

2da ENTREGA DEL PROYECTO FINAL

**CURSO SQL - CODERHOUSE
COMISIÓN #50055**

JANO AUGUSTO MENAZZI BALDINI

ÍNDICE

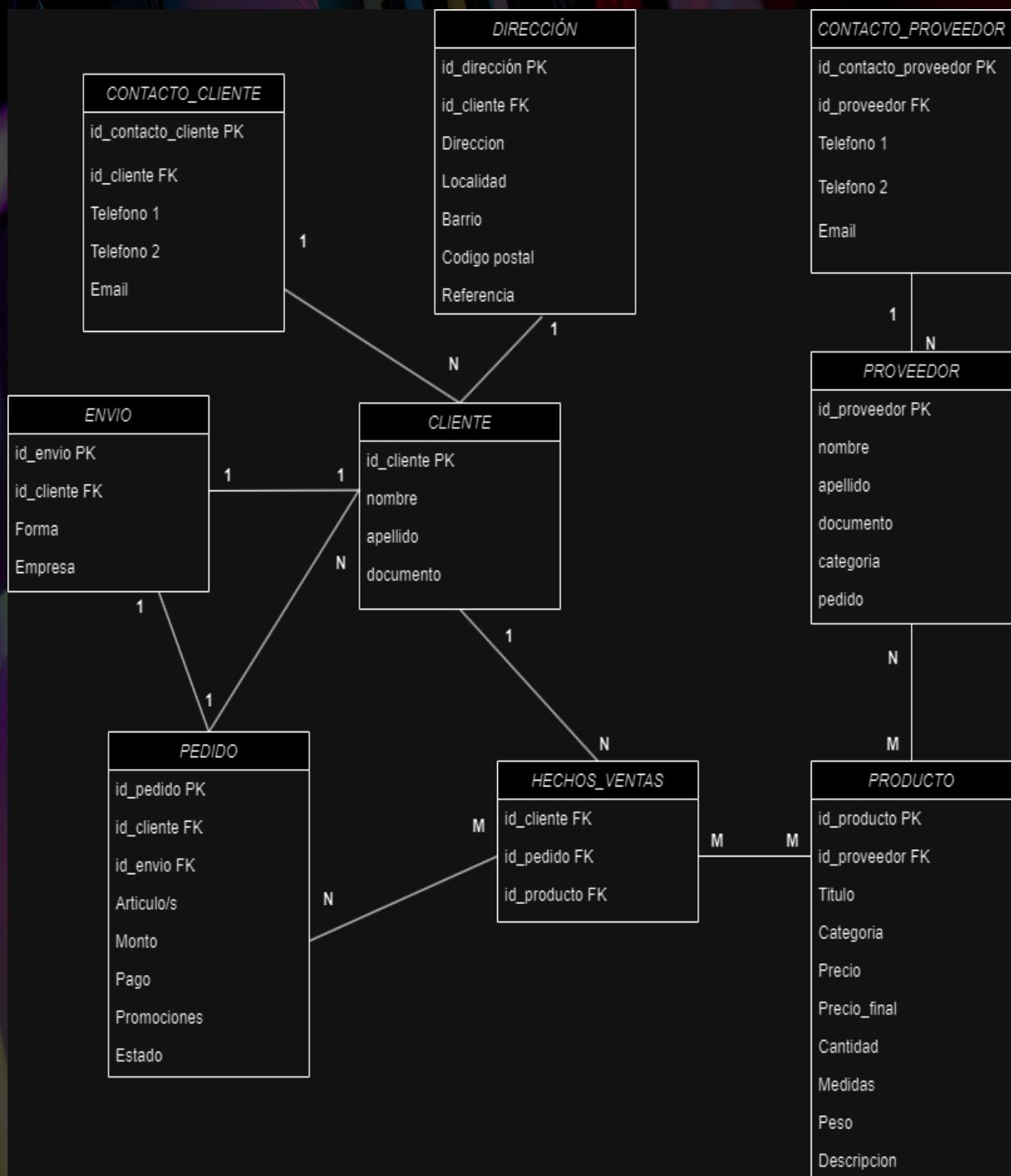
TEMÁTICA PROYECTO FINAL.....	3
DIAGRAMA - TIENDA.....	4
LISTADO DE VISTAS.....	5-6
LISTADO DE FUNCIONES.....	7
LISTADO DE STORED PROCEDURES.....	8
PASO A PASO INSERCIÓN DE DATOS MEDIANTE IMPORTACIÓN.....	9-11

