

Link al programa donde se encuentra el diagrama del modelo relacional:
<https://drive.google.com/file/d/1OblgmOTQw36bdGJGI6hgoYnP4vbxxi1l/view> o
 revisar también el pdf disponible en el repositorio del taller.

2. Diccionario de datos.

Evidencias del diccionario de datos

Administrador					
Atributo	PK	FK	Tipo de dato	Longitud del dato	Explicación
Id_admin	SI	NO	int	N.A	Identificador único de cada administrador
Nombre	NO	SI	varchar	200	Almacena el nombre que ingresará un administrador
Vendedor					
Atributo	PK	FK	Tipo de dato	Longitud del dato	Explicación
Id_vendedor	SI	NO	int	N.A	Identificador único de cada vendedor
Nombre	NO	NO	varchar	100	Almacena el nombre que ingresará un vendedor
Calificacion	NO	NO	varchar	20	Almacena la calificación dada por un comprador a un producto
Id_admin	NO	SI	int	N.A	Referencia a la tabla de Administrador
Id_producto	NO	SI	int	N.A	Referencia a la tabla de Producto
Comprador					
Atributo	PK	FK	Tipo de dato	Longitud del dato	Explicación
Id_comprador	SI	NO	int	N.A	Identificador único de cada comprador
Tipo_identificacion	NO	NO	varchar	20	Almacena el tipo de identificación que ingresará un comprador
Num_identificacion	NO	NO	int	N.A	Almacena el número de identificación que ingresará un comprador
Nombre	NO	NO	varchar	200	Almacena el nombre que ingresará un comprador
Nombre_usuario	NO	NO	varchar	200	Almacena el nombre de usuario(nickname) que ingresará un administrador
Contraseña	NO	NO	varchar	200	Almacena la contraseña que creará un comprador
Correo	NO	NO	varchar	200	Almacena el correo electrónico que ingresará un comprador
Ciudad	NO	NO	varchar	200	Almacena la ciudad de donde es un comprador
Dirección	NO	NO	varchar	200	Almacena la dirección de donde vive un comprador
Id_vendedor	NO	SI	int	N.A	Referencia a la tabla de Vendedor
Producto					
Atributo	PK	FK	Tipo de dato	Longitud del dato	Explicación
Id_producto	SI	NO	int	N.A	Identificador único de cada producto
Nombre	NO	NO	varchar	100	Almacena el nombre de cada producto que se ingrese
Categoría	NO	NO	varchar	20	Almacena la categoría a la que pertenece cada producto ingresado
Marca	NO	NO	varchar	30	Almacena la marca a la que pertenece cada producto ingresado
Stock	NO	NO	int	N.A	Almacena la cantidad de unidades disponibles de cada producto ingresado
Descripcion	NO	NO	varchar	200	Almacena la descripción de cada producto ingresado
Precios	NO	NO	varchar	100	Almacena el precio de cada producto ingresado
Id_carrito	NO	SI	int	N.A	Referencia a la tabla de Carrito
Compra					
Atributo	PK	FK	Tipo de dato	Longitud del dato	Explicación
Id_compra	SI	NO	int	N.A	Identificador único de cada compra
Estado_de_pago	NO	NO	varchar	20	Indica el estado en el que se encuentra el pago del producto seleccionado
Estado_compra	NO	NO	varchar	20	Indica el estado en el que se encuentra la compra del producto seleccionado
Id_comprador	NO	SI	int	N.A	Referencia a la tabla de Comprador
Id_producto	NO	SI	int	N.A	Referencia a la tabla de Producto
Id_envio	NO	SI	int	N.A	Referencia a la tabla de Envio
Carrito					
Atributo	PK	FK	Tipo de dato	Longitud del dato	Explicación
Id_carrito	SI	NO	int	N.A	Identificador único del carrito

Bases de datos 2

Docente: Fabian Camilo Peña Lozano

Juan Felipe Organista Sanchez

Diana Marcela Rios Gaviria

Jordan William Soto Diaz

Universidad El Bosque

V Semestre



Envio					
Atributo	PK	FK	Tipo de dato	Longitud del dato	Explicación
Id_envio	SI	NO	int	N.A	Identificador único de cada envío
Estado	NO	NO	varchar	50	Indica el estado en el que se encuentra el envío del producto seleccionado
Guia	NO	NO	varchar	70	Indica el número de guia con la que se envió el producto seleccionado

