Base de Datos 2 - 72.41

Trabajo Práctico Obligatorio

Informe

Profesores:

Rodríguez, Guillermo
Rodriguez Babino, Cecilia

Grupo 6:

Francois, Gaston Ariel (62500)
Mentasti, José Rodolfo (62248)
Preiti Tasat, Axel Facundo (62618)
Boizumault, Maïwenn (65988)

11 de octubre de 2023

Índice

Introducción

En este informe se detallarán los aspectos técnicos y las decisiones tomadas durante el desarrollo del trabajo práctico obligatorio. Se comenzará explicando el ambiente de ejecución del proyecto, siguiendo con la estrategia de migración llevada a cabo y mostrando algunas capturas de pantalla de los servicios. Además, habrá un apartado para las queries y las vistas solicitadas para ambas DBMS.

Aclaraciones importantes

1. Para facilitar la inserción de nuevas tuplas en la tabla de clientes y productos, se decidió cambiar el tipo de dato de sus identificadores (nro_cliente y nro_factura, respectivamente) a SERIAL. Por lo tanto, se modificaron las tablas y los datos insertados provistos por la cátedra.
2. En cuanto a la migración entre PostreSQL y MongoDB, se determinó que no sea automática sino que sea mediante la instrucción del usuario en tiempo de compilación.

Ambiente de ejecución

El proyecto se ejecuta dentro de una red de contenedores Docker, aprovechando las herramientas ofrecidas por Docker Compose. Se cuenta con un contenedor para el servicio de la API, un contenedor para MongoDB y un contenedor para PostgreSQL, donde los tres servicios se conectan internamente gracias a la red proporcionada por Docker Compose.

Fig. 1: Conexión desde la máquina local hacia la red de contenedores Docker

Docker

Se utilizó Docker como aplicación de sincronización para los servicios. Gracias a Docker Compose, con un simple comando en la terminal, se crea una red de contenedores: ***bd2_tpo_bd_postgres*** para PostgreSQL, ***bd2_tpo_bd_mongo*** para MongoDB y ***bd2_tpo_server*** para la API. Además, se agregó un volumen compartido en donde se almacena temporalmente los archivos de migración creados por PostgreSQL e importados por MongoDB.

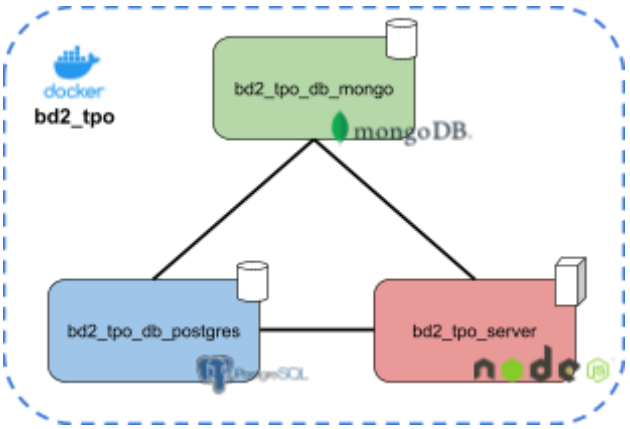

Fig. 2: Esquema lógico dentro de la red de contenedores Docker

API

Decidimos implementar la API en un servidor Node.js, y su implementación se desarrolló con la librería Express por su facilidad. Por otro lado, se utilizaron las librerías *pg* para las consultas SQL y *mongodb* para las consultas de MongoDB.

