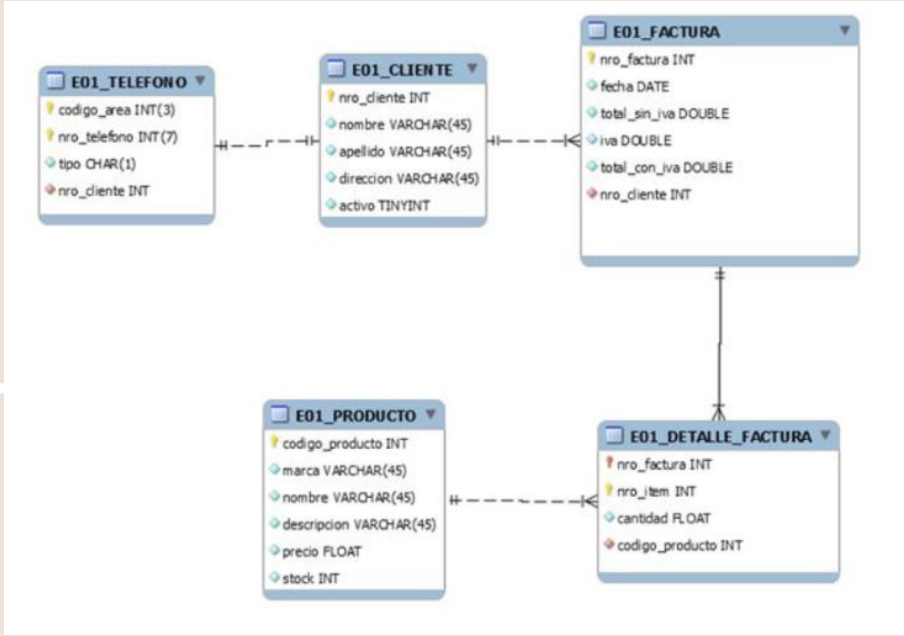
Trabajo Práctico Obligatorio
Base de Datos II
Grupo 16

Benjamin Delasoie - 61231
Franco Panighini - 61258
Santiago Rivas Betancourt - 61007

PostgreSQL

Queries y Vistas



Query 3

```sql
SELECT
    *
FROM
    E01_CLIENTE
WHERE
    NRO_CLIENTE NOT IN (
        SELECT
            DISTINCT NRO_CLIENTE
        FROM
            E01_FACTURA
    );
```

Query 10

```sql
SELECT
    nombre,
    apellido,
    SUM(total_con_iva)
FROM
    E01_CLIENTE NATURAL
    JOIN E01_FACTURA
GROUP BY
    nombre,
    apellido;
```

Vista 1:
Listar las facturas por fecha de creación

```sql
CREATE VIEW facturas_por_fecha AS
SELECT
    *
FROM
    E01_FACTURA
ORDER BY
    fecha;
```

Vista 2:
Listar los productos que aún no han sido facturados.

```sql
CREATE VIEW productos_no_facturados AS
SELECT
    *
FROM
    E01_PRODUCTO
WHERE
    codigo_producto NOT IN (
        SELECT
            codigo_producto
        FROM
            E01_DETALLE_FACTURA NATURAL
            JOIN E01_FACTURA
    );
```

MongoDB

Modelo, migración, queries y vistas

Modelo en MongoDB

Desnormalización: oportunidades y limitaciones.

```
bd2tpo> db.e01_telefono.find({})
[
  {
    _id: ObjectId("655811bc059aa50eb3bc65fb"),
    codigo_area: 193,
    nro_telefono: 4625992,
    tipo: 'F',
    nro_cliente: 42
  },
```

```
bd2tpo> db.e01_factura.find({detalle_facturas: {$exists:true}}).limit(1)
[
  {
    _id: ObjectId("656584b973754bbb204f6728"),
    nro_factura: 18,
    fecha: 'Wed Aug 10 2016 00:00:00 GMT-0300 (GMT-03:00)',
    total_sin_iva: 198124.46400000004,
    iva: 21,
    total_con_iva: 239730.60144000006,
    nro_cliente: 3,
    detalle_facturas: [
      { codigo_producto: 32, nro_item: 25, cantidad: 14 },
      { codigo_producto: 12, nro_item: 31, cantidad: 28 },
      { codigo_producto: 3, nro_item: 69, cantidad: 89 },
      { codigo_producto: 28, nro_item: 81, cantidad: 6 },
      { codigo_producto: 17, nro_item: 88, cantidad: 27 },
      { codigo_producto: 33, nro_item: 100, cantidad: 38 },
      { codigo_producto: 54, nro_item: 176, cantidad: 16 },
      { codigo_producto: 61, nro_item: 179, cantidad: 57 },
      { codigo_producto: 88, nro_item: 207, cantidad: 27 },
      { codigo_producto: 69, nro_item: 212, cantidad: 88 }
    ]
  }
]
```