CuentaDePago					
Atributo	PK	FK	Tipo de dato	Longitud del dato	Explicacion
Id_cuenta_pago	SI	NO	int	N.A	Identificador único de cada cuenta de pago
Dinero_en_cuenta	NO	NO	numeric	20	Indica monto de dinero disponible en una cuenta
Id_tarjeta	NO	SI	int	N.A	Referencia a la tabla de Tarjeta
Id_comprador	NO	SI	int	N.A	Referencia a la tabla de Comprador

Tarjeta					
Atributo	PK	FK	Tipo de dato	Longitud del dato	Explicacion
Id_tarjeta	SI	NO	int	N.A	Identificador único de una tarjeta bancaria
Tipo_tarjeta	NO	NO	varchar	100	Almacena el tipo de tarjeta (debito, crédito(visa,mastercard,etc))
Numero_tarjeta	NO	NO	varchar	100	Almacena el número de tarjeta (debito, crédito(visa,mastercard,etc))
Fecha_ex	NO	NO	date	N.A	Almacena la fecha en la expira la tarjeta
Cvc	NO	NO	varchar	10	Almacena el código cvc de una tarjeta

Vende					
Atributo	PK	FK	Tipo de dato	Longitud del dato	Explicacion
Id_vendedor	SI	SI	int	N.A	Referencia a la tabla de Vendedor(Es tanto una PK como una FK)
Id_producto	SI	SI	int	N.A	Referencia a la tabla de Producto(Es tanto una PK como una FK)

Se adjunta la evidencia de la creación del diccionario de datos para el respectivo modelo relacional diseñado en el numeral anterior, en este diccionario se explica más a fondo las funcionalidad de cada atributo que compone cada tabla, y si el atributo es o no una **PK**(Primary Key), o si es o no es un **FK**(Foreign Key).

Link al documento donde se encuentra el diccionario:

<https://docs.google.com/spreadsheets/d/1ToL12NETjGbtM9oyXYrbnJQnK4AbzmnlIX77Ou4OHEw/edit?usp=sharing> o revisar también el pdf disponible en el repositorio del taller.

3. Diseño de vistas (mín. 2) e índices (mín. 3, adicional a los PK). Evidencia del diseño de índices.

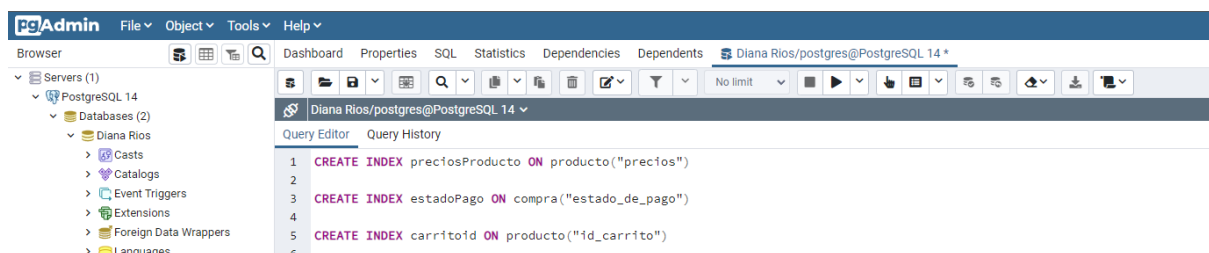


Imagen 1. Código de la creación de los índices.

Evidencia del diseño de vistas.

