Instituto Tecnológico de Buenos Aires

Trabajo Práctico Obligatorio - Grupo 10

72.41 - Base de Datos II

Integrantes:

Sartorio, Alan
61379

Digon, Lucía
59030

Diaz Kralj, Luciana
60495

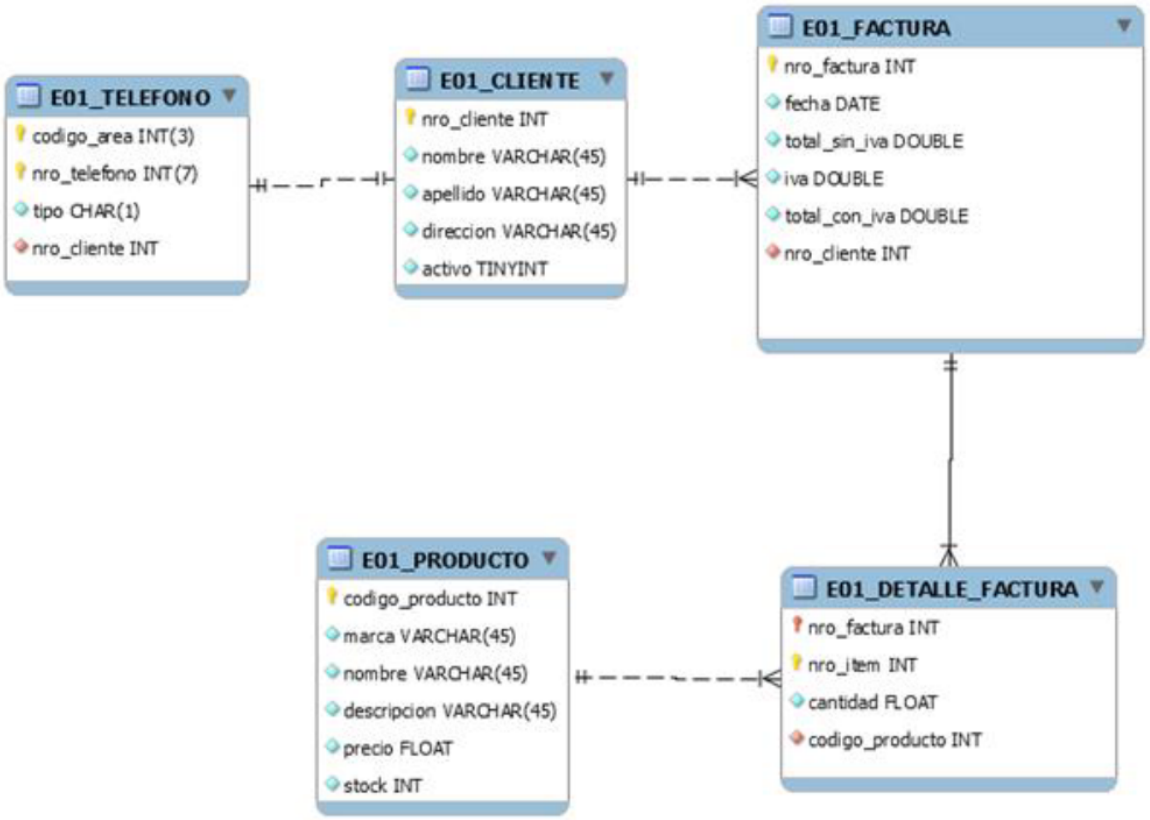
Índice

Enunciado

Usted es contratado para realizar un sistema de facturación, para lo cual deberá llevar el control de los productos comprados por los clientes. La facturación de productos a un cliente consiste en chequear la disponibilidad en el stock de los productos, decrementar la cantidad vendida y calcular el monto total de la factura considerando el IVA y los descuentos que se aplican de acuerdo al volumen de productos que se compran. El sistema deberá proporcionar los mecanismos necesarios para:

- Dar de alta a nuevos clientes, baja y modificación de los ya existentes.
- Dar de alta a nuevos productos y modificación de los ya existentes. Tenga en cuenta que el precio de un producto es sin IVA.