TEMÁTICA PROYECTO FINAL

Mi objetivo con este curso es aprender las bases de SQL para poder aplicarlo a una tienda de componentes informáticos, con sus clientes, proveedores, productos, servicios, empleados, ventas, etc.

El objetivo de la base de datos es llevar organizado y detallado todo lo relacionado a la tienda, los costos, turnos, descuentos, desde los sueldos de los empleados hasta las ventas realizadas, con el fin de poder agilizar procesos y realizar acciones de marketing o demás con esa información.

DIAGRAMA - TIENDA



LISTADO DE VISTAS

Vista N°1 compuesta por las tablas “Producto” y “Proveedor”

```
-- vista1, productos que tienen como proveedor al id_proveedor = 1  
-- o su numero de documento "12457865"  
  
CREATE OR REPLACE VIEW VISTA1 AS  
SELECT P.id_producto, P.titulo, P.categoría, P.precio, A.nombre, A.id_proveedor  
FROM PRODUCTO P  
JOIN proveedor A ON (P.id_proveedor = A.id_proveedor)  
WHERE A.documento = 12457865;
```

Vista N°2 compuesta por la tabla “Producto”

```
-- vista2, productos que tienen un precio mayor a 1600 con categoria de gabinete  
  
CREATE OR REPLACE VIEW VISTA2 AS  
SELECT P.id_producto, P.titulo, P.categoría, P.precio  
FROM producto P  
WHERE categoría = "Gabinetes" AND precio > 1600;
```

Vista N°3 compuesta por la tabla “Producto”

```
-- vista3, vista de los datos que se deberian mostrar en la pagina web a los clientes  
-- (precio + 30%)  
  
CREATE OR REPLACE VIEW VISTA3 AS  
SELECT P.titulo, P.categoría, P.precio * 1.3 as PRECIO_FINAL  
FROM producto P order by precio;
```

Vista N°4 compuesta por las tablas “Pedido”, “Cliente” y “Contacto_cliente”

```
-- vista4, una vista que recopile de la tabla pedidos los que sean de pago con debito  
-- a su vez me muestre de la tabla cliente su nombre, apellido, documento  
-- y de la tabla contacto_cliente sus contactos
```

```
CREATE OR REPLACE VIEW VISTA4 AS
```

```
SELECT P.id_pedido, P.Articulos, P.Monto, P.Pago, C.nombre, C.apellido, C.documento, CC.telefono1, CC.telefono2, CC.email  
FROM pedido P  
JOIN cliente C ON P.id_cliente = C.id_cliente  
JOIN contacto_cliente CC ON P.id_cliente = CC.id_cliente  
WHERE P.Pago = 'Debito' order by Monto;
```

Vista N°5 compuesta por las tablas “Cliente”, “Pedido”, “Contacto_cliente” y “Envio”

```
-- vista5 hacer una vista que me muestre al cliente, su contacto, y su pedido  
-- sabiendo que su pedido tiene definido el envio por la empresa andreani  
-- supuesto caso de que se atrasen los envios y haya que avisar a los clientes
```

```
CREATE OR REPLACE VIEW VISTA5 AS
```

```
SELECT C.nombre, C.apellido, C.documento, CC.telefono1, CC.telefono2, CC.email, P.id_pedido, P.Articulos, E.id_envio, E.empresa  
FROM cliente C  
JOIN pedido P ON P.id_cliente = C.id_cliente  
JOIN contacto_cliente CC ON P.id_cliente = CC.id_cliente  
JOIN envio E ON E.id_envio = P.id_envio  
WHERE E.empresa = 'Andreani';
```