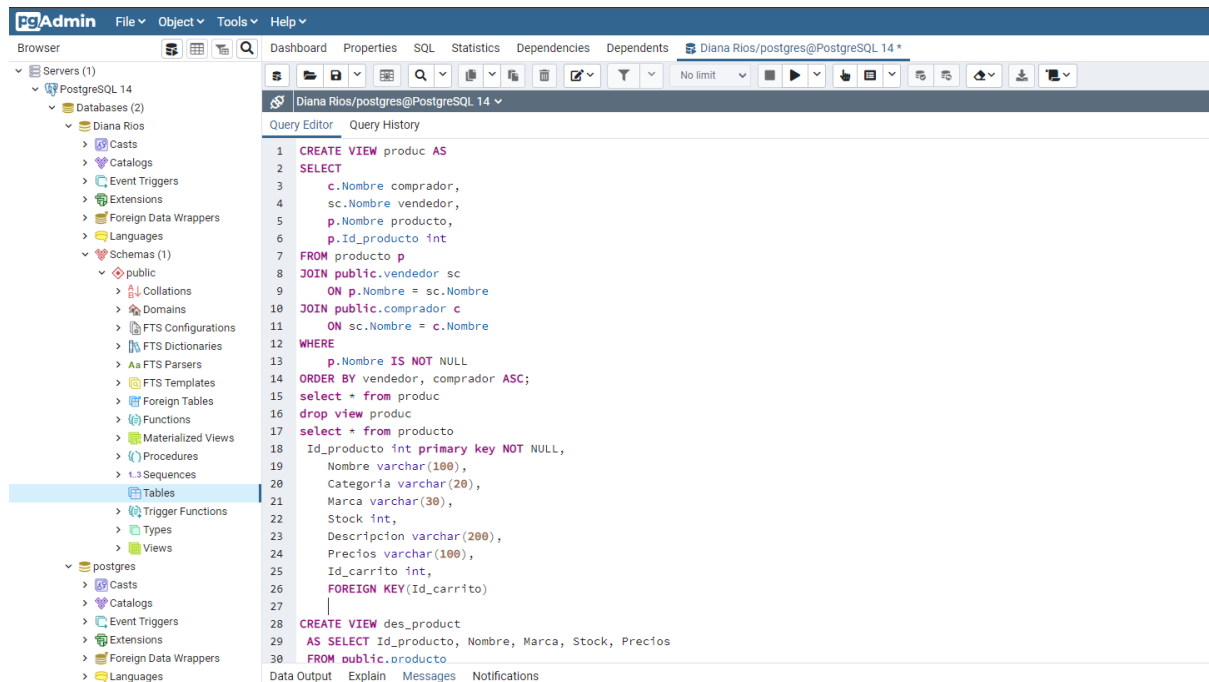


Imagen 2. Código de la creación de las vistas.

Describa textualmente las vistas e índices que deberían extender el modelo de datos.

Descripción del diseño de vistas.

El diseño de las vistas se realiza enfocándose en las posibles consultas más concurridas dentro de la base de datos. De manera que para aprovechar la memoria volátil de la máquina se realizaron dos vistas. la primera se denominó product que pudiese hacerse es una cista donde se cruzan cuatro columnas las cuales son: el nombre del vendedor, el nombre del comprador, el nombre del producto y el id del producto en forma que un usuario con la autorización pueda revisar a detalle quién hizo cual compra y quien lo vendió, esta vista no fue una vista materializada. La segunda vista que se hizo se denominó des_product donde vemos reflejado la reducción de los atributos que componen esta tabla así dando cabida a salvaguardar datos en las tablas que nos puedan ver todos.

Responda las preguntas realizando los supuestos que considere necesarios:

- ¿Las vistas que decide crear a qué requerimiento no funcional obedecen? Seguridad o facilidad de consulta. ¿Deberían ser vistas materializadas? Argumente.

R. Las vistas que se crearon se hicieron pensando en la forma de facilitar la consulta y la visualización de los datos a cualquier usuario con los permisos necesarios en forma que se puedan ver los aspectos más relevantes de las tablas. y no, no considero que deban ser vistas materializadas ya que no será tan necesario tener una duplicidad de los registros, por ese motivo las vistas que se plantearon deben ser vistas no materializadas.

Descripción del diseño de índices.

En esta sección se observa la creación de 3 índices, estos índices se pensaron dadas las posibles búsquedas que pueden tener varios registros en la base de datos.

De esta manera el primer índice es preciosProducto, dicho índice se considera necesario dado que las búsquedas por precios en muchas ocasiones son muy necesarias; de esta forma, estos índices facilitarán la búsqueda puntual de los precios o de rangos.