El sistema estará en base al siguiente DER:



Parte Relacional

Para esta sección del trabajo práctico desarrollamos las consultas en PostgreSQL, ejecutándolo desde un contenedor de Docker, creando y poblando las tablas con el esquema y los datos proporcionados por la cátedra. Las consultas y operaciones se realizaron mediante IntelliJ IDEA.

Para las consultas que no explicitan campos particulares, se devuelven todas las columnas. Todas las consultas pueden encontrarse en `relacional/queries.sql`.

Consultas PostgreSQL

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

```sql
SELECT t.codigo_area, t.nro_telefono, c.nro_cliente
FROM E01_TELEFONO t
         INNER JOIN E01_CLIENTE c ON t.nro_cliente = c.nro_cliente
WHERE c.nombre = 'Wanda' AND c.apellido = 'Baker';
```

Resultados (2 tuplas):

2 rows

	codigo_area	nro_telefono	nro_cliente
1	313	4578765	52
2	593	4118559	52

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

```sql
SELECT * FROM E01_CLIENTE c
WHERE EXISTS (
    SELECT 1
    FROM E01_FACTURA f
    WHERE c.nro_cliente = f.nro_cliente
);
```

Resultados (4 de 98 tuplas):

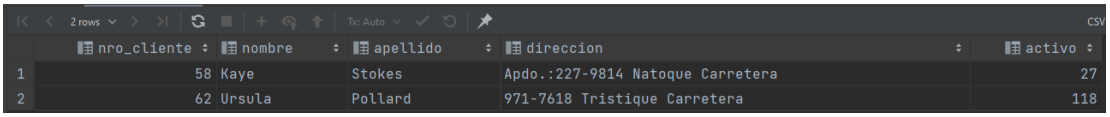
98 rows

	nro_cliente	nombre	apellido	direccion	activo
1	1	Xerxes	Hale	129-5974 Suspendisse Ctra.	89
2	2	Brent	Leblanc	Apartado núm.: 372, 5244 Nibh.	58
3	3	Kasper	Shannon	Apdo.:304-6908 Class Ctra.	38
4	4	Pandora	Tate	1221 Egestas. Carretera	124

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

```
SELECT * FROM E01_CLIENTE c
WHERE NOT EXISTS (
    SELECT 1
    FROM E01_FACTURA f
    WHERE c.nro_cliente = f.nro_cliente
);
```

Resultados (2 tuplas):

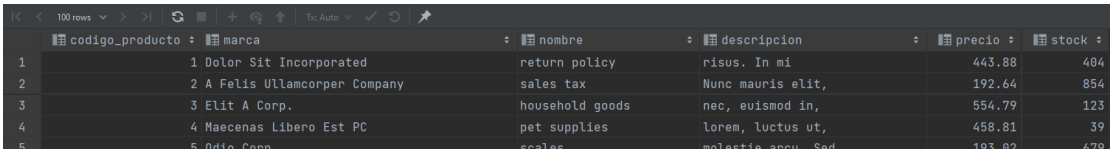


	nro_cliente	nombre	apellido	direccion	activo
1	58	Kaye	Stokes	Apdo.:227-9814 Natoque Carretera	27
2	62	Ursula	Pollard	971-7618 Tristique Carretera	118

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

```
SELECT * FROM E01_PRODUCTO p
WHERE EXISTS (
    SELECT 1
    FROM E01_DETALLE_FACTURA df
    WHERE p.codigo_producto = df.codigo_producto
);
```

Resultados (4 de 100 tuplas):

