

Northwind

Table of contents

1 Analisis de la base de datos Northwind	3
2 Modificación de la Tabla Products: Atributos Dinámicos con JSON	4
3 Relleno de datos JSON	6
4 Consultas a la columna JSON	8
5 Ampliacion de Base de datos con Nueva Tabla	9
6 Creacion de vistas	10

1 Analisis de la base de datos Northwind

Introducción La base de datos Northwind es un conjunto de datos para prácticas de SQL, que simula una empresa de venta de productos alimenticios. Contiene información sobre clientes, empleados, pedidos, productos y proveedores. En este miniproyecto, realizaremos un análisis completo que incluye:

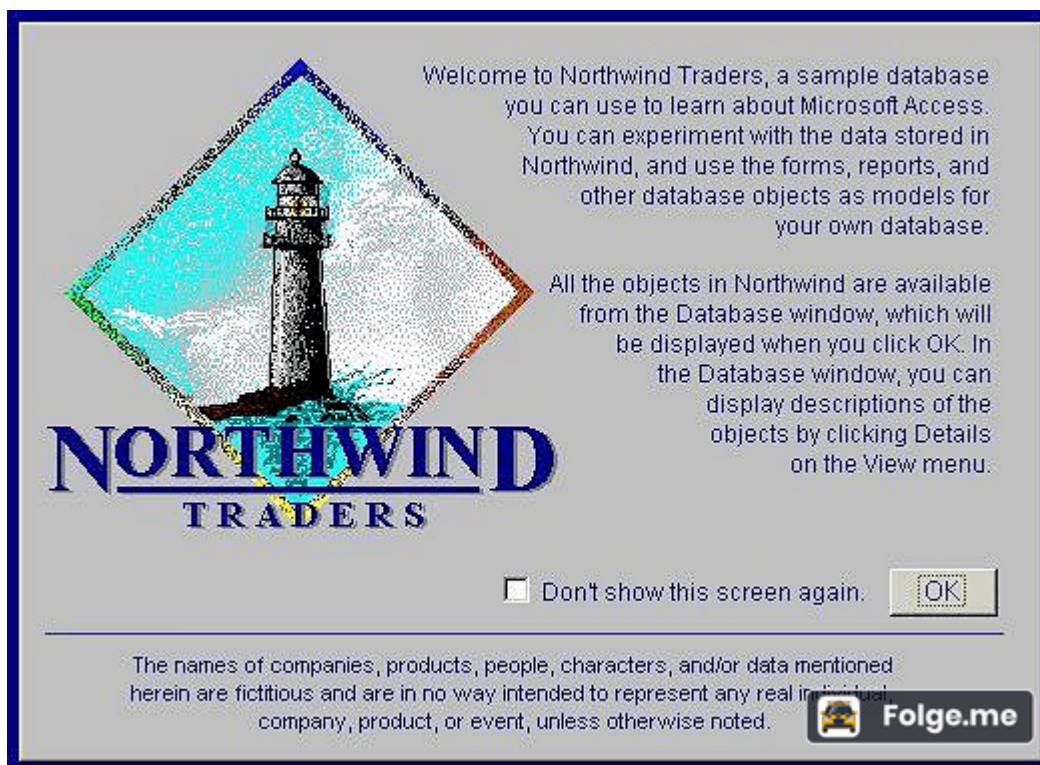
- **Creación de un diagrama entidad-relación**

- **Consultas SQL avanzadas**

- **Procedimientos almacenados**

- **Funciones Triggers**

- **Visualización de datos**



2 Modificación de la Tabla Products: Atributos Dinámicos con JSON

Añadir columna JSON a la tabla Products:

- Añadir columna para atributos dinámicos

```
-- Añadir columna para atributos dinámicos
ALTER TABLE Products
ADD COLUMN características_json JSON;
```

- Verificar la estructura actualizada

```
DESCRIBE Products;
\d Productos;
```

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'Tables (14)' folder is expanded, and the 'Columns (11)' folder for the 'products' table is selected. The columns listed are: product_id, product_name, supplier_id, category_id, quantity_per_unit, unit_price, units_in_stock, units_on_order, reorder_level, discontinued, and **caracteristicas_json**. A red circle with the number '2' highlights the 'caracteristicas_json' column.

On the right, the 'Query' tab is active. A red circle with the number '1' highlights the first line of the SQL query:

```
-- Añadir columna para atributos dinámicos
ALTER TABLE Products
ADD COLUMN caracteristicas_json JSON;
```

Below the query editor, the 'Messages' tab is selected, showing the execution result:

```
ALTER TABLE

Query returned successfully in 69 msec.
```

A 'Folge.me' logo is visible in the bottom right corner of the interface.