El segundo índice se llama estadoPago, el cual es índice que se aplica en la columna "estado_de_pago" de la tabla compra. Esto debido a que muchas veces se requiere filtrar por los pagos aprobados, cancelados o rechazados.

Nuestro tercer y último índice se realizó pensando en que dentro de la base de datos pueden existir muchos carritos, ya sea cancelados o generados, de un mismo cliente o no; analizando esto, se decide crear un índice que ayudará en esta búsqueda el cual se llama "carritoid" el cual se implementa en la tabla carrito.

- b.** ¿Cuáles consultas a la base de datos, a partir de los requerimientos dados, pueden optimizarse mediante índices? ¿De qué tipos deben ser dichos índices? Argumente.

R. Dentro de la base de datos que se tiene, son muy necesarias las búsquedas, de manera que, cómo se vio anteriormente, los índices se realizaron pensando en base a las compras. Esto ayudará en las consultas que necesiten información de la compra, ya sea precios, estados o buscar un carrito de los múltiples que pueden haber.

Los índices que se plantean, deben ser índices no agrupados, dado que en la s Tablas, los registros a los cuales hacen referencia los índices, no cuentan con un orden. También deben cumplir con la organización de árbol b+ dado que los índices que se aplicaron deben tener la posibilidad de encontrar rangos con mayor facilidad.

4. Script DDL a ser ejecutado en PostgreSQL. No se acepta el uso de herramientas de generación de código. Cree los índices previamente definidos utilizando los algoritmos dispuestos por el motor de base de datos.

Evidencia del Script DDL

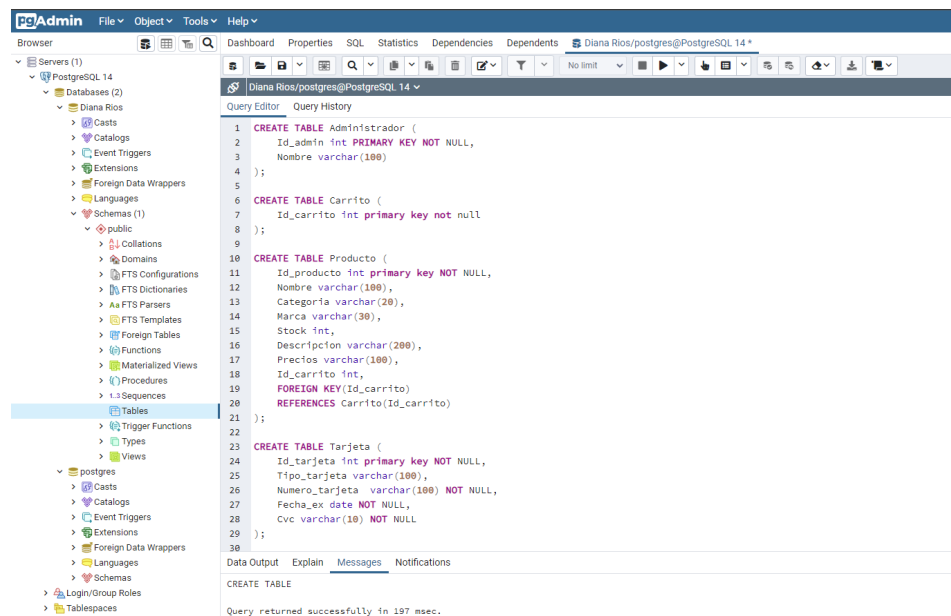


Imagen 3. Código para la creación de las tablas

Se evidencia la creación de las tablas junto con el tiempo que se demoró en el proceso.

