



Bases de Datos - SQL Server

Procesual **Hito 3**

Base de Datos I - 2021

Consigna

Diseñe un sistema de Base de Datos Relacional utilizando el gestor de Base de Datos **SQL Server** teniendo como premisa el uso de buenas prácticas en diseño de la base de datos aplicados al siguiente escenario.

Una pequeña empresa de comida rápida de nombre **the delicious** desea implementar un nuevo sistema para poder administrar los **PEDIDOS** de sus productos.

Detalles puntuales y análisis del problema

The delicious FF	
Problema	<p>En función al escenario se identificó que las posibles entidades son categorías, productos, cliente, pedido y detalle pedido, pues un cliente realiza un pedido de productos, en donde cada producto tiene una categoría a la cual pertenece y la descripción del pedido debería tener un detalle de pedido.</p> <p>En tal sentido se deberían crear las siguientes entidades y/o tablas.</p> <ul style="list-style-type: none">• categorias• productos• cliente• pedido• detalle_pedido <p>Detalle de las entidades.</p>

categorias

id_categoria => Primary key

tipo => posibles valores como (juguetes, verduras, etc)

productos

id_producto => Primary key

nombre

stock

precio_venta

precio_compra

categoria => Foreign key con la entidad categoria

cliente

id_cliente => primary key

nombres

apellidos

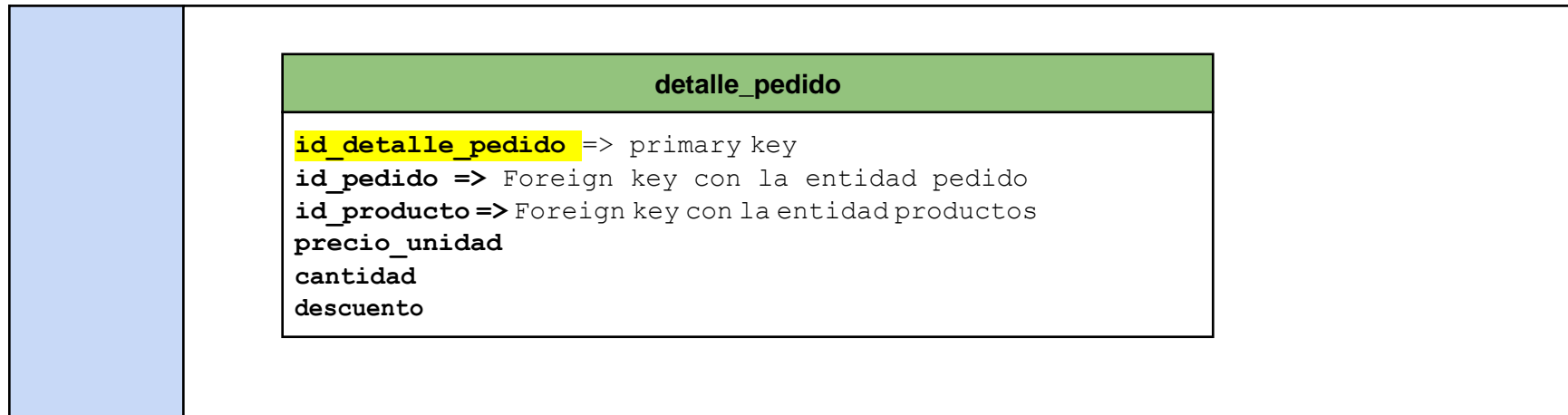
direccion

tipo_cliente => posibles valores como (GOLD, VIP y NORMAL)