Modelo en MongoDB

Desnormalización: oportunidades y limitaciones.

```
await createUniqueIndex(client, 'e01_cliente', ['nro_cliente']);
await createUniqueIndex(client, 'e01_factura', ['nro_factura']);
await createUniqueIndex(client, 'e01_producto', ['codigo_producto']);
await createUniqueIndex(client, 'e01_telefono', ['codigo_area', 'nro_telefono']);
```

Migración de datos

```
async function insertDataIntoMongoDB(dataArray, client, collectionName, columns) {
    const db = client.db(dbName);
    const collection = db.collection(collectionName);

    try {
        const data = dataArray.split('\n').map((line) => {
            const values = line.split('\t');
            const obj = {};
            for (let i = 0; i < values.length; i++) {
                if (isNumeric(values[i])) {
                    obj[columns[i]] = parseFloat(values[i]);
                } else {
                    obj[columns[i]] = values[i];
                }
            }
            return obj;
        });

        const result = await collection.insertMany(data);
        console.log(`${result.insertedCount} documents inserted into MongoDB for collection ${collectionName}`);
    } catch (error) {
        console.error('Error inserting data into MongoDB:', error);
    }
}
```

Migración de datos

```
async function insertDetalleFacturasIntoFacturas(detalleFacturas, client) {
    const db = client.db(dbName);
    const facturaCollection = db.collection('e01_factura');

    try {
        for (const detalleFactura of detalleFacturas) {
            const nroFactura = detalleFactura.nro_factura;

            const detalleData = {
                codigo_producto: detalleFactura.codigo_producto,
                nro_item: detalleFactura.nro_item,
                cantidad: detalleFactura.cantidad,
            };

            // Find the factura document with matching nro_factura
            const facturaDocument = await facturaCollection.findOne({ nro_factura: nroFactura });
            if (facturaDocument) {
                await facturaCollection.updateOne(
                    { nro_factura: nroFactura },
                    {
                        $push: {
                            detalle_facturas: detalleData,
                        },
                    }
                );
            } else {
                console.error(`No matching factura found for nro_factura ${nroFactura}`);
            }
        }
        console.log(`${detalleFacturas.length} detalle_facturas inserted into factura documents`);
    } catch (error) {
        console.error('Error inserting detalle_facturas into factura documents:', error);
    }
}
```

Migración de datos

```
transferData = async () => {
    let client;

    client = new MongoClient(mongoURL);
    await client.connect();
    console.log(client);

    // Drop collections if they exist
    await dropCollections(client);
    try {

        for (const table of tablesToExport) {
            // Fetch data and column names from the table
            const data = await db.any(`SELECT * FROM ${table}`);
            const columns = Object.keys(data[0]);

            const rows = [];

            data.forEach((row) => {
                rows.push(columns.map((column) => row[column]).join('\t'));
            });

            // Convert data to TSV format
            const tsvData = rows.join('\n');

            await insertDataIntoMongoDB(tsvData, client, table, columns);
        }
```

```
        const detalleFacturaData = await fetchDetalleFacturasFromDatabase();

        await insertDetalleFacturasIntoFacturas(detalleFacturaData, client);

        // Create unique indexes for the specified fields
        await createUniqueIndex(client, 'e01_cliente', ['nro_cliente']);
        await createUniqueIndex(client, 'e01_factura', ['nro_factura']);
        await createUniqueIndex(client, 'e01_producto', ['codigo_producto']);
        await createUniqueIndex(client, 'e01_telefono', ['codigo_area', 'nro_telefono']);

    } catch (error) {
        console.error('Error:', error);
    } finally {
        console.log(client);
        pgp.end();
        client.close();

    }
};
```