Bases de datos 2
Docente: Fabian Camilo Peña Lozano
Juan Felipe Organista Sanchez
Diana Marcela Rios Gaviria
Jordan William Soto Diaz
Universidad El Bosque
V Semestre

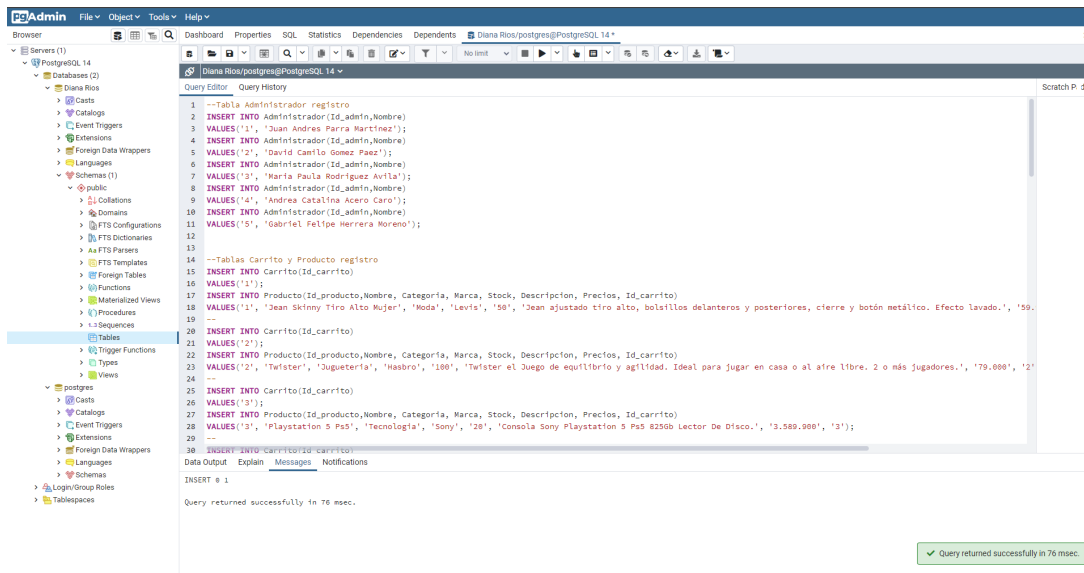


Imagen 4. Código para la creación de los registros de prueba

Se evidencia la alimentación de las tablas con datos de prueba y el tiempo que demoró en el proceso.

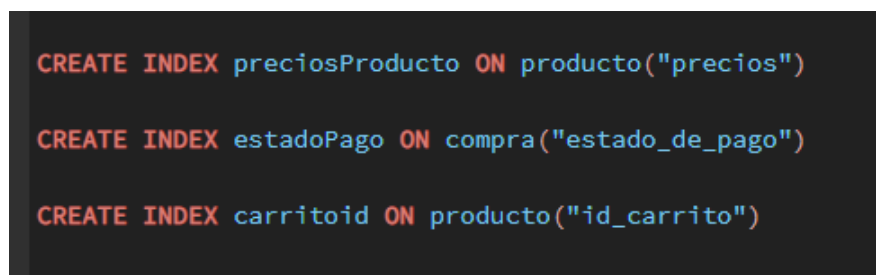


Imagen 5. Creación de los índices

Se evidencia el código mediante el cual se crean los índices de la base de datos.