pedido

id_pedido => primary key

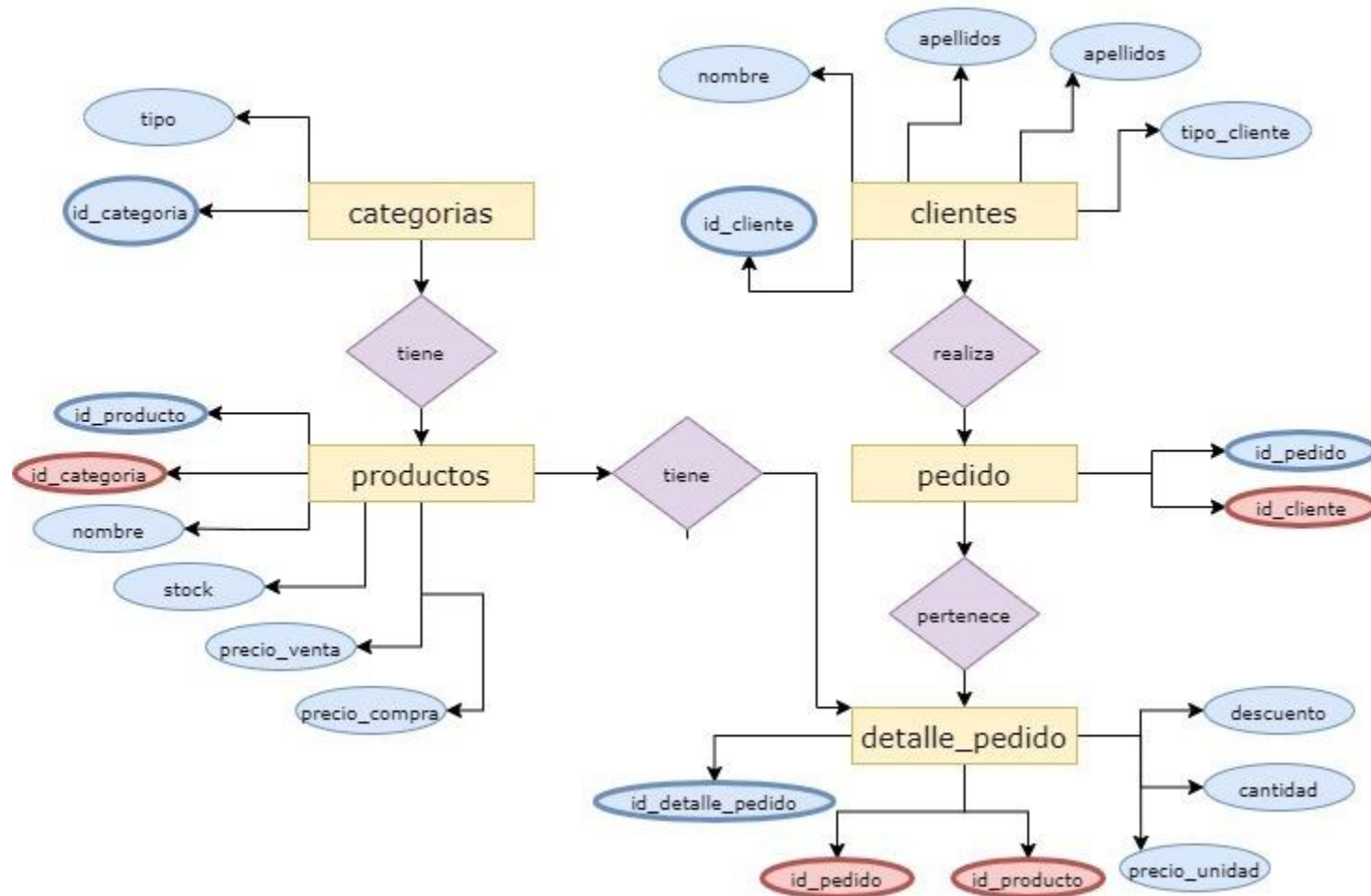
id_cliente



1. Diseño de base de datos.

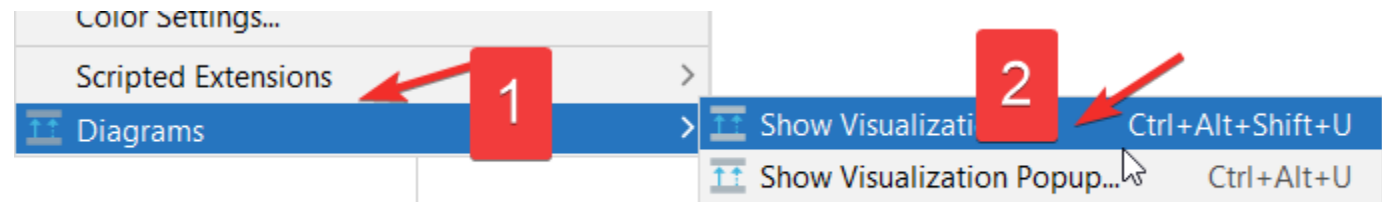
- 1.1. Dado el detalle explicado en la parte inicial de este documento debería generar el modelo entidad relación.
- Para poder generar el diagrama **entidad relación**, podría utilizar la plataforma [Diagrams](#)

Base de Datos Pedidos - Entidad Relación



1.2. Después de generar el **modelo lógico** de la base de datos. El mismo debería quedar similar a lo siguiente:

- Utilizar Datagrip para poder generar el diagrama





1.3. Agregar los siguientes registros a las tablas

categorias	
id_categoria	tipo
1	electrodomesticos
2	juguetes
3	verduras

id_producto	nombre	stock	precio_venta	precio_compra	categoria
1	refrigerador	15	1500	1000	1
2	microonda	4	800	500	1
3	los vengadores ...	2	2500	1700	2

id_cliente	nombres	apellidos	direccion	tipo_cliente
1	nombre_cliente1	apellidos_cli...	6 de agosto e...	GOLD
2	nombre_cliente2	apellidos_cli...	Plaza abaroa	VIP
3	nombre_cliente3	apellidos_cli...	Plaza del est...	NORMAL
4	nombre_cliente4	apellidos_cli...	Teatro al air...	NORMAL

pedido					
	id_pedido		id_cliente		
1	1		1		
2	2		2		

detalle_pedido					
id_detalle_pedido	id_pedido	id_producto	precio_unidad	cantidad	descuento
1	1	1	1000	2	0
2	1	2	800	1	0
3	2	2	800	1	0

2. Manejo de conceptos

2.1. Que es el modelo **entidad relación**.

R.- Es un modelo de datos que fue desarrollado para facilitar el diseño de las bases de datos, ya que permite la creación de un esquema que representa la estructura global lógica de la base de datos.

2.2. Que es el **modelo lógico** en bases de datos relacionales.