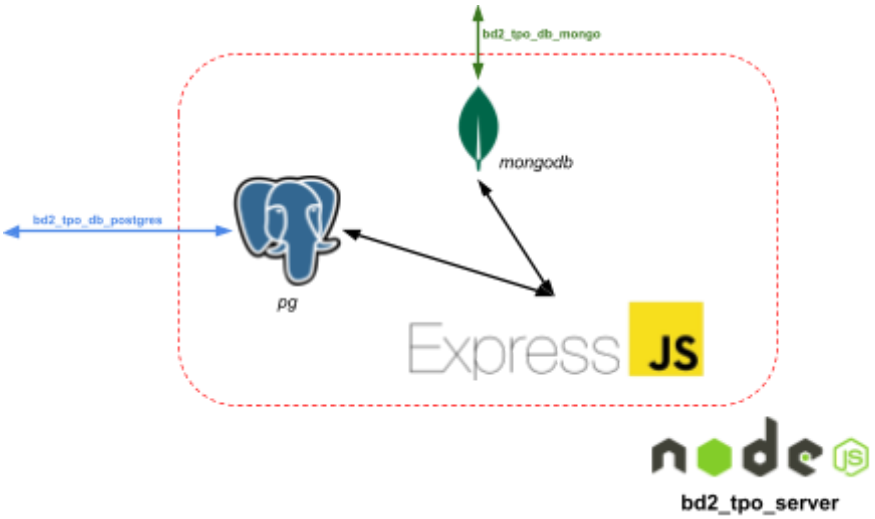

Fig. 3: Esquema lógico del servicio API

Estrategia de migración

Para lograr la migración de los datos de PostgreSQL a MongoDB, se realizó una serie de pasos para cada una de las tablas: copiar los datos de PostgreSQL a un archivo, eliminar la colección en MongoDB e importar el archivo a MongoDB sobre la colección recién borrada. El segundo paso se realiza para evitar almacenar información desactualizada y, en cambio, empezar con un “snapshot” de la base de datos de PostgreSQL.

El primer paso se realizó utilizando el comando ***\copy*** ofrecido por PostgreSQL. Para las tablas E01_CLIENTE, E01_TELEFONO y E01_PRODUCTO se exportaron como archivos TSV. Para el caso de E01_FACTURA y E01_DETALLE_FACTURA, se decidió embeber las tuplas de la segunda tabla en su correspondiente tupla de la primera tabla, exportandolo como un archivo JSON. Esto fue posible gracias a que PostgreSQL cuenta con los procedimientos *array_agg*, *array_to_json* y *row_to_json* para transformar una junta de tablas en un arreglo de objetos JSON. Estos archivos se almacenan en un volumen compartido, como se mencionó anteriormente, para que puedan ser utilizados por el contenedor de MongoDB.

Para el tercer paso, se utilizó el comando *mongoimport* para copiar dichos archivos a la base de datos de MongoDB. Siguiendo las convenciones de MongoDB, las claves de las tablas E01_CLIENTE, E01_PRODUCTO y E01_FACTURA (nro_cliente, codigo_producto, nro_factura, respectivamente) se almacenan en el campo *_id*.

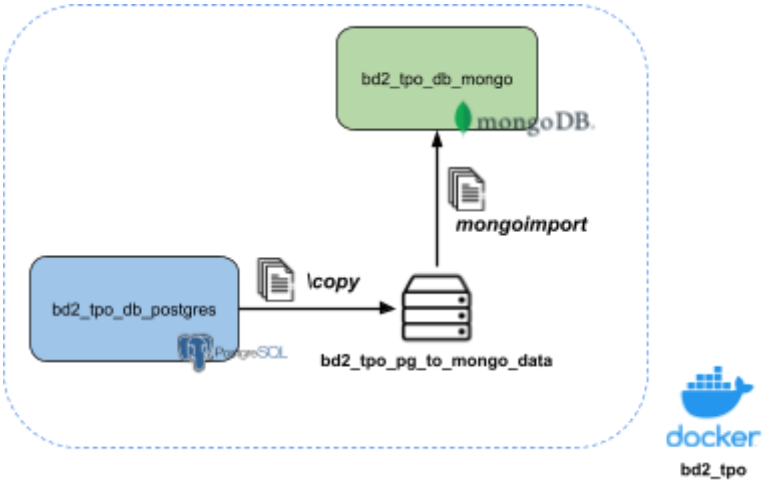

Fig. 4: Esquema lógico de la migración

Queries

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

PostgreSQL

```
SELECT nro_telefono, nro_cliente
FROM E01_CLIENTE NATURAL JOIN E01_TELEFONO
WHERE nombre='Wanda' and apellido='Baker';
```

