



# EVALUACIÓN PROCESUAL HITO 3

BASE DE DATOS I

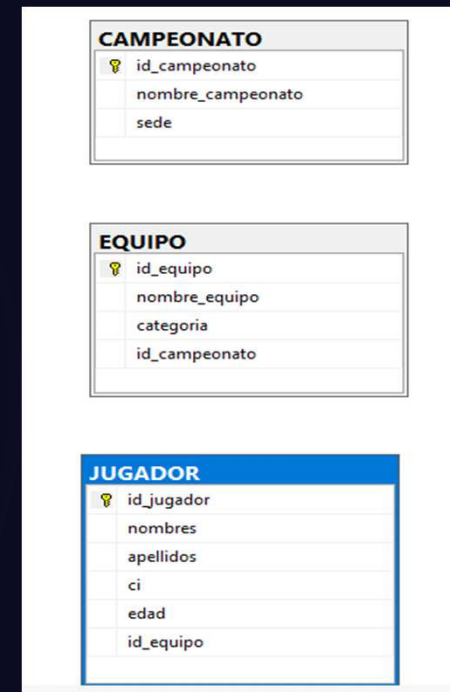
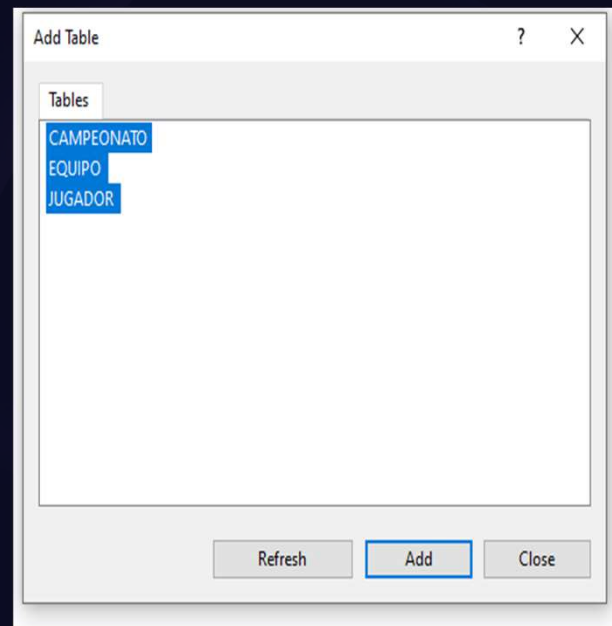
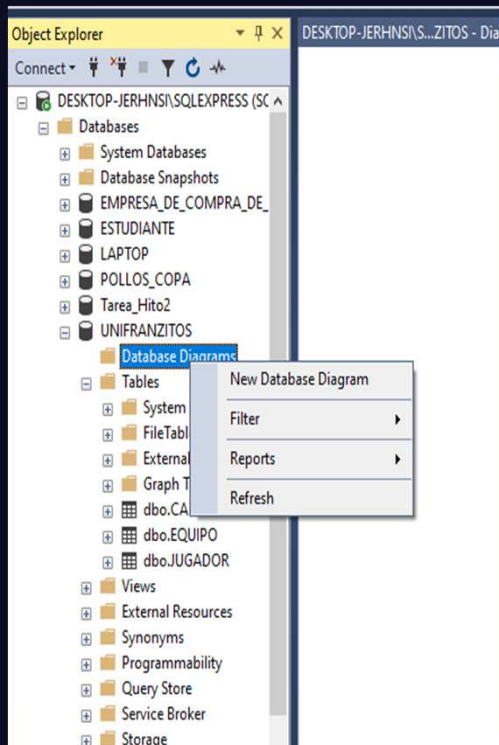
PRESENTA:

SANTOS BRAYAN HUMIRI QUISPE

SIS-12925430

# Diagrama E-R en SQL Server

El diagrama E-R es una representación visual de la estructura de una base de datos. Puede ser creado utilizando herramientas como DATAGRIP o SQL Server Management Studio.



# DDL en SQL Server

DDL (Data Definition Language) se utiliza para definir la estructura de la base de datos.

**CREATE** *(Crea base de datos o tabla)*

**CREATE DATABASE** personal;

**CREATE TABLE** usuarios;

**DROP** *(Elimina base de datos o tabla)*

**DROP DATABASE** personal;

**DROP TABLE** usuarios;

**DROP FOREIGN KEY** id\_curso;

**ALTER** *(Modifica tabla, vista, atributos )*

**ALTER TABLE** usuarios **ADD** edad int;

**ALTER TABLE** usuarios **DROP COLUMN** edad;

**ALTER TABLE** usuarios **ADD CONSTRAINT**

**FK\_Curso FOREIGN KEY** (id\_curso)

**REFERENCES** Cursos(id\_curso)

**TRUNCATE** *(Elimina el contenido de una tabla)*

**TRUNCATE TABLE** usuarios;

# DML en SQL Server

DML (Data Manipulation Language) se utiliza para manipular los datos. datos.