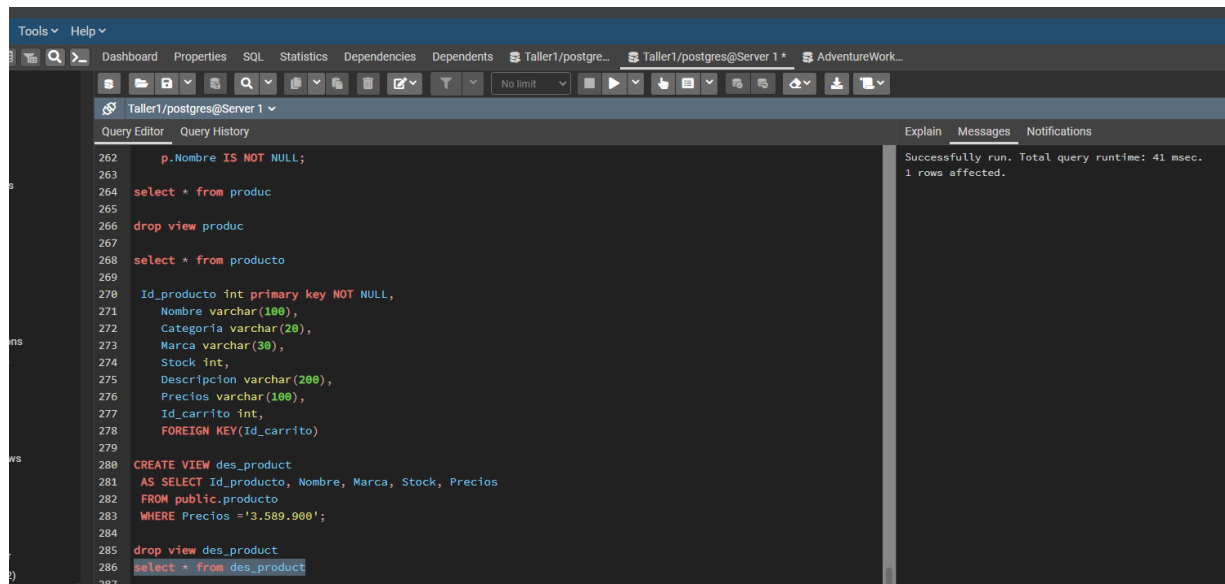


Imagen 6. Creación de las vistas

Se evidencia el código mediante el cual se crean las vistas solicitadas en los requerimientos.

5. Evidencia de la ejecución del script en algún cliente de base de datos como pgAdmin.

Evidencia de la ejecución del script

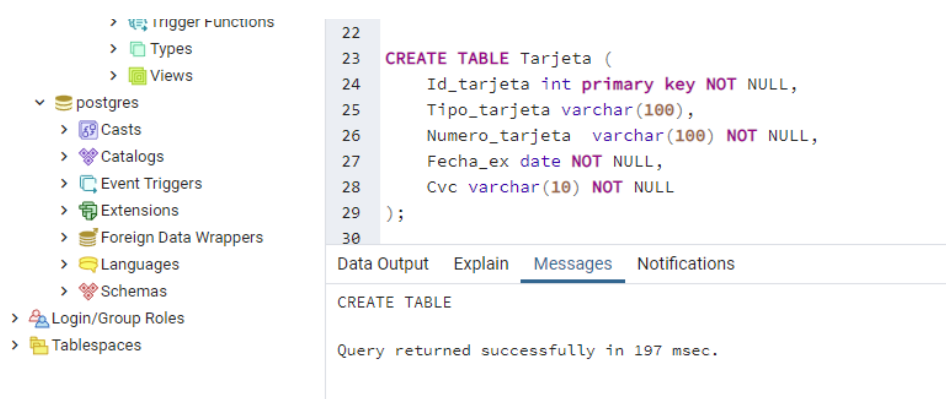


Imagen 7. Tiempo de ejecución para la creación de las tablas

En la creación de las tablas se muestra un tiempo de ejecución de 197 msec.

Bases de datos 2
Docente: Fabian Camilo Peña Lozano
Juan Felipe Organista Sanchez
Diana Marcela Rios Gaviria
Jordan William Soto Diaz
Universidad El Bosque
V Semestre



```
22 INSERT INTO Producto(Id_producto,Nombre, Categoria, Marca, Stock, Descripcion, Precios, Id_carrito)
23 VALUES('2', 'Twister', 'Jugueteria', 'Hasbro', '100', 'Twister el Juego de equilibrio y agilidad. Ideal para jug
24 --
25 INSERT INTO Carrito(Id_carrito)
26 VALUES('3');
27 INSERT INTO Producto(Id_producto,Nombre, Categoria, Marca, Stock, Descripcion, Precios, Id_carrito)
28 VALUES('3', 'Playstation 5 Ps5', 'Tecnologia', 'Sony', '20', 'Consola Sony Playstation 5 Ps5 825Gb Lector De Dis
29 --
30 INSERT INTO Carrito(Id_carrito)
```

Data Output Explain Messages Notifications

INSERT 0 1

Query returned successfully in 76 msec.

Imagen 8. Tiempo de ejecución para la alimentación de las tablas

En la alimentación de las tablas se muestra un tiempo de ejecución de 76 msec.