Query 3

```
db.e01_cliente.find({
    "nro_cliente": {
        $nin: db.e01_factura.distinct("nro_cliente")
    }
})
```

Query 10

```
db.e01_factura.aggregate([
  {
    $group: {
      _id: "$nro_cliente",
      totalGasto: { $sum: "$total_con_iva" }
    }
  },
  {
    $lookup: {
      from: "e01_cliente",
      localField: "_id",
      foreignField: "nro_cliente",
      as: "cliente_info"
    }
  },
  {
    $unwind: "$cliente_info"
  },
  {
    $project: {
      _id: 0,
      "Nombre del Cliente": "$cliente_info.nombre",
      "Apellido del Cliente": "$cliente_info.apellido",
      "Gasto Total (con IVA)": "$totalGasto"
    }
  }
])
```

Vista 1

```
db.createView("facturas_sorted_by_fecha", "e01_factura", [
  {
    $sort: { fecha: 1 }
  }
])
```

Vista 2

```
db.createView("productos_not_facturados", "e01_producto",
  {
    $lookup: {
      from: "e01_factura",
      localField: "codigo_producto",
      foreignField: "detalle_facturas.codigo_producto",
      as: "facturas"
    }
  },
  {
    $match: {
      facturas: { $size: 0 }
    }
  }
]);
```

API: Express.js

Inicialización

```
const express = require('express');
const bodyParser = require('body-parser');
const { MongoClient } = require('mongodb');
const dotenv = require('dotenv');
const app = express();
const port = 3000;

app.use(bodyParser.json());
```

API GET Routes

```
// Get all clients from MongoDB
app.get('/mongodb/clients', async (req, res) => {
    try {
        const mongoDb = mongoClient.db(dbConfig.database);
        const clientsCollection = mongoDb.collection('e01_cliente');
        const data = await clientsCollection.find({}).toArray();
        res.json(data);
    } catch (error) {
        console.error(error);
        res.status(500).json({ error: 'An error occurred.' });
    }
});
```

```
// Get all products from MongoDB
app.get('/mongodb/products', async (req, res) => {
    try {
        const mongoDb = mongoClient.db(dbConfig.database);
        const clientsCollection = mongoDb.collection('e01_producto');
        const data = await clientsCollection.find({}).toArray();
        res.json(data);
    } catch (error) {
        console.error(error);
        res.status(500).json({ error: 'An error occurred.' });
    }
});
```

API POST Routes

```
app.post('/mongodb/clients/insert', async (req, res) => {
    try {
        const { p_nro_cliente, p_nombre, p_apellido, p_direccion, p_activo } = req.body;

        // Connect to the MongoDB database
        await mongoClient.connect();
        const mongoDb = mongoClient.db(dbConfig.database);
        const clientsCollection = mongoDb.collection('e01_cliente');

        const newClient = {
            nro_cliente: p_nro_cliente,
            nombre: p_nombre,
            apellido: p_apellido,
            direccion: p_direccion,
            activo: p_activo,
        };

        const result = await clientsCollection.insertOne(newClient);
        res.status(201).json({ message: 'Client inserted successfully' });
    } catch (error) {
        console.error(error);
        res.status(500).json({ error: 'An error occurred while inserting the client into MongoDB' });
    } finally {
        mongoClient.close();
    }
});
```

API POST Routes

```
app.post('/mongodb/products/update', async (req, res) => {
    try {
        const { p_codigo_producto, p_marca, p_nombre, p_descripcion, p_precio, p_stock } = req.body;

        // Connect to the MongoDB database
        await mongoClient.connect();
        const mongoDb = mongoClient.db(dbConfig.database);
        const productsCollection = mongoDb.collection('e01_producto');
        const filter = { codigo_producto: p_codigo_producto };

        const updateData = {
            $set: {
                marca: p_marca,
                nombre: p_nombre,
                descripcion: p_descripcion,
                precio: p_precio,
                stock: p_stock,
            },
        };

        const result = await productsCollection.updateOne(filter, updateData);

        console.log(result);
        mongoClient.close();
        res.status(200).json({ message: 'Product updated successfully' });
    } catch (error) {
        console.error(error);
        res.status(500).json({ error: 'An error occurred while updating the product in MongoDB' });
    }
});
```

DEMO

GRACIAS