Clases SQL

EnriqueBDL

Índice:

Crear Tabla:

```
-- En SQL puedes hacer un comentario de esta manera.

-- Ahora vamos a ver cómo crear una tabla.

CREATE TABLE MERCEDES (

    ID_MODELO NUMBER(5) NOT NULL,

-- Esta línea añade la columna "ID_MODELO" que solo contiene valores tipo "NUMBER" con hasta 5 dígitos. Esta columna no puede estar vacía (NULL).

    N_MODELO VARCHAR2(30),

     -- Esta línea añade la columna "N_MODELO" que solo contiene una cadena de texto con hasta 30 caracteres.

    PRECIO NUMBER(8,2)

 -- Esta línea añade la columna "PRECIO" que contiene un valor numérico con hasta 8 dígitos en total, incluyendo 2 decimales.
);
```

```
CREATE TABLE EMPLEADOS (

    ID_EMPLEADOS INT PRIMARY KEY,

--Esta línea añade la columna que almacena datos tipo "int". "PRIMARY KEY" define la columna "ID_EMPLEADOS" como clave primaria de la tabla.

    N_EMPLEADOS VARCHAR(100),

    SALARIO DECIMAL(10, 2),

--Esta línea añade la columna que almacena datos tipo "float".

    FECHA_DE_NACIMIENTO DATE

 -- Esta línea añade la columna que almacena datos tipo fecha "YYYY-MM-DD".

);
```

-- DEFAULT se utiliza para especificar un valor por defecto para una columna en una tabla.

-- Esto significa que si no se proporciona un valor para esa columna cuando se inserta una nueva fila, se utilizará el valor por defecto especificado.

```
CREATE TABLE barcos (
   id_barco NUMBER PRIMARY KEY,
   nombre VARCHAR2(100) NOT NULL,
   tipo VARCHAR2(50) DEFAULT 'Desconocido',
   longitud NUMBER(10, 2) DEFAULT 30.0,
   fecha_construccion DATE DEFAULT SYSDATE
);
```

-- UNIQUE se utiliza para garantizar que los valores en una columna (o una combinación de columnas) sean únicos en una tabla.

-- Esto significa que no puede haber dos filas con el mismo valor en la columna (o combinación de columnas).

```
CREATE TABLE USUARIOS (
   ID_USUARIO INT PRIMARY KEY,
   NOMBRE VARCHAR(100),
   EMAIL VARCHAR(100) UNIQUE
);
```

-- CHECK se utiliza para definir una condición que debe cumplir el valor de una columna en una tabla.

```
CREATE TABLE TRABAJADORES (
   ID_TRABAJADOR INT PRIMARY KEY,
   NOMBRE VARCHAR(100),
   salario DECIMAL(10, 2) CHECK (salario > 0)
);
```

-- FOREIGN KEY se utiliza para establecer una relación entre dos tablas en una base de datos.

-- [En este caso FOREIGN KEY asegura que cada id_departamento en la tabla Profesores debe coincidir con un id_departamento existente en la tabla departamentos.

-- Si intentas insertar un nuevo profesor con un id_departamento que no existe en la tabla departamentos, la inserción fallará.]

```
CREATE TABLE Profesores (
    id_profesores INT PRIMARY KEY,
    nombre VARCHAR(100),
    id_departamento INT,
    FOREIGN KEY (id_departamento) REFERENCES departamentos (id_departamento)
);
```

Eliminar Tabla:

--Elimina la tabla entera con el nombre "JEEP".

```
DROP TABLE JEEP;
```

Añadir columna:

-- Para añadir una columna, por ejemplo a columna "PRECIO" de la tabla "MERECEDES" tienes que escribir:

```
ALTER TABLE MERCEDES
ADD PRECIO NUMBER(8,2);
```

Eliminar Columna:

--Para eliminar una sola columna, por ejemplo la columna "PRECIO" DE LA TABLA "MERCEDES" tienes que escribir lo siguiente.

```
ALTER TABLE MERCEDES
DROP COLUMN PRECIO;
```

Añadir datos a Tabla:

-- Para insertar datos en una tabla, se tiene que hacer:

-- Recuerda no poner más números o caracteres de los establecidos al declarar la tabla.
-- Para insertar una cadena de caracteres, tienes que ponerlo entre comillas simples(').
-- Al meter el precio en el "NUMBER(8,2)", no es obligatorio poner los decimales si el precio es entero.

```
INSERT INTO MERCEDES (ID_MODELO, N_MODELO, PRECIO)
VALUES (00003,'SLS',245640.65);
```

Actualizar datos de la Tabla:

-- Muestra todos los datos de la tabla.