A Continuación, se presenta un ejemplo en el cual se puede observar la ejecución de un select (Imagen 9) el cual se demora 121msec en ejecutar. Ahora bien, se ejecuta el índice que lleva por nombre preciosProducto (Imagen 10), y con este índice ya implementado, se vuelve a ejecutar el select, y esta vez la misma consulta se demoró 39 msec (Imagen 11).

```
221 VALUES('1','2');
222 INSERT INTO Vende(Id_vendedor, Id_producto)
223 VALUES('2', '4');
224 INSERT INTO Vende(Id_vendedor, Id_producto)
225 VALUES('3', '5');
226 INSERT INTO Vende(Id_vendedor, Id_producto)
227 VALUES('4', '1');
228 INSERT INTO Vende(Id_vendedor, Id_producto)
229 VALUES('5', '3');
230
231
232
233 CREATE INDEX preciosProducto ON producto("precios")
234
235 CREATE INDEX estadoPago ON compra("estado_de_pago")
236
237 CREATE INDEX carritoid ON producto("id_carrito")
238
239
240 Select * from producto
241 where "precios" = '3.589.900'
242
243
244 drop index preciosProducto
245
246 drop index estadoPago
247
248
```

Successfully run. Total query runtime: 121 msec.
1 rows affected.

id_producto	nombre	categoria	marca	stock	descripcion	precios	
[PK] integer	character varying (100)	character varying (20)	character varying (30)	integer	character varying (200)	numeric (15, 2)	
1	3	Playstation 5 Ps5	Tecnologia	Sony	20	Consola Sony Playstation 5 Ps5 825Gb Lector De Disco.	3.589.900

Imagen 9. Ejecución de un select

Bases de datos 2
Docente: Fabian Camilo Peña Lozano
Juan Felipe Organista Sanchez
Diana Marcela Rios Gaviria
Jordan William Soto Diaz
Universidad El Bosque
V Semestre



```

221 VALUES('1','2');
222 INSERT INTO Vende(Id_vendedor, Id_producto)
223 VALUES('2', '4');
224 INSERT INTO Vende(Id_vendedor, Id_producto)
225 VALUES('3','5');
226 INSERT INTO Vende(Id_vendedor, Id_producto)
227 VALUES('4','1');
228 INSERT INTO Vende(Id_vendedor, Id_producto)
229 VALUES('5','3');
230
231
232
233 CREATE INDEX preciosProducto ON producto("precios")
234

```

Query returned successfully in 98 msec.

Imagen 10. Ejecución del índice.

```

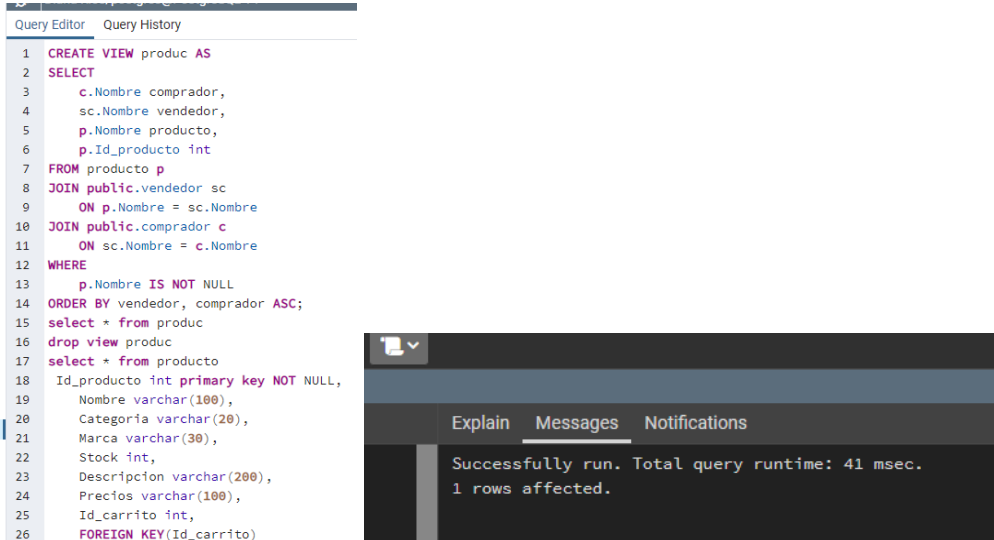
221 VALUES('1','2');
222 INSERT INTO Vende(Id_vendedor, Id_producto)
223 VALUES('2', '4');
224 INSERT INTO Vende(Id_vendedor, Id_producto)
225 VALUES('3','5');
226 INSERT INTO Vende(Id_vendedor, Id_producto)
227 VALUES('4','1');
228 INSERT INTO Vende(Id_vendedor, Id_producto)
229 VALUES('5','3');
230
231
232
233 CREATE INDEX preciosProducto ON producto("precios")
234
235 CREATE INDEX estadoPago ON compra("estado_de_pago")
236
237 CREATE INDEX carritoId ON producto("id_carrito")
238
239
240 Select * from producto
241 where "precios" = '3.589.900'
242
243
244 drop index preciosProducto
245
246 drop index estadoPago
247
248