R.- Se trata de una estructura formada por filas y columnas que almacena los datos referentes a una determinada entidad o relación del mundo real.

2.3. Describe y menciona que formas(**shapes**) se utiliza para graficar un modelo entidad relación.

2.4. Qué es una **función de agregación**.

R.- Una función de agregación es una función que resume las filas de un grupo en un solo valor. COUNT, MIN y MAX son ejemplos de funciones de agregación.

2.5. Muestre ejemplo del uso de 2 funciones de **agregación**.

```
R.- SELECT COUNT(1)
FROM detalle_pedido as dp
INNER JOIN productos as pro ON pro.id_producto = dp.id_producto
WHERE dp.id_pedido = 2
```



```

SELECT SUM(dp.precio_unidad)
FROM detalle_pedido as dp
INNER JOIN productos as pro ON pro.id_producto = dp.id_producto
GROUP BY dp.id_pedido

```

Output x SUM(dp.precio_unidad):numeric x

2 rows v	
<anonymous>	
1	1800.00
2	800.00

2.6. Muestre un ejemplo del uso de **JOINS**.

```

3. SELECT pro.nombre
FROM detalle_pedido as dp
INNER JOIN productos as pro ON pro.id_producto = dp.id_producto
WHERE dp.id_pedido = 1

```

3.1. Qué es **SQL** y **NoSQL**.

R.- **SQL** es un lenguaje de computación para trabajar con conjuntos de datos y las relaciones entre ellos.

Las bases de datos NoSQL utilizan una variedad de modelos de datos para acceder y administrar datos. Estos tipos de bases de datos están optimizados específicamente para aplicaciones que requieren grandes volúmenes de datos, baja latencia y modelos de datos flexibles, lo que se logra mediante la flexibilización de algunas de las restricciones de coherencia de datos en otras bases de datos.