3 Relleno de datos JSON

Actualizar productos existentes con datos

```
UPDATE Products SET caracteristicas_json = '  
{  
  "categoria": "Bebidas",  
  "subcategoria": "Calientes",  
  "especificaciones": {  
    "tamano": "500 ml",  
    "material": "Vidrio",  
    "ingredientes": ["T  ", "Especias"]  
  }  
}'  
WHERE product_id = 1; -- Chai
```

```
UPDATE Products SET caracteristicas_json = '  
{  
  "categoria": "Condimentos",  
  "subcategoria": "Salsas",  
  "especificaciones": {  
    "peso": "200 g",  
    "origen": "Asia",  
    "nivel_picante": 3  
  }  
}'  
WHERE product_id = 3; -- Aniseed Syrup
```

```
UPDATE Products SET caracteristicas_json = '  
{  
  "categoria": "Bebidas",  
  "subcategoria": "Refrigerados",  
  "especificaciones": {  
    "tamano": "1 litro",  
    "azucar": 25,  
    "conservantes": false  
  }  
}'  
WHERE product_id = 24; -- Guaran   Fant  stica
```

Verificar datos actualizados

```
SELECT product_id , product_name, caracteristicas_json
FROM Products
WHERE caracteristicas_json IS NOT NULL;
```

The image shows two side-by-side PostgreSQL query editors. The left editor contains three UPDATE queries for products 1, 3, and 24, each setting a JSON field 'caracteristicas_json'. The right editor contains a SELECT query for products where 'caracteristicas_json' is not null. Below the queries, the 'Data Output' tab shows the results of the SELECT query, displaying three rows of product data.

Left Editor Query:

```
1 UPDATE Products SET caracteristicas_json = '
2 {
3   "categoria": "Bebidas",
4   "subcategoria": "Calientes",
5   "especificaciones": {
6     "tamano": "500 mL",
7     "material": "Vidrio",
8     "ingredientes": ["T ", "Especias"]
9   }
10 }'
11 WHERE product_id = 1; -- Chai
12
13 UPDATE Products SET caracteristicas_json = '
14 {
15   "categoria": "Condimentos",
16   "subcategoria": "Salsas",
17   "especificaciones": {
18     "peso": "200 g",
19     "origen": "Asia",
20     "nivel_picante": 3
21   }
22 }'
23 WHERE product_id = 3; -- Aniseed Syrup
24
25 UPDATE Products SET caracteristicas_json = '
26 {
27   "categoria": "Bebidas",
28   "subcategoria": "Refrigerados",
29   "especificaciones": {
30     "tamano": "1 litro",
31     "azucar": 25,
32     "conservantes": false
33   }
34 }'
35 WHERE product_id = 24; -- Guaran  Fant stica
```

Right Editor Query:

```
1 SELECT product_id, product_name, caracteristicas_json
2 FROM Products
3 WHERE caracteristicas_json IS NOT NULL;
```

Data Output (Right Editor):

product_id [PK] smallint	product_name character varying (40)	caracteristicas_json json
1	Chai	
2	Aniseed Syrup	
3	Guaran� Fant�stica	

Showing rows: 1 to 3 | Page No: 1 of 1

Folge.me

4 Consultas a la columna JSON

```
SELECT características_json->>'categoria' AS Categoria,  
características_json->>'subcategoria' AS subcategoria,  
características_json->'especificaciones'->'tamano' AS especificaciones  
FROM Products  
WHERE características_json IS NOT NULL;
```

Query Query History

```
1 SELECT características_json->>'categoria' AS Categoria,  
2 características_json->>'subcategoria' AS subcategoria,  
3 características_json->'especificaciones'->'tamano' AS especificaciones  
4 FROM Products  
5 WHERE características_json IS NOT NULL;
```

Data Output Messages Notifications



Showing rows: 1 to 3 Page No: 1 of 1



	categoria text	subcategoria text	especificaciones json
1	Bebidas	Calientes	"500 ml"
2	Condimentos	Salsas	[null]
3	Bebidas	Refrigerados	"1 litro"



5 Ampliacion de Base de datos con Nueva Tabla

Descuentos por volumen

```
CREATE TABLE volume_discounts (
  DiscountID INT AUTO_INCREMENT PRIMARY KEY,
  ProductID INT NOT NULL,
  MinQuantity INT NOT NULL,
  Discount DECIMAL(4,2) NOT NULL CHECK (Discount BETWEEN 0 AND 1),
  FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

Query History