*-- SELECT * FROM VENTAS; (Descomenta para usarlo)*

-- Actualiza un dato concreto en la tabla.

```
UPDATE VENTAS
SET PRECIO = 43.20
WHERE ID_PRODUCTO = 101 AND ID = 8;
```

-- Eliminar un dato concreto en la tabla.

```
DELETE FROM VENTAS
WHERE ID = 5;
```

-- Vacía todas las columnas de la tabla.

TRUNCATE TABLE HORARIO

Fecha del Sistema:

-- Para sacar la fecha del sistema, escribe:

```
SELECT SYSDATE
FROM DUAL;
```

-- Para sacar la hora del sistema, escribe:

```
SELECT TO_CHAR(SYSDATE, 'HH24:MI:SS') AS Hora
FROM DUAL;
```

-- Para insertar la fecha actual a una columna, escribe:

-- Tomemos como ejemplo una tabla EMPLEADOS, que le queremos meter la fecha de hoy como el día que empezó a trabajar un trabajador.
-- Recuerda que el dato debe ser tipo "DATE" para introducir la fecha del sistema.

```
INSERT INTO EMPLEADOS
VALUES (3456,'Jose',10234,SYSDATE);
```

-- Para introducir en una columna la hora, introduce:

-- Recuerda que la columna debe ser "VARCHAR2(8)".

```
INSERT INTO HORARIO
VALUES ('Martes',TO_CHAR(SYSDATE, 'HH24:MI:SS'));
```

Mostrar datos de Tabla:

-- Para sacar datos de una tabla, haz:

```
SELECT ID_MODELO,N_MODELO,PRECIO
FROM MERCEDES
```

-- Para sacar datos de una tabla, llamndole a las columas de una manera diferente, haz:

```
SELECT ID_MODELO AS "ID",N_MODELO AS "NOMBRE",PRECIO
FROM MERCEDES
```

-- Para sacar datos de una tabla, con condicion, haz:

--(En este caso, solo muestra los Mercedes con ID superior a 1)

```
SELECT ID_MODELO,N_MODELO,PRECIO
FROM MERCEDES
WHERE ID_MODELO>1;
```

-- Para sacar datos de una tabla, ordenado de forma ascendente o descendente, haz:

--[Ascendente = ASC || Descendente = DESC]

```
SELECT ID_MODELO,N_MODELO,PRECIO
FROM MERCEDES
ORDER BY ID_MODELO ASC
```

```
-- Base de datos usada: (Descomentala y usala antes de hacer el código de abajo para usarla)

--CREATE TABLE VENTAS (
--    ID INT PRIMARY KEY,
--    ID_PRODUCTO INT,
--    FECHA DATE,
--    CANTIDAD INT,
--    PRECIO DECIMAL(10, 2)
-- );


-- En primer lugar, hacemos tres columnas: "ID", "Total Cantidad" que contiene la suma de todas las cantidades con el mismo ID y "Total Ventas" que contiene la suma de todos los precios * cantidad.

-- Por último, los imprime agrupados por ID.

SELECT ID_PRODUCTO AS "ID", SUM(CANTIDAD) AS "Total Cantidad", SUM(PRECIO * CANTIDAD) AS "Total Ventas"
FROM VENTAS
GROUP BY ID_PRODUCTO;
```

```
-- Para contar el número de elementos de una columna, escribe COUNT(Nombre Columna).

SELECT COUNT(N_MODELO)
FROM MERCEDES;
```

```
-- Función "if()else()" en SQL:
-- Sintaxis: CASE WHEN condición THEN resultado ELSE otro_resultado END

SELECT PRECIO, CASE WHEN PRECIO > 219300 THEN PRECIO * 2 ELSE PRECIO / 2 END AS "Precios Modificados"
FROM MERCEDES;
```

```
-- Para eliminar duplicados de las columnas,escribe:

SELECT DISTINCT ID_PRODUCTO
FROM VENTAS
```

-- Calcula el número de meses entre dos fechas.