3.2. A que se refiere cuando se habla de **ISO**, que es una **ISO**.

R.- Son un conjunto de estándares con reconocimiento internacional que fueron creados con el objetivo de ayudar a las empresas a establecer unos niveles de homogeneidad en relación con la gestión

3.3. Quien creo el modelo entidad relación o más conocido como **E-R**

R.- Fue definido por Peter Chen en 1976.

3.4. Crear una función que permita sumar 3 números.

```
CREATE OR ALTER function sumarNumero (  
@num1 int,  
@num2 int,  
@num3 int)  
RETURNS integer  
AS  
BEGIN  
    DECLARE @resultado int = 0;  
  
    set    @resultado = @num1 + @num2 +@num3;  
  
    RETURN @resultado;  
end;  
  
SELECT dbo.sumarNumero(2,10,8);
```

4. Manejo de consultas

4.1. Mostrar los productos(**Nombre y stock**) con stock mayor igual a 10.

The screenshot displays the Microsoft SQL Server Enterprise Manager interface. The left pane shows the database structure for 'procesual', including tables like 'categorias', 'clientes', 'detalle_pedido', 'pedido', and 'productos'. The central pane contains a SQL script with several INSERT statements followed by three queries labeled 3.1, 3.2, and 3.3. Query 3.1 is highlighted and shows the result of a SELECT statement filtering for products with a stock of 10 or more. The bottom pane shows the 'Output' window with the results of the query 'procesual.dbo.productos', displaying a single row for a refrigerator with a stock of 15.

```
54 insert into pedido(id_cliente) values(1);
55 insert into pedido(id_cliente) values(2);
56
57
58 insert into detalle_pedido(id_pedido, id_producto, precio_unidad, cantidad, descuento) values(1,1,1000,2,0);
59 insert into detalle_pedido(id_pedido, id_producto, precio_unidad, cantidad, descuento) values(1,2,800,1,0);
60 insert into detalle_pedido(id_pedido, id_producto, precio_unidad, cantidad, descuento) values(2,2,800,1,10);
61
62 /* Consultas */
63 /* 3.1 */
64 SELECT nombre, stock
65 FROM productos
66 WHERE stock >= 10
67 /* 3.2 */
68 SELECT pro.nombre, cat.nombre
69 FROM productos as pro
70 INNER JOIN categorias as cat ON cat.categoria_id = pro.categoria
71 WHERE cat.tipo = "electrodomesticos"
72 /* 3.3 */
73 SELECT pro.nombre
74 FROM detalle_pedido as dp
75 INNER JOIN producto as pro ON pro.id_producto = dp.id_producto
76 WHERE dp.id_pedido = 1
```

nombre	stock
1 refrigerador	15

- 4.2. Mostrar el nombre del producto y la categoría de los productos pertenecen a la categoría de "electrodomesticos".

The screenshot displays the Microsoft SQL Server Enterprise Manager interface. On the left, the 'Server Objects' tree shows the 'dbo' schema with tables including 'categorias', 'clientes', 'detalle_pedido', 'pedido', and 'productos'. The main pane shows a SQL script with the following queries:

```
54 insert into pedido(id_cliente) values(1);
55 insert into pedido(id_cliente) values(2);
56
57
58 insert into detalle_pedido(id_pedido, id_producto, precio_unidad, cantidad, descuento) values(1,1,1000,2,0);
59 insert into detalle_pedido(id_pedido, id_producto, precio_unidad, cantidad, descuento) values(1,2,800,1,0);
60 insert into detalle_pedido(id_pedido, id_producto, precio_unidad, cantidad, descuento) values(2,2,800,1,10);
61
62 /* Consultas */
63 /* 3.1 */
64 SELECT nombre, stock
65 FROM productos
66 WHERE stock >= 10
67
68 /* 3.2 */
69 SELECT pro.nombre, cat.tipo
70 FROM productos as pro
71 INNER JOIN categorias as cat ON cat.categoria_id = pro.categoria
72 WHERE cat.tipo = 'electrodomesticos'
73
74 /* 3.3 */
75 SELECT pro.nombre
76 FROM detalle_pedido as dp
77 INNER JOIN producto as pro ON pro.id_producto = dp.id_producto
78 WHERE dp.id_pedido = 1
```

The query on line 69 is highlighted. Below the script, the 'Output' pane shows the results of the query:

nombre	tipo
1 refrigerador	electrodomesticos
2 microonda	electrodomesticos

- 4.3. Que **productos(nombre)** tiene el pedido con id igual a 1.

