TEMA 4.2.- CONSULTA DE LOS DATOS

Para recuperar información o lo que es lo mismo, para realizar consultas a la base de datos, utilizaremos una única sentencia SELECT. Con esta sentencia, que tiene distintos niveles de complejidad como se irá viendo más adelante, se especifica qué es lo que se quiere obtener, y no dónde ni cómo.

De la consulta se puede obtener: cualquier unidad de datos, todos los datos, cualquier subconjunto de datos, cualquier conjunto de subconjuntos de datos.

A. SENTENCIA SELECT

```
SELECT [ALL | DISTINCT]
 [expre_colum1, expre_colum2,...,expre_column | *]
FROM [nombre_tabla1, nombre_tabla2,...,nombre_tablan]
[WHERE condición]
[ORDER BY expre_colum [ASC | DESC] [, expre_colum [ASC | DESC]
      ...]];
```

expre_colum: puede ser una columna de una table, una constant, una expresión aritmética, una función o varias funciones anidadas.

B. CLÁUSULAS DE SELECT

- SELECT: Especifica los campos, constantes y expresiones que se mostrarán en el resultado de la consulta.
- ALL: De forma predeterminada, se muestran todas las filas del resultado de la consulta.
- DISTINCT: Excluye duplicados de cualquier fila del resultado de la consulta.

  Nota: Puede utilizar DISTINCT únicamente una vez por cada cláusula SELECT.
- FROM: especifica la tabla o lista de tablas de las que se recuperan los datos. Por ejemplo, consultamos los nombre de los alumnos y su nota en la tabla ALUMNOS
- WHERE: obtiene las filas que cumplen la condición expresada. La complejidad de la expresión es prácticamente ilimitada. La expresión puede ser una

constante, una expresión aritmética, un valor nulo o un nombre de una columna. Los operadores de comparación se pueden ver en la siguiente tabla:

=, >, <, >=, <=, !=, <>
IN, NOT IN; BETWEEN, NOT BETWEEN
LIKE

Se pueden construir condiciones múltiples usando los operadores lógicos booleanos estándares: AND, OR y NOT. Se permite usar paréntesis para forzar el orden. Ejemplos:

```
WHERE nota=5;
WHERE (nota >=5) AND (curso=1);
WHERE (nota IS NULL) OR (UPPER(nom_alum)='PEDRO');
```

- ORDER BY: Especifica el criterio de clasificación del resultado de la columna. ASC (por defecto) para ordenación ascendente, DESC, descendente. Ejemplo:

```
SELECT * FROM alumnos ORDER BY nota; (ordena por nota)
SELECT * FROM alumnos ORDER BY nom_alum, curso DESC (por nombre ascendente y curso descendente)
SELECT dept_no, dnombre, loc FROM departamento ORDEN BY 2; (el número indica que se ordene por el segundo atributo del SELECT)
```

C. SELECCIÓN DE COLUMNAS.

El formato SELECT que nos permite seleccionar las columnas de una tabla es:

```
SELECT [ALL | DISTINCT]
 [expre_colum1, expre_colum2,...,expre_column | *]
FROM [nombre_tabla1, nombre_tabla2,...,nombre_tablan];
```

Trabajaremos con el ejemplo, para lo cual debemos crear las siguientes tablas:

EMPLE		
emp_no	NUMBER(4)	PK
Apellido	VARCHAR2(10)	
Oficio	VARCHAR2(10)	
Dir	NUMBER(4)	
fecha_alt	DATE	
Salario	NUMBER(7)	
Comisión	NUMBER(7)	
dept_no	NUMBER(2)	FK

DEPART		
dept_no	NUMBER(2)	PK
dnombre	VARCHAR2(14)	
loc	VARCHAR2(14)	

Para recuperar los datos tenemos dos formas:

1. Ponemos los nombres de todas la columnas, uno tras otro, separados por comas:

```
SELECT emp_no, apellido, oficio, dir, fecha_alt,salario,comisión, dept_no
FROM emple;
```

2. Ponemos *, que representa todas las columnas de la tabla:

```
SELECT * FROM emple;
```

Para seleccionar determinadas columnas solo ponemos el nombre de aquellas columnas que deseemos:

```
SELECT  dnombre, dept_no
FROM depart;
```

D. SELECCIÓN POR FILAS

Para seleccionar determinadas filas necesitamos usar la cláusula WHERE

Ejercicio 1 relación complementaria

E. ALIAS

Cuando se consultan las bases de datos, los nombres de las columnas se usan como cabeceras de representación. Si el nombre resulta demasiado larg, corto o críptico, existe la oportunidad de cambiarlo con la misma sentencia SQL de consulta creando ALIAS. Ese alias se pone entre comillas dobles, a la derecha de la columna.

Ejercicio 2 relación complementaria