-- ROUND sirve para redondear el número al número de decimales que pongas. en este caso 2.

```
SELECT ROUND(MONTHS_BETWEEN('11-23-2025','01-02-2024'),2)
FROM DUAL;
```

-- Extrae componentes específicos de una fecha (año, mes o día)
-- [MONTH | DAY | YEAR]

```
SELECT EXTRACT (YEAR FROM FECHA)
FROM VENTAS;
```

-- Para imprimir en una columna el valor de las columnas que quieras, haz:

```
SELECT ID_MODELO||N_MODELO AS "MODELOS"
FROM MERCEDES;
```

-- Para imprimir en una tabla el valor de las tablas que quieras, con un mensaje/caracter/espacio entre datos, haz:

```
SELECT ID_MODELO||'_'||N_MODELO||'_'||PRECIO AS "MODELOS"
FROM MERCEDES;
```

-- Otra forma para imprimir en una tabla el valor de las tablas que quieras:

```
SELECT N_MODELO||q'[ Precio del modelo: ]'||PRECIO AS "MODELOS"
FROM MERCEDES;
```

-- Puede o cumplir una u otra condición.

```
SELECT *
FROM Empleados
WHERE EDAD > 35 OR SALARIO > 30000;
```

-- Te da todos los datos del usuario que tenga el ID que escribas por teclado.

```
SELECT *
FROM EMPLEADOS
WHERE ID_EMPLEADO = :NUMEROID;
```

-- Para pasar una columna a minúscula.

```
SELECT LOWER(NOMBRE) AS NOMBRE_EN_MINUSCULAS
FROM EMPLEADOS;
```

-- Para pasar una columna a mayúscula.

```
SELECT UPPER(NOMBRE) AS NOMBRE_EN_MAYUSCULA
FROM EMPLEADOS;
```

-- La primera en mayúscula y las demás en minúscula.

```
SELECT INITCAP(NOMBRE) AS NOMBRE
FROM EMPLEADOS;
```

-- Solo imprime los caracteres entre una posición y otra.

```
SELECT SUBSTR(NOMBRE,1,3) AS NOMBRE
FROM EMPLEADOS;
```

-- Para saber el número de caracteres de una palabra, haz:

```
SELECT LENGTH(NOMBRE) AS "NUMERO DE CARACTERES"
FROM EMPLEADOS;
```

-- Para saber la posición de una letra, haz:

--(Solo reconoce la primera y la que es minúscula en este caso)

```
SELECT NOMBRE,INSTR(NOMBRE, 'a') AS "POSICION_a"
FROM EMPLEADOS;
```

-- Pone a la izquierda cuantos caracteres quieras.

-- (En el caso de que quieras hacer lo mismo pero hacia la derecha, usa "RPAD" en vez de "LPAD".)

```
SELECT LPAD(ID_EMPLEADO, 5, '0') AS ID_EMPLEADO_COMPLETO
FROM Empleados;
```

-- Permite recortar los caracteres finales o de encabezado (o ambos) de una cadena de caracteres.

```
SELECT DIRECCION, LENGTH(DIRECCION) AS "LONGITUD SIN TRIM",
TRIM(DIRECCION) AS "DIRECCION CON TRIM", LENGTH(TRIM(DIRECCION)) AS
"LONGITUD CON TRIM"
FROM EMPLEADOS2;
```

-- Reemplaza una palabra por otra. (También puedes hacerlo con números. pero siempre entre '')

```
SELECT DIRECCION, REPLACE(DIRECCION, 'Calle', 'Avenida') AS
DIRECCION_MODIFICADA
FROM EMPLEADOS2;
```

-- ROUND redondea a lo que tu quieras un número.
-- TRUNC trunca el número a lo que tu quieras.
-- MOD devuelve el resto de la división.