The screenshot displays the Microsoft SQL Server Enterprise Manager interface. On the left, the 'Database' pane shows the 'procesual' database with tables 'categorias', 'clientes', 'detalle_pedido', 'pedido', and 'productos'. The 'Services' pane at the bottom shows the 'console_2' service running. The main 'Query Editor' window contains the following SQL script:

```
57  
58 insert into detalle_pedido(id_pedido, id_producto, precio_unidad, cantidad, descuento) values(1,1,1000,2,0);  
59 insert into detalle_pedido(id_pedido, id_producto, precio_unidad, cantidad, descuento) values(1,2,800,1,0);  
60 insert into detalle_pedido(id_pedido, id_producto, precio_unidad, cantidad, descuento) values(2,2,800,1,10);  
61  
62 /* Consultas */  
63 /* 3.1 */  
64 SELECT nombre, stock  
65 FROM productos  
66 WHERE stock >= 10  
67 /* 3.2 */  
68 SELECT pro.nombre, cat.tipo  
69 FROM productos as pro  
70 INNER JOIN categorias as cat ON cat.categoria_id = pro.categoria  
71 WHERE cat.tipo = 'electrodomesticos'  
72 /* 3.3 */  
73 SELECT pro.nombre  
74 FROM detalle_pedido as dp  
75 INNER JOIN productos as pro ON pro.id_producto = dp.id_producto  
76 WHERE dp.id_pedido = 1  
77 /* 3.4 */  
78 SELECT COUNT(1)  
79 FROM detalle_pedido as dp
```

The 'Output' pane at the bottom right shows the results of the query for 'procesual.dbo.productos':

nombre
1 refrigerador
2 microonda

4.4. Cuantos(count) productos tiene el pedido con id igual a = 2.

The screenshot displays the Microsoft SQL Server Enterprise Manager interface. The left pane shows the server structure for 'Microsoft SQL Server - @localhost', with the 'procesual' database selected. The central pane contains a SQL script with the following code:

```

66 WHERE stock >= 10
67 /* 3.2 */
68 SELECT pro.nombre, cat.tipo
69 FROM productos as pro
70 INNER JOIN categorias as cat ON cat.categoria_id = pro.categoria
71 WHERE cat.tipo = 'electrodomesticos'
72 /* 3.3 */
73 SELECT pro.nombre
74 FROM detalle_pedido as dp
75 INNER JOIN productos as pro ON pro.id_producto = dp.id_producto
76 WHERE dp.id_pedido = 1
77 /* 3.4 */
78 SELECT COUNT(1)
79 FROM detalle_pedido as dp
80 INNER JOIN productos as pro ON pro.id_producto = dp.id_producto
81 WHERE dp.id_pedido = 2
82 /* 3.5 */
83 CREATE OR ALTER function sumarNumero(@num1 int,@num2 int,@num3 int)
84 RETURNS VARCHAR(100)
85 AS
86 BEGIN
87     DECLARE @resultado int = 0;
88

```

The script is executed, and the results are shown in the bottom pane. The 'Output' tab displays the results of the 'COUNT(1):int' query, showing a single row with the value 1.