```

Successfully run. Total query runtime: 39 msec.
1 rows affected.

id_producto (PK) integer	nombre character varying (100)	categoria character varying (20)	marca character varying (30)	stock integer	descripcion character varying (200)	precio character varying (20)
1	Playstation 5 Ps5	Tecnologia	Sony	20	Consola Sony Playstation 5 Ps5 825Gb Lector De Disco.	3.589.900

Imagen 11. Ejecución del select una vez implementado el índice.



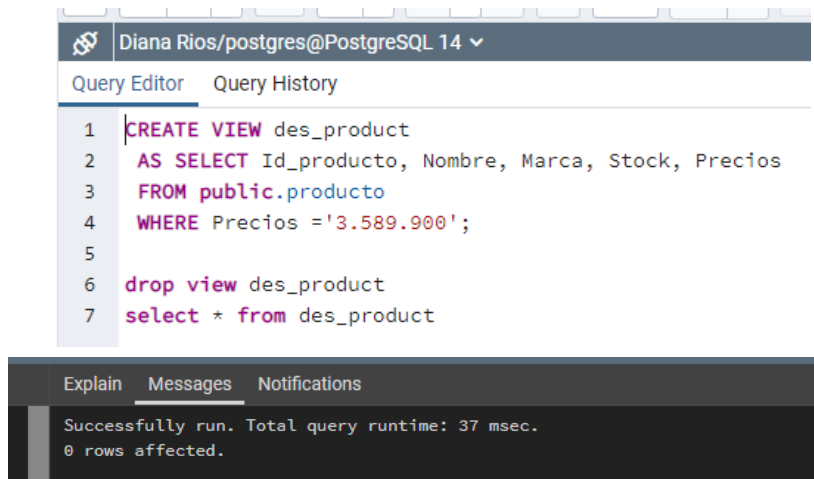
```
1 CREATE VIEW produc AS
2 SELECT
3     c.Nombre comprador,
4     sc.Nombre vendedor,
5     p.Nombre producto,
6     p.Id_producto int
7 FROM producto p
8 JOIN public.vendedor sc
9     ON p.Nombre = sc.Nombre
10 JOIN public.comprador c
11     ON sc.Nombre = c.Nombre
12 WHERE
13     p.Nombre IS NOT NULL
14 ORDER BY vendedor, comprador ASC;
15 select * from produc
16 drop view produc
17 select * from producto
18     Id_producto int primary key NOT NULL,
19     Nombre varchar(100),
20     Categoria varchar(20),
21     Marca varchar(30),
22     Stock int,
23     Descripción varchar(200),
24     Precios varchar(100),
25     Id_carrito int,
26     FOREIGN KEY(Id_carrito)
```

Query Editor Query History

Explain Messages Notifications

Successfully run. Total query runtime: 41 msec.
1 rows affected.

Imagen 12. Ejecución de la primera vista su código y tiempo de ejecución



```
1 CREATE VIEW des_product
2 AS SELECT Id_producto, Nombre, Marca, Stock, Precios
3 FROM public.producto
4 WHERE Precios = '3.589.900';
5
6 drop view des_product
7 select * from des_product
```

Query Editor Query History

Explain Messages Notifications

Successfully run. Total query runtime: 37 msec.
0 rows affected.

Imagen 13. Ejecución de la segunda vista su código y tiempo de ejecución