-- Para ver la diferencia entre ROUND y TRUNC, el número debe tener más de dos decimales. (Con dos no hay diferencia)

```
SELECT PRECIO, ROUND(PRECIO,2) AS "PRECIO REDONDEADO",
TRUNC(PRECIO,2) AS "PRECIO TRUNCADO", MOD(PRECIO,30) AS "RESTO DE LA
DIVISION"
FROM MERCEDES;
```

-- PARA CALCULAR LOS MESES ENTRE DOS FEHCAS:

```
SELECT FECHA_CONTRATACION,
MONTHS_BETWEEN(SYSDATE,FECHA_CONTRATACION) AS MESES
FROM EMPLEADOS3;
```

-- PARA AÑADIR LOS MESES QUE QUIERAS A UNA FECHA:

```
SELECT FECHA_CONTRATACION, ADD_MONTHS(FECHA_CONTRATACION,5) AS
"MESES+5"
FROM EMPLEADOS3;
```

-- Para saber el proximo dia de la semana, pones la fecha de hoy y el día de la semana.

-- (En este caso hoy es viernes y el próximo viernes será el que te muestre.)

```
SELECT FECHA_CONTRATACION, NEXT_DAY(FECHA_CONTRATACION,'FRIDAY') AS
"PROXIMO VIERNES"
FROM EMPLEADOS3;
```

-- Para saber el último día del mes.

```
SELECT FECHA_CONTRATACION, LAST_DAY(FECHA_CONTRATACION) AS "ULTIMO
DIA DEL MES"
FROM EMPLEADOS3;
```

-- Pasa el mes a char.

```
SELECT TO_CHAR(FECHA_CONTRATACION, 'DD-MON-YYYY') AS FECHA_FORMATO
FROM EMPLEADOS3;
```

-- Pasa de una cadena de caracteres a un decimal.

```
SELECT TO_CHAR(SALARIO, '$999G999D99') AS SALARIO
FROM EMPLEADOS3;
```

-- Ejemplo de TO_CHAR con fechas:

```
SELECT NOMBRE, APELLIDO, TO_CHAR(FECHA_CONTRATACION,'MONTH DAY
YEAR') AS "FECHA CONTRATACION"
FROM EMPLEADOS3;
```

-- Si el valor es NULL, devuelve 0.

```
SELECT NVL(SALARIO, 0) AS SALARIO
FROM EMPLEADOS3;
```

-- Si es NULL devuelve "Sin salario", sino, devuelve "Con salario".

```
SELECT NOMBRE,NVL2(SALARIO, 'Con salario', 'Sin salario') AS "SALARIO"
FROM EMPLEADOS3;
```

-- Compara dos expresiones y devuelve un valor nulo si son iguales; si no son iguales, devuelve la primera expresión.

```
SELECT NULLIF(SALARIO, 28000) AS "SALARIO MODIFICADO"
FROM EMPLEADOS3;
```

-- "COALESCE" devuelve la primera expresión no nula en la lista de expresiones.

```
SELECT ID_MOTO,MARCA, MODELO, COALESCE(PRECIO_ANUAL, PRECIO * 12) AS
"PRECIO ANUAL MODIFICADO"
FROM MOTOS;
```

-- "CASE" realiza evaluaciones condicionales y devuelve un valor específico basado en una o más condiciones.

```
SELECT NOMBRE,SALARIO,CASE
       WHEN SALARIO < 2000 THEN 'Bajo'
       WHEN SALARIO >= 20000 AND SALARIO < 40000 THEN 'Medio'
       ELSE 'Alto'
   END AS Categoria_salario
FROM
   EMPLEADOS;
```

-- DECODE se utiliza para comparar.

-- En este caso si el resultado del ROUND es 0 = Bajo, si es 1 = Medio y si es otro da Alto.

```
SELECT NOMBRE,SALARIO,DECODE(ROUND(SALARIO / 60000),0, 'Bajo',1,
'Medio','Alto') AS Categoria_salario
FROM EMPLEADOS;
```

-- Diferentes operaciones:

```
SELECT AVG(SALARIO) AS "VALOR MEDIO",MAX(SALARIO) AS "VALOR MAXIMO",
MIN(SALARIO) AS "VALOR MINIMO",
STDDEV(SALARIO) AS "DESVIACION ESTANDAR", SUM(SALARIO) AS "SUMA",
VARIANCE(SALARIO) AS "VARIANZA"
FROM EMPLEADOS3;
```

-- "HAVING" se utiliza para filtrar los resultados de una consulta después de que se ha aplicado la agrupación con "GROUP BY".

```
SELECT ID_PRODUCTO AS "ID PRODUCTO", SUM(CANTIDAD) AS "cANTIDAD
TOTAL"
FROM VENTAS
GROUP BY ID_PRODUCTO
HAVING SUM(CANTIDAD)>20
```

-- JOIN se utiliza para combinar filas de dos o más tablas en función de una columna relacionada entre ellas.