Output
1

4.5. Crear una función que permita sumar 3 números.

The screenshot displays the SQL Server Enterprise Manager interface. The left pane shows the database structure for 'Microsoft SQL Server - @localhost', with the 'procesual' database selected. The central pane shows a query editor with the following SQL code:

```
78 SELECT COUNT(1)
79 FROM detalle_pedido as dp
80 INNER JOIN productos as pro ON pro.id_producto = dp.id_producto
81 WHERE dp.id_pedido = 2
82 /* 3.5 */
83 CREATE OR ALTER function sumarNumero (
84 @num1 int,
85 @num2 int,
86 @num3 int)
87 RETURNS integer
88 AS
89 BEGIN
90     DECLARE @resultado int = 0;
91
92     set @resultado = @num1 + @num2 + @num3;
93
94     RETURN @resultado;
95 end;
96
97 SELECT dbo.sumarNumero( @num1: 2, @num2: 10, @num3: 8);
98 /* 3.6 */
99 CREATE OR ALTER function restarNumero(@num1 int,@num2 int,@num3 int)
100 RETURNS VARCHAR(100)
```

The right pane shows the results of the query, displaying a single row with the value 20.

The bottom pane shows the 'Services' section, listing the following services:

- Microsoft SQL Server - @localhost
 - console
 - console_1
 - console_2 210 ms

4.6. Crear una función que permita restar 3 números.

The screenshot displays the SQL Server Enterprise Manager (left pane) and the SQL Server Enterprise Console (right pane). The left pane shows the database structure for 'Microsoft SQL Server - @localhost', including databases like 'evaluacion', 'evaluacion2', 'master', and 'procesual'. The 'procesual' database is expanded, showing the 'dbo' schema with tables and routines. The right pane shows the SQL Server Enterprise Console with a script editor and an output window.

The script editor contains the following SQL code:

```

93
94     RETURN @resultado;
95 end;
96
97 SELECT dbo.sumarNumero( @num1: 2, @num2: 10, @num3: 8);
98 /* 3.6 */
99 CREATE OR ALTER function restarNumero(@num1 int,@num2 int,@num3 int)
100 RETURNS VARCHAR(100)
101 AS
102 BEGIN
103     DECLARE @resultado int = 0;
104
105     set @resultado = @num1 - @num2 - @num3;
106
107     RETURN @resultado;
108 end;
109
110 ✓ SELECT dbo.restarNumero( @num1: 20, @num2: 4, @num3: 7);
111 /* 3.7 */
112 CREATE OR ALTER function unificado(@tipo int,@num1 int,@num2 int,@num3 int)
113 RETURNS VARCHAR(100)
114 AS
115 BEGIN

```

The output window shows the result of the execution of the `SELECT` statement:

```

Output x  dbo.restarNumero(20,4,7):varchar(100) x
1 9

```

4.7. Cómo unificaría en una sola función el ejercicio 3.5 y 3.7(los dos anteriores).

The screenshot displays the SQL Server Enterprise Manager interface. The left pane shows the database structure for 'Microsoft SQL Server - @localhost', with the 'procesual' database selected. The central pane shows a T-SQL script with the following content:

```
108 end;
109
110 SELECT dbo.restarNumero( @num1: 20, @num2: 4, @num3: 7);
111
112 /* 3.7 */
113 CREATE OR ALTER function unificado(@tipo int,@num1 int,@num2 int,@num3 int)
114 RETURNS VARCHAR(100)
115 AS
116 BEGIN
117     DECLARE @resultado int = 0;
118
119     IF @tipo = 1
120         set @resultado = @num1 + @num2 +@num3;
121     ELSE
122         set @resultado = @num1 - @num2 - @num3;
123
124     RETURN @resultado;
125 end;
126
127 SELECT dbo.unificado( @tipo: 1, @num1: 20, @num2: 4, @num3: 7);
```

The bottom pane shows the 'Services' section with a list of console windows. The 'console_2' window is selected, showing the output of the function call:

Output
1 31

Detalles sobre la entrega de la tarea

1. Crear un documento(archivo PPT) por cada pregunta adjuntar una respuesta(texto explicativo) y una captura de pantalla(Screenshot). Es decir debe crear una presentación power point.
2. El documento generado en el paso 1 convertir en un archivo PDF.
3. Crear un único vídeo explicando todos los pasos realizados para resolver esta actividad.
4. El documento PDF y el video tienen que estar en una carpeta de nombre **procesual** en github (**hito3/procesual**)
5. En la plataforma Moodle solo subir la carátula.
 - a. En la parte inferior agregar **URL de github:** {{mi_url_de_github}}