EXISTS

Su formato es: [NOT] EXISTS (Subselect)

Es verdadero si el resultado de la sentencia subordinada contiene una o más filas.

Los siguientes ejemplos de predicados EXISTS se encuentran en “predicado EXISTS.SQL”. El archivo pertenece a EJEMPLOS_LMD.

```
USE EMPRESA

/*
   EXISTS     = TRUE  si el subselect devuelve 1 o más filas
   EXISTS     = FALSE si el subselect devuelve 0 filas

   NOT EXISTS = TRUE  si el subselect devuelve 0 filas
   NOT EXISTS = FALSE si el subselect devuelve 1 o más filas
*/

/*Si existe algún departamento, entonces decir cuántos empleados
tiene la empresa.*/
SELECT COUNT(*)
FROM temple
WHERE EXISTS (SELECT * FROM tdepto);




/*Si todos los directores son en propiedad, entonces decir
cuántos empleados tiene la empresa, o lo que es lo mismo, si no
existe ningún director en funciones, entonces decir cuántos
empleados tiene la empresa*/
SELECT COUNT(*)
FROM temple
WHERE  NOT EXISTS (SELECT * FROM tdepto WHERE tidir='F');
```

```sql
--OTRA FORMA DE HACER EL EJEMPLO:
/*Usamos la cláusula CASE. (Ver SQL Case en el enlace Manual SQL
w3schools)*/

SELECT CASE WHEN (SELECT COUNT(*) FROM tdepto) =
                 (SELECT COUNT(*) FROM tdepto WHERE tidir='P')
            THEN COUNT(*)
            ELSE 0
            END

FROM temple;

/*Obtener el nombre de los centros que tienen al menos un
departamentos con presupuestos inferior a 3000 euros*/

--Vemos todos los departamentos y centros.
SELECT * FROM tdepto;
SELECT * FROM tcentr;

SELECT nomce
FROM tcentr C
WHERE (EXISTS (SELECT * FROM tdepto  D WHERE D.numce=C.numce AND
presu<3000));

--OTRAS FORMAS DE HACER EL EJEMPLO:

--Con predicado ANY.
SELECT nomce
FROM tcentr C
WHERE numce = ANY (SELECT numce FROM tdepto  WHERE presu<3000);

--Con predicado IN.
SELECT nomce
FROM tcentr C
WHERE numce IN (SELECT numce FROM tdepto  WHERE presu<3000);
```

```sql
--Con JOIN.
/*En este caso hay que poner la cláusula DISTINCT, porque en el
caso de que hubiera más de un departamento en el mismo centro
que cumpliera el WHERE, el nombre del centro se repetiría.
Prueba con presu<=3000
*/
SELECT DISTINCT nomce
FROM tcentr c JOIN tdepto D ON (C.numce=d.numce)
WHERE presu<3000;


/* Obtener el nombre de los departamentos que no tienen
empleados con sueldos inferiores a 1500 euros*/

--Vemos todos los departamentos y empleados.
SELECT * FROM tdepto;
SELECT * FROM temple ORDER BY numde;

SELECT nomde
FROM tdepto D
WHERE (NOT EXISTS (SELECT * FROM temple E WHERE D.numde=E.numde
AND salar<1500));

--OTRAS FORMAS DE HACER EL EJEMPLO:

--Con predicado ALL
SELECT nomde
FROM tdepto
WHERE numde <> ALL(SELECT numde FROM temple  WHERE salar<1500);

--Con Predicado IN
SELECT nomde
FROM tdepto
WHERE numde NOT IN (SELECT numde FROM temple  WHERE salar<1500);
```

PREDICADOS COMPUESTOS

Son combinaciones de otros predicados simples o compuestos, con los operadores lógicos AND, OR o NOT.

5. La **cláusula GROUP BY** indica que se han de agrupar filas de la tabla, de modo que todas las que tengan iguales valores en las columnas de agrupamiento formen un grupo.

6. La **cláusula HAVING** sirve para descartar grupos de filas.

Los siguientes ejemplos de la cláusula GROUP BY se encuentran en Ejemplos GROUP BY y HAVING.SQL". El archivo pertenece a EJEMPLOS_LMD.

```
USE EMPRESA

/*CUANDO USAMOS LA CLÁUSULA GROUP BY, LA EXPRESIÓN QUE VA JUNTO
AL SELECT SOLO ADMITE COMO EXPRESIÓN EL/LOS NOMBRE/S
DE COLUMNA/S UTILIZADA/S PARA AGRUPAR Y FUNCIONES
DE AGREGADO.*/

/*EN LA CLÁUSULA HAVING NOS ENCONTRAMOS NORMALMENTE:

FUNCIÓN DE AGREGADO OP RELACIONAL EXPRESIÓN.

RECUERDA QUE EXPRESIÓN PUEDE SER UN SUBSELECT.
Y SI EL SUBSELECT PRODUCE MÁS DE UNA FILA, PUEDES
PONER LAS CLÁUSULAS ALL O ANY. TAMBIÉN PODEMOS TENER
UN PREDICADO PREDICADO IN*/

/*Listar para cada departamento, su número (numde) y el máximo
salario que se gana dentro de él.*/

--Vemos los datos que nos interesa de los empleados.
SELECT numem,numde,salar
FROM temple
ORDER BY 2;

SELECT numde, MAX(salar) AS 'Máximo salario del departamento'
FROM temple
GROUP BY numde;
```

```
/*Igual, pero indicar también cuántos empleados hay en cada
departamento. La consulta es solo para los departamentos con
código superior a 100*/

SELECT numde, COUNT(*) AS 'Número de empleados del dpto.',
       MAX(salar) AS 'Máximo salario del dpto.'
FROM temple
WHERE numde > 100
GROUP BY numde;

--Listar para cada salario cuántos empleados los ganan.
SELECT salar, COUNT(*) AS
      'Número de empleados con el mismo salario'
FROM temple
GROUP BY salar;

/*Listar para los departamentos con más de dos empleados,
su número (numde) y el máximo salario.*/
SELECT numde, MAX(salar) AS 'Máximo salario del departamento'
FROM temple
GROUP BY numde
HAVING COUNT(*)>2
ORDER BY 1;

--Usamos todas las cláusulas:

/*Listar para cada departamento distinto del 112, su número
(numde) y el máximo salario que se gana dentro de él.
Teniendo en cuenta que el departamento debe tener más
de dos empleados. */
SELECT numde, MAX(salar)
FROM temple
WHERE numde <> 112
GROUP BY numde
HAVING COUNT(*)>2
ORDER BY 1;
```

```
--Ejemplo con Subselect en HAVING:

/*Igual que el anterior, pero teniendo en cuenta que el
departamento debe tener más empleados que los que
tiene el departamento 110 */
SELECT numde, MAX(salar),COUNT(*)
FROM temple
WHERE numde <> 112
GROUP BY numde
HAVING COUNT(*)> (SELECT COUNT(*) FROM temple WHERE numde=110)
ORDER BY 1;


--TAMBIÉN PODEMOS AGRUPAR POR MÁS DE UNA COLUMNA.

/*Obtener en cada departamento cuántas personas hay con el mismo
número de hijos*/

--Vemos los datos que nos interesa de los empleados.
SELECT numem, numde,numhi
FROM temple
ORDER BY numde,numhi;

/*Aquí estamos agrupando por número de departamento y dentro de
cada grupo (cada departamento), por número de hijos
(igual número de hijos) */

SELECT numde,numhi,COUNT(*) AS 'número de elementos del grupo'
FROM temple
GROUP BY numde,numhi
ORDER BY 1;
```