**INSERT** (Almacena información)

**INSERT INTO** usuarios (nombre ,edad) VALUES ("Luis", 33);

**DELETE** (Borra información)

**DELETE FROM** usuarios WHERE nombre="Luis";

**SELECT** (Consulta información)

**SELECT \* FROM** usuarios;

**SELECT** nombre, edad **FROM** usuarios;

# PRIMARY KEY y FOREIGN KEY

- PRIMARY KEY es una restricción que identifica de manera única cada registro en una tabla.

La clave primaria es un campo que identifica de manera única a cada registro de una tabla (Id).

- ❖ El valor debe ser único
- ❖ Tiene que ser **NOT NULL**  
(no se puede dejar en blanco)

- FOREIGN KEY es una restricción que establece una relación entre dos tablas.

La clave foránea es un campo que relaciona una tabla con otra utilizando la clave primaria (PRIMARY KEY) de la otra tabla.

- ❖ sigue la regla de integridad referencial



# Tablas e IDENTITY

Una tabla en SQL Server es una estructura que almacena datos en filas y columnas.

IDENTITY es una propiedad que se utiliza para generar valores únicos automáticamente en una columna.

```
CREATE TABLE Editorials
(
    ID INT IDENTITY(1,1) NOT NULL, -- IDENTITY(SEED - SEMILLA, INCREMENTO)
    Editorial NVARCHAR(250) NOT NULL
)
```

# La cláusula WHERE

- La cláusula WHERE se utiliza para filtrar resultados en una consulta. Permite especificar una condición que debe cumplirse para que los registros sean incluidos en el resultado.
- La cláusula WHERE le dice a SQL que incluya solo ciertas filas de datos en los resultados de la consulta.

```
SELECT (Columnas o Campos de la Tabla)  
FROM (Fuente de Datos o Tabla)  
WHERE (Condición de Filtrado de Filas)
```

# La instrucción INNER JOIN

La instrucción INNER JOIN se utiliza para combinar registros de dos o más tablas basándose en una condición. Solo se incluyen los registros que cumplen con la condición de unión.

```
CREATE TABLE Clientes
(
    ID_Cliente INT PRIMARY KEY,
    Nombre VARCHAR(50)
);
INSERT INTO Clientes (ID_Cliente, Nombre)
VALUES (1111, 'juan')
INSERT INTO Clientes (ID_Cliente, Nombre)
VALUES (2222, 'alvaro')
INSERT INTO Clientes (ID_Cliente, Nombre)
VALUES (3333, 'elio')
CREATE TABLE Pedidos
(
    ID_Pedido INT PRIMARY KEY,
    ID_Cliente INT,
    Producto VARCHAR(50),
);
INSERT INTO Pedidos (ID_Pedido, ID_Cliente, Producto)
VALUES (1, 1111, 'Pollos')
INSERT INTO Pedidos (ID_Pedido, ID_Cliente, Producto)
VALUES (2, 1111, 'jugos')
INSERT INTO Pedidos (ID_Pedido, ID_Cliente, Producto)
VALUES (3, 3333, 'alitas de pollo')
```

```
SELECT Clientes.Nombre, Pedidos.Producto
FROM Clientes
INNER JOIN Pedidos ON Clientes.ID_Cliente = Pedidos.ID_Cliente;
```

	Nombre	Producto
1	juan	Pollos
2	juan	jugos
3	elio	alitas de pollo

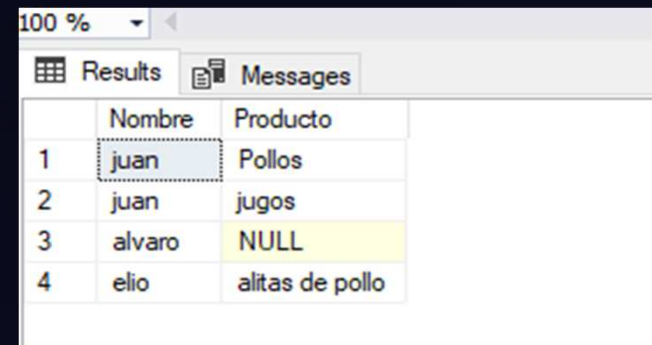


# INTRODUCCION A LEFT JOIN

En un LEFT JOIN se obtienen todos los registros de la tabla izquierda y los registros coincidentes de la tabla derecha. Si no hay coincidencias, se muestran valores nulos en los campos de la tabla derecha.

```
CREATE TABLE Clientes
(
    ID_Cliente INT PRIMARY KEY,
    Nombre VARCHAR(50)
);
INSERT INTO Clientes (ID_Cliente, Nombre)
VALUES (1111, 'juan')
INSERT INTO Clientes (ID_Cliente, Nombre)
VALUES (2222, 'alvaro')
INSERT INTO Clientes (ID_Cliente, Nombre)
VALUES (3333, 'elio')
CREATE TABLE Pedidos
(
    ID_Pedido INT PRIMARY KEY,
    ID_Cliente INT,
    Producto VARCHAR(50),
);
INSERT INTO Pedidos (ID_Pedido, ID_Cliente, Producto)
VALUES (1, 1111, 'Pollos')
INSERT INTO Pedidos (ID_Pedido, ID_Cliente, Producto)
VALUES (2, 1111, 'jugos')
INSERT INTO Pedidos (ID_Pedido, ID_Cliente, Producto)
VALUES (3, 3333, 'alitas de pollo')
```

```
SELECT Clientes.Nombre, Pedidos.Producto
FROM Clientes
LEFT JOIN Pedidos ON Clientes.ID_Cliente = Pedidos.ID_Cliente ;
```