```
CREATE TABLE volume_discounts (
  discount_id INT PRIMARY KEY,
  product_id INT NOT NULL,
  MinQuantity INT NOT NULL,
  Discount DECIMAL(4,2) NOT NULL CHECK (Discount BETWEEN 0 AND 1),
  FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
INSERT INTO volume_discounts (discount_id, product_id, MinQuantity, Discount) VALUES
-- Descuentos para Chai (ProductID = 1)
(1,1, 10, 0.05), -- 5% de descuento para compras de 10+ unidades
(2,1, 20, 0.10), -- 10% de descuento para compras de 20+ unidades
(3,1, 50, 0.15), -- 15% de descuento para compras de 50+ unidades

-- Descuentos para Chang (ProductID = 2)
(4,2, 5, 0.07), -- 7% de descuento para compras de 5+ unidades
(5,2, 15, 0.12), -- 12% de descuento para compras de 15+ unidades

-- Descuentos para Sir Rodney's Marmalade (ProductID = 20)
(6,20, 8, 0.10), -- 10% de descuento para compras de 8+ unidades
(7,20, 30, 0.25), -- 25% de descuento para compras de 30+ unidades

-- Descuentos para Louisiana Hot Spiced Okra (ProductID = 46)
(8,46, 12, 0.15), -- 15% de descuento para compras de 12+ unidades
(9,46, 24, 0.20), -- 20% de descuento para compras de 24+ unidades

-- Descuentos para Scottish Longbreads (ProductID = 59)
(10,59, 6, 0.05), -- 5% de descuento para compras de 6+ unidades
(11,59, 12, 0.12), -- 12% de descuento para compras de 12+ unidades
(12,59, 36, 0.30); -- 30% de descuento para compras de 36+ unidades
```



6 Creacion de vistas

Analisis de volumen de descuento

```
CREATE VIEW analisis_volumen_descuento AS
SELECT
    vd.discount_id,
    vd.product_id,
    p.product_name,
    p.unit_price,
    vd.MinQuantity,
    vd.Discount AS DiscountRate,
    -- Precio con descuento unitario
    ROUND((p.unit_price * (1 - vd.Discount))::NUMERIC, 2) AS DiscountedPrice,
    -- Ahorro por unidad
    ROUND((p.unit_price * vd.Discount)::NUMERIC, 2) AS SavingsPerUnit,
    -- Precio mínimo para el lote
    (vd.MinQuantity * p.unit_price) AS OriginalBatchPrice,
    -- Precio con descuento para el lote
    ROUND((vd.MinQuantity * p.unit_price * (1 - vd.Discount))::NUMERIC, 2) AS DiscountedBatchPrice,
    -- Ahorro total en el lote
    ROUND((vd.MinQuantity * p.unit_price * vd.Discount)::NUMERIC, 2) AS TotalSavings
FROM volume_discounts vd
JOIN Products p ON vd.product_id = p.product_id;
```

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Query History' for the view creation. The right pane shows the 'Query' results, which include a list of discounts for various products. The bottom pane shows the 'Data Output' for the view, displaying a table with columns: discount_id, product_name, unit_price, minquantity, discountrate, discountedprice, savingsperunit, originalbatchprice, discountedbatchprice, and totalsavings. The table contains 10 rows of data.

discount_id	product_name	unit_price	minquantity	discountrate	discountedprice	savingsperunit	originalbatchprice	discountedbatchprice	totalsavings
1	Chai	18	10	0.05	17.10	0.90	180	171.00	18.00
1	Chai	18	20	0.10	16.20	1.80	360	324.00	36.00
1	Chai	18	50	0.15	15.30	2.70	900	765.00	135.00
2	Chang	19	5	0.07	17.67	1.33	95	88.35	16.65
2	Chang	19	15	0.12	16.72	2.28	285	250.80	34.20
20	Sir Rodney's Marmalade	81	8	0.10	72.90	8.10	648	568.20	80.00
20	Sir Rodney's Marmalade	81	30	0.25	60.75	20.25	2430	1822.50	607.50
46	Spegesid	12	12	0.15	10.20	1.80	144	122.40	21.60
46	Spegesid	12	24	0.20	9.60	2.40	288	230.40	57.60
59	Raclette Courdavault	55	6	0.05	52.25	2.75	330	313.50	16.50
59	Raclette Courdavault	55	12	0.12	48.40	6.60	660	580.80	79.20
59	Raclette Courdavault	55	36	0.30	38.50	16.50	1980	1386.00	594.00