MongoDB

```
db.E01_CLIENTE.aggregate([
    {
        $lookup:
        {
            from:"E01_TELEFONO",
            localField:"_id",
            foreignField:"nro_cliente",
            as:"telefonos"
        }
    },
    {
        $match:{nombre:"Wanda", apellido:"Baker"}
    },
    {
        $project:{
            "telefonos":1,
            _id:1
        }
    }
]);
```

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

PostgreSQL

```
SELECT distinct cliente.*
FROM E01_CLIENTE cliente NATURAL JOIN E01_FACTURA factura;
```

MongoDB

```
db.E01_CLIENTE.aggregate([
    {
```

```
        $lookup:{
            from:"E01_FACTURA",
            localField:"_id",
            foreignField:"nro_cliente",
            as:"facturas"
        }
    },
    {
        $match:{
            facturas:{"$ne":[]}
        }
    },
    {
        $project:{
            _id:1,
            nombre:1,
            apellido:1,
            direccion:1,
            activo:1
        }
    }
]);
```

Seleccionar todos los clientes que no tengan registrada una factura.

PostgreSQL

```
SELECT *
FROM e01_cliente
WHERE nro_cliente NOT IN (
    SELECT nro_cliente
    FROM e01_factura
    );
```

MongoDB

```
db.E01_CLIENTE.aggregate([
    {
        $lookup:{
            from:"E01_FACTURA",
            localField:"_id",
```

```
            foreignField:"nro_cliente",
            as:"facturas"
        }
    },
    {
        $match:{
            facturas:[]
        }
    },
    {
        $project:{
            _id:1,
            nombre:1,
            apellido:1,
            direccion:1,
            activo:1
        }
    }
]);
```

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

PostgreSQL

```
SELECT distinct producto.*
FROM e01_producto producto NATURAL JOIN e01_detalle_factura;
```

MongoDB

```
db.E01_PRODUCTO.find({
 _id:
   {$in:
      db.E01_FACTURA.distinct("detalles_factura.codigo_producto")}
})
```

Seleccionar los datos de los clientes junto con sus teléfonos.

PostgreSQL

```
SELECT cliente.*, nro_telefono
```

```
FROM e01_cliente cliente LEFT OUTER JOIN e01_telefono ON
(cliente.nro_cliente=e01_telefono.nro_cliente);
```

MongoDB

```
db.E01_CLIENTE.aggregate([
    {
        $lookup:
        {
            from:"E01_TELEFONO",
            localField:"_id",
            foreignField:"nro_cliente",
            as:"telefonos"
        }
    }
]);
```

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

PostgreSQL

```
SELECT coalesce(cliente.nro_cliente, factura.nro_cliente) as
nro_cliente, count(nro_factura) as cantidad
FROM e01_cliente cliente FULL OUTER JOIN e01_factura factura
ON (cliente.nro_cliente = factura.nro_cliente)
GROUP BY cliente.nro_cliente, factura.nro_cliente;
```

MongoDB

```
db.E01_CLIENTE.aggregate([
    {
        $lookup:{
            from:"E01_FACTURA",
            localField:"_id",
            foreignField:"nro_cliente",
            as:"facturas"
        }
    },
    {
        $project:{
            _id:1,
```

```
            cantidad_facturas:{$size:"$facturas"}
        }
    }
]);
```

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

PostgreSQL

```
SELECT factura.*
FROM e01_factura factura NATURAL JOIN e01_cliente cliente
WHERE cliente.nombre='Pandora' and cliente.apellido='Tate';
```

MongoDB

```
db.E01_FACTURA.aggregate([
    {
        $lookup:{
            from:"E01_CLIENTE",
            localField:"nro_cliente",
            foreignField:"_id",
            as:"cliente"
        }
    },
    {

$match:{"cliente.nombre":"Pandora","cliente.apellido":"Tate"}
    },
    {
        $project:{
            fecha:1,
            iva:1,
            nro_factura:1,
            total_sin_iva:1,
            total_con_iva:1,
            nro_cliente:"$cliente._id"
        }
    }
]);
```