	Nombre	Producto
1	juan	Pollos
2	juan	jugos
3	alvaro	NULL
4	elio	alitas de pollo

# INTRODUCCION A RIGHT JOIN

En un RIGHT JOIN se obtienen todos los registros de la tabla derecha y los registros coincidentes de la tabla izquierda. Si no hay coincidencias, se muestran valores nulos en los campos de la tabla izquierda.

```
CREATE TABLE Clientes
(
    Id_Cliente INT PRIMARY KEY,
    Nombre VARCHAR(50),
    Apellido VARCHAR(50),
    Ciudad VARCHAR(50)
);
INSERT INTO Clientes (Id_Cliente, Nombre, Apellido, Ciudad)
VALUES (1, 'Juan', 'Pérez', 'Madrid')

INSERT INTO Clientes (Id_Cliente, Nombre, Apellido, Ciudad)
VALUES (2, 'María', 'Gómez', 'Barcelona')
CREATE TABLE Pedidos
(
    Id_Pedido INT PRIMARY KEY,
    Id_Cliente INT,
    Fecha_Pedido DATE
);
INSERT INTO Pedidos (Id_Pedido, Id_Cliente, Fecha_Pedido)
VALUES (1, 1, '2023-10-15')

INSERT INTO Pedidos (Id_Pedido, Id_Cliente, Fecha_Pedido)
VALUES (2, 3, '2023-10-16')
CREATE TABLE Productos
(
    Id_Producto INT PRIMARY KEY,
    Nombre_Producto VARCHAR(50)
);
INSERT INTO Productos (Id_Producto, Nombre_Producto)
VALUES (1, 'Camiseta')

INSERT INTO Productos (Id_Producto, Nombre_Producto)
VALUES (2, 'Pantalón')
```

```
SELECT Clientes.Id_Cliente, Clientes.Nombre, Clientes.Apellido, Clientes.Ciudad, Pedidos.Id_Pedido, Pedidos.Fecha_Pedido
FROM Clientes
RIGHT JOIN Pedidos ON Clientes.Id_Cliente = Pedidos.Id_Cliente;
```

	Id_Cliente	Nombre	Apellido	Ciudad	Id_Pedido	Fecha_Pedido
1	1	Juan	Pérez	Madrid	1	2023-10-15
2	NULL	NULL	NULL	NULL	2	2023-10-16

# Uso de INNER JOIN con 3 tablas

```
CREATE TABLE Clientes
(
    Id_Cliente INT PRIMARY KEY NOT NULL,
    Nombre VARCHAR(50) NOT NULL,
    Apellido VARCHAR(50) NOT NULL,
    Ciudad VARCHAR(50) NOT NULL
);
INSERT INTO Clientes (Id_Cliente, Nombre, Apellido, Ciudad)
VALUES (1, 'Juan', 'Pérez', 'Madrid')

INSERT INTO Clientes (Id_Cliente, Nombre, Apellido, Ciudad)
VALUES (2, 'María', 'Gómez', 'Barcelona')
CREATE TABLE Pedidos
(
    Id_Pedido INT PRIMARY KEY NOT NULL,
    Id_Cliente INT NOT NULL,
    Fecha_Pedido DATE NOT NULL
);
INSERT INTO Pedidos (Id_Pedido, Id_Cliente, Fecha_Pedido)
VALUES (1, 1, '2023-10-15')

INSERT INTO Pedidos (Id_Pedido, Id_Cliente, Fecha_Pedido)
VALUES (2, 2, '2023-10-16')
CREATE TABLE Productos
(
    Id_Producto INT PRIMARY KEY NOT NULL,
    Nombre_Producto VARCHAR(50) NOT NULL
);
INSERT INTO Productos (Id_Producto, Nombre_Producto)
VALUES (1, 'Camiseta')

INSERT INTO Productos (Id_Producto, Nombre_Producto)
VALUES (2, 'Pantalón')
```

```
SELECT Clientes.Id_Cliente, Clientes.Nombre, Clientes.Apellido, Clientes.Ciudad, Pedidos.Id_Pedido, Pedidos.Fecha_Pedido
FROM Clientes
INNER JOIN Pedidos ON Clientes.Id_Cliente = Pedidos.Id_Cliente;
```

Results		Messages				
	Id_Cliente	Nombre	Apellido	Ciudad	Id_Pedido	Fecha_Pedido
1	1	Juan	Pérez	Madrid	1	2023-10-15
2	2	María	Gómez	Barcelona	2	2023-10-16