LISTADO DE FUNCIONES

Función N°1

```
-- funcion1, esta destinada a la tabla producto, manipula el monto de estos,  
-- una funcion que le de como parametro un monto de un pedido/articulo  
-- y me devuelva su monto final con promociones/descuentos  
  
DELIMITER $$  
CREATE FUNCTION f_descuento(monto DECIMAL(10,2), descuento INT) RETURNS DECIMAL(10,2) READS SQL DATA  
BEGIN  
  
DECLARE resultado DECIMAL(10,2);  
SET resultado = monto - (monto * (descuento/100));  
  
RETURN resultado;  
END$$  
DELIMITER ;
```

Función N°2

```
-- funcion2, una funcion para calcular el precio total de un producto  
-- dado su precio unitario y la cantidad a comprar, devuelve el precio final de esa cantidad unitaria  
-- multiplicada por la cantidad  
  
DELIMITER $$  
CREATE FUNCTION f_precio_final(precio_unitario DECIMAL(10,2), cantidad INT) RETURNS DECIMAL(10,2) READS SQL DATA  
BEGIN  
  
DECLARE total DECIMAL(10,2);  
SET total = precio_unitario * cantidad;  
  
RETURN total;  
END$$  
DELIMITER ;
```

LISTADO DE STORED PROCEDURES

SP Nº1

```
/* sp1, este lo hicimos tal cual en clase, estaba bastante automatizado para poder
hacer el proceso para cualquier tabla, se le puede sumar la seguridad de los IF de corroborar que los registros ingresados
esten en las tablas, y que a la vez sea valido el orden */

drop procedure if exists sp_ordenar_tabla;

DELIMITER $$

CREATE PROCEDURE sp_ordenar_tabla(IN tabla VARCHAR(20), IN campo VARCHAR(30), in orden VARCHAR(4))
BEGIN

    SET @ordenar = CONCAT('SELECT * FROM', ' ', tabla, ' ', 'ORDER BY', ' ', campo, ' ', orden);

    PREPARE ordenar FROM @ordenar;
    EXECUTE ordenar;
    DEALLOCATE PREPARE ordenar;

END$$

DELIMITER ;
```

SP Nº2

```
/* sp2 sp que inserte registros en la tabla cliente, este sp no tiene como parametro el id ya que este es auto_increment
y se deberia asignar solo, a su vez tiene un IF para corroborar que el DNI del nuevo registro este ya registrado*/

• DROP PROCEDURE IF EXISTS sp_insertar_registro_cliente;

DELIMITER $$

• CREATE PROCEDURE sp_insertar_registro_cliente(IN n_nombre VARCHAR(30), IN n_apellido VARCHAR(30), IN n_documento VARCHAR(14))
BEGIN

    DECLARE documento_existente INT;

    SELECT COUNT(*) INTO documento_existente FROM cliente WHERE documento = n_documento;

    IF documento_existente = 0 THEN
        INSERT INTO cliente
        (id_cliente, nombre, apellido, documento) VALUES (default, n_nombre, n_apellido, n_documento);
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El documento ingresado ya esta asignado a otro cliente';
    END IF;

    /* este procedimiento guarda en "documento_existente" el conteo del total de los documentos
    encontrados iguales al ingresado como parametro en "n_documento"
    si hay algun documento igual al que se ingreso como parametro, "documento_existente" se carga con esa cantidad
    que deberia ser 1 maximo ya que documento es UNIQUE
    si es asi, salta error asignado, sino sigue la carga del nuevo registro con ese documento, ya que es valido. */

END$$

DELIMITER ;
```

PASO A PASO INSERCIÓN DE DATOS MEDIANTE IMPORTACIÓN

1. Tenemos la carpeta llamada “CSV” la cual contiene todos los archivos de inserción de datos para la BD “curso_coderhouse” del archivo .sql “llamado PROYECTO-MENAZZI-JANO”.
2. Estos archivos CSV están numerados del 1 al 8 en el que se deben insertar.