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

PostgreSQL

```
SELECT distinct factura.*
FROM e01_factura factura NATURAL JOIN e01_detalle_factura
d_factura
NATURAL JOIN e01_producto producto
WHERE marca='In Faucibus Inc.';
```

MongoDB

```
db.E01_FACTURA.aggregate([
    {
        $unwind:"$detalles_factura"
    },
    {
        $lookup:{
            from:"E01_PRODUCTO",
            localField:"detalles_factura.codigo_producto",
            foreignField:"_id",
            as:"producto"
        }
    },
    {
      $match:{"producto.marca":"In Faucibus Inc."}
    },
    {
        $project:{
            nro_factura:1,
            fecha:1,
            iva:1,
            nro_cliente:1,
            total_con_iva:1,
            total_sin_iva:1,
            _id:0
        }
    }
]);
```

Mostrar cada teléfono junto con los datos del cliente.

PostgreSQL

```
SELECT telefono.*, cliente.nro_cliente
FROM e01_telefono telefono LEFT OUTER JOIN e01_cliente cliente
ON (cliente.nro_cliente=telefono.nro_cliente);
```

MongoDB

```
db.E01_TELEFONO.aggregate([
    {
        $lookup: {
            from: "E01_CLIENTE",
            localField: "nro_cliente",
            foreignField: "_id",
            as: "cliente"
        }
    },
]);
```

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

PostgreSQL

```
SELECT nombre, apellido, COALESCE(sum(total_con_iva), 0) as
total_gastado
FROM e01_cliente cliente LEFT OUTER JOIN e01_factura factura
ON (cliente.nro_cliente = factura.nro_cliente)
GROUP BY cliente.nro_cliente, nombre, apellido;
```

PostgreSQL

```
db.E01_CLIENTE.aggregate([
    {
        $lookup: {
            from: "E01_FACTURA",
            localField: "_id",
            foreignField: "nro_cliente",
            as: "facturas"
        }
    },
```

```
    {
        $project:{
            nombre: 1,
            apellido: 1,
            _id: 0,
            total_gastado: {$sum:"$facturas.total_con_iva"}
        }
    }
]);
```

Vistas

Vista que devuelva las facturas ordenadas por fecha.

PostgreSQL

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

MongoDB

```
db.createView(
    "facturas_por_fecha",
    "E01_FACTURA",
    [
        {
            $sort: {fecha: 1}
        },
        {
            $project: {
                detalles_factura: 0
            }
        }
    ]
);
```

Vista que devuelva todos los productos que aún no han sido facturados.

PostgreSQL

```
CREATE VIEW productos_no_facturados AS
SELECT *
FROM e01_producto
WHERE codigo_producto NOT IN (
    SELECT codigo_producto
    FROM e01_detalle_factura
    WHERE codigo_producto IS NOT NULL
    );
```

MongoDB

```
db.createView(
    "productos_no_facturados",
    "E01_PRODUCTO",
    [
       {
            $lookup: {
                from: "E01_FACTURA",
                localField: "_id",
                foreignField: "detalles_factura.codigo_producto",
                as: "facturas"
            }
       },
       {
            $match: {
                facturas: { $eq: [] }
            }
       },
       {
            $project: {
                facturas: 0
            }
       }
    ]
);
```

Conclusión

Una vez finalizado el presente proyecto, podemos verificar las diferencias entre cada tecnología. Pudiéndose observar claramente las diferencias en la sintaxis de cada lenguaje en la creación y en la consulta de datos. A su vez, se pudieron entender las similitudes y los estrechos vínculos entre lo relacional y lo no relacional a la hora de la consulta de datos. Finalmente, se puede observar como a nivel de API resulta transparente el tipo de implementación interna que se utilice.