Informe Primera Entrega Trabajo Práctico Base de Datos II

Integrantes del grupo:

- Matías Manzur, 62498
- Franco Rupnik, 62032
- Federico Shih, 62293
- Mauro Báez, 61747

Fecha de Entrega: 11/10/23

Introducción

En este informe encontrarán las queries de PostgreSQL requeridas para el trabajo práctico obligatorio.

1. Obtener el teléfono y el número de cliente del cliente con nombre “Wanda” y apellido “Baker”.

```
SELECT T.nro_telefono as "Número de Teléfono", C.nro_cliente as "Número de Cliente"
FROM E01_cliente C NATURAL JOIN E01_TELEFONO T
WHERE nombre = 'Wanda' AND apellido = 'Baker';
```

2. Seleccionar todos los clientes que tengan registrada al menos una factura.

```
SELECT *
FROM E01_CLIENTE C
WHERE EXISTS(SELECT *
FROM E01_FACTURA F
WHERE F.nro_cliente = C.nro_cliente);
```

3. Seleccionar todos los clientes que no tengan registrada una factura.

```
SELECT *
FROM E01_CLIENTE C
WHERE NOT EXISTS(SELECT *
        FROM E01_FACTURA F
        WHERE F.nro_cliente = C.nro_cliente);
```

4. Seleccionar los productos que han sido facturados al menos 1 vez.

```
SELECT *
FROM e01_producto p
WHERE exists (SELECT * FROM e01_detalle_factura df WHERE p.codigo_producto = df.codigo_producto);
```

5. Seleccionar los datos de los clientes junto con sus teléfonos.

```
SELECT *
FROM e01_cliente cliente LEFT JOIN e01_telefono telefono ON
cliente.nro_cliente = telefono.nro_cliente;
```

6. Devolver todos los clientes, con la cantidad de facturas que tienen registradas (admitir nulos en valores de Clientes).

```
SELECT C.nro_cliente as "Número de Cliente", C.nombre, C.apellido,
COUNT(F.nro_factura) as "Cantidad de Facturas Registradas"
FROM E01_CLIENTE C LEFT OUTER JOIN E01_FACTURA F ON C.nro_cliente
= F.nro_cliente
GROUP BY C.nro_cliente, C.nombre, C.apellido;
```

7. Listar todas las Facturas que hayan sido compradas por el cliente de nombre "Pandora" y apellido "Tate".

```
SELECT nro_factura as "Número de Factura", fecha,
round(CAST(total_sin_iva as numeric),2) as "Total Sin IVA", round(CAST(iva as
numeric),2) as IVA, round(CAST(total_con_iva as numeric),2) as "Total Con
IVA", nro_cliente as "Número de Cliente"
FROM e01_factura fa
WHERE EXISTS (SELECT * FROM e01_cliente cl WHERE cl.nro_cliente =
fa.nro_cliente AND cl.nombre = 'Pandora' AND cl.apellido = 'Tate');
```

8. Listar todas las Facturas que contengan productos de la marca “In Faucibus Inc.”

```
SELECT *
FROM E01_FACTURA F
WHERE F.nro_factura in (SELECT distinct D.nro_factura
                FROM E01_DETALLE_FACTURA D NATURAL JOIN
E01_PRODUCTO P
                WHERE P.marca = 'In Faucibus Inc.');
```

9. Mostrar cada teléfono junto con los datos del cliente.

```
SELECT *
FROM e01_cliente cliente RIGHT JOIN e01_telefono telefono ON
cliente.nro_cliente = telefono.nro_cliente;
```

10. Mostrar nombre y apellido de cada cliente junto con lo que gastó en total (con IVA incluido).

```
SELECT nombre, apellido, round(CAST(sum(total_con_iva) as numeric),2)
FROM e01_cliente NATURAL JOIN e01_factura
GROUP BY nombre, apellido;
```

Vistas
1. Se debe realizar una vista que devuelva las facturas ordenadas por fecha.

```
CREATE VIEW facturas_por_fecha AS
SELECT * FROM e01_factura
ORDER BY fecha;
```

2. Se necesita una vista que devuelva todos los productos que aún no han sido facturados.

```
CREATE VIEW productos_sin_factura AS
SELECT * FROM e01_producto p
WHERE NOT EXISTS (SELECT * FROM e01_detalle_factura df WHERE
p.codigo_producto = df.codigo_producto);
```

ABM

Repositorio: [RupnikF/TPO-BDII-2Q2023 (github.com)]

Ambiente de ejecución

Requerimientos

- **NodeJS version 16.14.0**

- **NPM 8.3.1**

Pasos de ejecución

- Levantar servidor postgresql
- Modificar configuración de servidor de postgres en config.js
- Correr
    - $ npm install
    - $ npm start
- Importar *API BDII.postman_collection.json* a Postman
- Cambiar environment
- Probar Altas, Bajas y Modificaciones

Endpoints Disponibles:

GET http://localhost:3000/clientes

GET http://localhost:3000/productos

POST http://localhost:3000/clientes
Cuyo body ha de ser:

```
{
        “nombre”:String,
        “apellido”:String,
        “direccion”:String,
        “activo”:Int
}
```

POST http://localhost:3000/productos
Cuyo body ha de ser:

```
{
        “marca”:String,
        “nombre”:String,
        “descripcion”:String,
        “precio”: Int,
        “stock”: Int
}
```

PUT http://localhost:3000/clientes/:id
Cuyo body ha de ser:

```
{
        “nombre”:String,
        “apellido”:String,
        “direccion”:String,
        “activo”:Int
}
```

PUT http://localhost:3000/productos/:id
Cuyo body ha de ser:
{
	“marca”:String,
	“nombre”:String,
	“descripcion”:String,
	“precio”: Int,
	“stock”: Int
}

DELETE http://localhost:3000/clientes/:id

Para poder probar fácilmente este ABM, pueden hacer uso de la funcionalidad Importar de Postman como se encuentra explicitado en el README.