-- Esto permite acceder a datos de múltiples tablas en una sola consulta.

```
SELECT E.NOMBRE, E.APELLIDO, V.CANTIDAD
FROM EMPLEADOS3 E
JOIN VENTAS V ON E.ID_EMPLEADO = V.ID_VENTA;
```

-- "NATRUAL JOIN" se utiliza para combinar filas de dos o más tablas basándose en columnas con el mismo nombre en ambas tablas.

```
SELECT E.NOMBRE, E.APELLIDO, V.CANTIDAD
FROM EMPLEADOS3 E
NATURAL JOIN VENTAS V;
```

-- USING, solo se puede usar si en ambas tablas la columna se llama igual.

```
SELECT EMPLEADOS4.NOMBRE, EMPLEADOS4.APELLIDO, VENTAS2.CANTIDAD
FROM EMPLEADOS4
JOIN VENTAS2
USING (ID_VENTA);
```

-- Comprobar coincidencias entre tablas:

-- LEFT OUTER: devuelve todas las filas de la tabla, pero si no coincide NULL a la derecha.
-- RIGHT OUTER: devuelve todas las filas de la tabla, pero si no coincide NULL a la izquierda.
-- FULL OUTER: devuelve todas las filas de la tabla con los respectivos NULL.

```
SELECT *
FROM EMPLEADOS4 E
FULL OUTER JOIN VENTAS2 V
ON E.ID_VENTA = V.ID_VENTA;
```

-- "CROSS JOIN" combina cada fila de la primera tabla con cada fila de la segunda tabla.

```
SELECT *
FROM EMPLEADOS4 E
CROSS JOIN VENTAS2 V;
```

-- Ejemplo de subrutina:

```
SELECT NOMBRE,APELLIDO, SALARIO
FROM EMPLEADOS3
WHERE SALARIO > (
   SELECT AVG(SALARIO)
   FROM EMPLEADOS3
   );
```

```
-- EXISTS se utiliza para comprobar si existen filas que satisfacen cierta condición.
-- En este caso si al menos un elemento cumple "E.ID_venta = V.ID_venta", entonces se mostrará la columna entera.

SELECT E.NOMBRE ||' '||E.APELLIDO AS "EMPLEADO"
FROM EMPLEADOS4 E
WHERE EXISTS (
   SELECT 1
   FROM VENTAS2 V
   WHERE E.ID_venta = V.ID_venta
);
```

```
-- NOT EXISTS se utiliza para comprobar si no existen filas que satisfacen cierta condición.
-- Si existe al menos una venta que coincida, entonces el nombre del empleado no se incluye en el resultado final.

SELECT E.NOMBRE ||' '||E.APELLIDO AS "EMPLEADO"
FROM EMPLEADOS4 E
WHERE NOT EXISTS (
   SELECT 1
   FROM VENTAS2 V
   WHERE E.ID_venta = V.ID_venta
);
```

```
--UNION se utiliza para combinar los resultados de dos o más sentencias

SELECT  ID_VENTA
FROM EMPLEADOS4
UNION
SELECT ID_VENTA
FROM VENTAS2;
```

```
--INTERSECT se utiliza para devolver todas las filas comunes a varias consultas.


SELECT  ID_VENTA
FROM EMPLEADOS4
INTERSECT
SELECT ID_VENTA
FROM VENTAS2;
```

--MINUS se utiliza para que devuelva todas las filas en la primera consulta que no están
presentes en la segunda consulta.

```
SELECT  ID_VENTA
FROM EMPLEADOS4
MINUS
SELECT ID_VENTA
FROM VENTAS2;
```