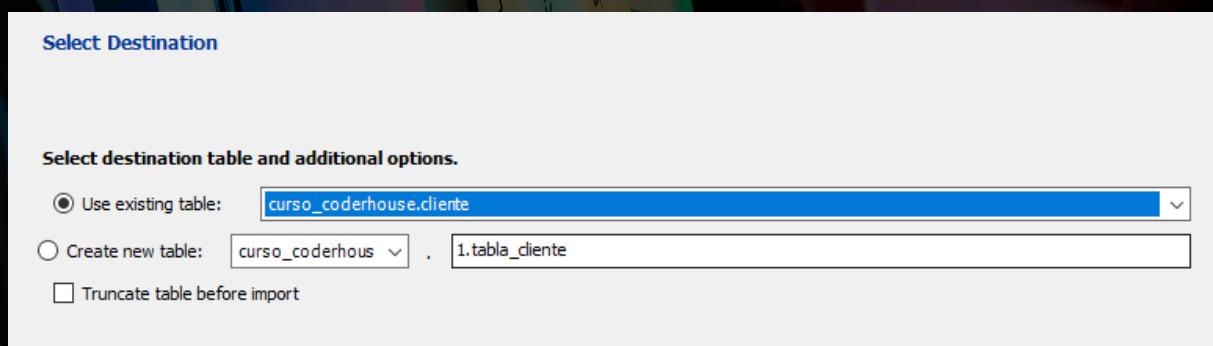
1.tabla_cliente.csv	4/12/2023 20:38	Archivo de valores...	1 KB
2.tabla_contacto_cliente.csv	4/12/2023 20:44	Archivo de valores...	1 KB
3.tabla_direccion.csv	4/12/2023 20:47	Archivo de valores...	1 KB
4.tabla_envio.csv	4/12/2023 20:49	Archivo de valores...	1 KB
5..csv	4/12/2023 20:53	Archivo de valores...	1 KB
6.tabla_proveedor.csv	4/12/2023 20:55	Archivo de valores...	1 KB
7.tabla_contacto_proveedor.csv	4/12/2023 20:57	Archivo de valores...	1 KB
8.tabla_producto.csv	16/12/2023 01:15	Archivo de valores...	1 KB

3. Llamamos a la tabla la cual queremos insertar los datos con un SELECT, por ejemplo la tabla “cliente”, nos debería aparecer en la parte de abajo algo como esto:

Result Grid Filter Rows: <input type="text"/> Edit: Export/Import: Wrap Cell Content:			
id_cliente	nombre	apellido	documento

4. En la parte de “EXport/Import:”, debemos seleccionar la opcion de “Import” para justamente importar los registros del .CSV dentro de la tabla seleccionada.

5. Los siguientes pasos son más simples, en la primer pestaña buscamos el archivo .CSV en el equipo, luego nos preguntará si queremos usar una tabla existente o crear una nueva, en nuestro caso ya tenemos la tabla la cual es “cliente”, ¡corroborar que la tabla existente seleccionada sea la correcta!.



6. Como ante-último paso debemos corroborar que las columnas del archivo .CSV y las columnas de la tabla coincidan, tal cual la imagen (si queremos insertar todas las columnas), en la pestaña de abajo deberíamos ver una muestra de los datos como quedarían insertados en la tabla.

Configure Import Settings

Detected file format: csv 

Encoding: utf-8

Columns:

Source Column	Dest Column
<input checked="" type="checkbox"/> id_cliente	<input type="text" value="id_cliente"/>
<input checked="" type="checkbox"/> nombre	<input type="text" value="nombre"/>
<input checked="" type="checkbox"/> apellido	<input type="text" value="apellido"/>
<input checked="" type="checkbox"/> documento	<input type="text" value="document"/>

id_cliente	nombre	apellido	documento
1	jacinto	matas	17308014
2	marco	peller	16457852
3	teofilo	padron	16354985
4	jose	leiva	43658412
5	ocio	carbonell	38256415

7. Damos “Next” abajo a la derecha, al hacer “Next” por última vez se ejecuta la importación de datos, para corroborar esta inserción de datos hacemos lo mismo que en le paso nº3, llamamos a la tabla con un SELECT y deberíamos ver los registros!