-- FOR UPDATE se utiliza para bloquear filas específicas de una tabla mientras se ejecuta una transacción.

```
DECLARE
   NUMERO NUMBER;
BEGIN

   SELECT 1
   INTO NUMERO
   FROM EMPLEADOS3
   WHERE ID_EMPLEADO = 1
   FOR UPDATE;


   UPDATE EMPLEADOS3
   SET SALARIO = SALARIO + 100
   WHERE ID_EMPLEADO = 1;


   COMMIT;
END;
```

Mostrar información de la Tabla:

-- Para mostrar la información de la tabla, escribe:

```
DESC[RIBE] MERCEDES;
```

Modos de Tabla:

-- Si haces READ ONLY, no podrás modificar la tabla, solo leerla.

```
ALTER TABLE MERCEDES READ ONLY;
```

-- Si haces READ WRITE, podrás leer y modificar la tabla.

```
ALTER TABLE MERCEDES READ WRITE;
```

Crear Vista:

-- CREATE VIEW se utiliza para crear una vista.
-- Una vista es una tabla virtual basada en el resultado de una consulta SQL.

```
CREATE VIEW VISTA_BARCOS AS
SELECT *
FROM BARCOS
WHERE LONGITUD > 30;
```

-- CREATE OR REPLACE se utiliza para crear una vista nueva o reemplazar una vista existente con el mismo nombre.

```
CREATE OR REPLACE VIEW VISTA_BARCOS AS
SELECT *
FROM BARCOS
WHERE LONGITUD>30;
```

-- WITH CHECK OPTION asegura que cualquier modificación (inserción o actualización) realizada a través de la vista no introduzca datos que no cumplan con los criterios de la vista.

```
CREATE OR REPLACE VIEW VISTA_BARCOS AS
SELECT *
FROM BARCOS
WHERE LONGITUD>30
WITH CHECK OPTION;
```

-- WITH READ ONLY en SQL se utiliza al crear una vista para hacer que esa vista sea de solo lectura.

-- Esto significa que no se pueden realizar operaciones de inserción, actualización o eliminación a través de esa vista.

```
CREATE OR REPLACE VIEW VISTA_BARCOS AS
SELECT *
FROM BARCOS
WHERE LONGITUD>30
WITH READ ONLY;
```

Eliminar Vista:

-- Elimina la vista.

```
DROP VIEW VISTA_BARCOS;
```

Crear Secuencia:

```
--Crear Secuencia.

CREATE SEQUENCE S_EMPLEADOS -- Incrementa el valor de la secuencia en 10
cada vez
INCREMENT BY 10 -- El valor inicial de la secuencia es 120
START WITH 120  -- El valor máximo que puede alcanzar la secuencia es 9999
MAXVALUE 9999   -- No almacena los valores en caché
NOCACHE -- La secuencia no se reinicia una vez alcanzado el valor máximo
NOCYCLE;
```

NEXTVAL:

```
-- Obtiene el siguiente valor de la secuencia, incrementándose

SELECT S_EMPLEADOS.NEXTVAL
FROM dual;
```

Modificar Secuencia:

```
-- Modificación de una Secuencia

ALTER SEQUENCE S_EMPLEADOS
INCREMENT BY 20
MAXVALUE 999999
NOCACHE
NOCYCLE;
```

Eliminar Secuencia:

```
-- Eliminar Secuencia

DROP SEQUENCE S_EMPLEADOS
```

Crear Indice:

```
-- Crear un Índice:

CREATE INDEX INDICE_NOMBRES
ON EMPLEADOS(NOMBRE);
```

Eliminar Indice:

```
-- Eliminar un Índice:

DROP INDEX INDICE_NOMBRES;
```

Crear Sinónimo:

```
-- Crear Sinonimo:

CREATE SYNONYM SINOM_MERCEDES
FOR MERCEDES;
```

Eliminar Sinónimo:

```
-- Eliminar Sinonimo:

DROP SYNONYM SINOM_MERCEDES;
```

Punto de Guardado y Retroceso:

```
BEGIN
  SAVEPOINT GUARDAR;    -- Crea punto de guardado.
  DELETE FROM EMPLEADOS3;
  ROLLBACK TO GUARDAR;   -- Retrocede al punto de guardado.
END;
```