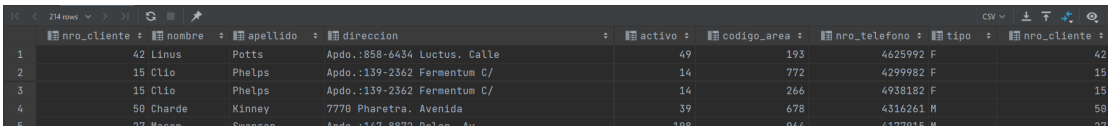


	codigo_producto	marca	nombre	descripcion	precio	stock
1	1	Dolor Sit Incorporated	return policy	risus. In mi	443.88	404
2	2	A Felis Ullamcorper Company	sales tax	Nunc mauris elit,	192.64	854
3	3	Elit A Corp.	household goods	nec, euismod in,	554.79	123
4	4	Maecenas Libero Est PC	pet supplies	lorem, luctus ut,	458.81	39

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

```
SELECT * FROM E01_CLIENTE c
    LEFT JOIN E01_TELEFONO t ON c.nro_cliente = t.nro_cliente;
```

Resultados (4 de 214 tuplas):



	nro_cliente	nombre	apellido	direccion	activo	codigo_area	nro_telefono	tipo	nro_cliente
1	42	Linus	Potts	Apdo.:858-6434 Luctus. Calle	49	193	4625992	F	42
2	15	Clio	Phelps	Apdo.:139-2362 Fermentum C/	14	772	4299982	F	15
3	15	Clio	Phelps	Apdo.:139-2362 Fermentum C/	14	266	4938182	F	15
4	50	Charde	Kinney	7770 Pharetra. Avenida	39	678	4316261	M	50

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

```
SELECT c.*, COUNT(f.nro_factura) AS cantidad_facturas
FROM E01_CLIENTE c
```

```
        LEFT JOIN E01_FACTURA f ON c.nro_cliente = f.nro_cliente
GROUP BY c.nro_cliente, c.nombre, c.apellido, c.direccion,
c.activo;
```

Resultados (4 de 100 tuplas):

	nro_cliente	nombre	apellido	direccion	activo	cantidad_facturas
1	55	Camden	Casey	4816 Magna Ctra.	52	5
2	27	Mason	Swanson	Apdo.:147-8872 Dolor, Av.	108	5
3	23	Drew	Boyle	663-3481 Nisl. Ctra.	88	4
4	56	Darius	Myers	6187 Eu, Ctra.	15	6

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

```
SELECT f.* FROM E01_FACTURA f
     INNER JOIN E01_CLIENTE c ON f.nro_cliente = c.nro_cliente
WHERE c.nombre = 'Pandora' AND c.apellido = 'Tate';
```

Resultados (7 tuplas):

	nro_factura	fecha	total_sin_iva	iva	total_con_iva	nro_cliente
1	17	2016-06-01	265939.596	21	321786.91116	4
2	1	2016-05-28	294369.60099999997	21	356187.21721	4
3	61	2016-12-10	141156.963	21	170799.92523	4
4	60	2017-02-05	108632.29500000001	21	131445.07695000002	4
5	66	2016-05-23	77261.163	21	93486.00723	4
6	55	2016-06-21	132705.16199999998	21	160573.24601999996	4
7	52	2016-07-01	187440.55400000003	21	226803.07034000003	4

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

```
SELECT f.* FROM E01_FACTURA f
     INNER JOIN E01_DETALLE_FACTURA df ON f.nro_factura =
df.nro_factura
     INNER JOIN E01_PRODUCTO p ON df.codigo_producto =
p.codigo_producto
WHERE p.marca = 'In Faucibus Inc.';
```

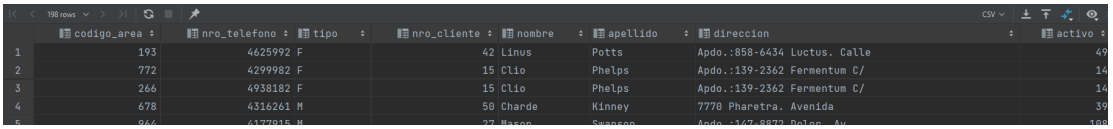
Resultados (4 de 27 tuplas):

	nro_factura	fecha	total_sin_iva	iva	total_con_iva	nro_cliente
1	5	2016-12-29	127843.699	21	154690.87579	20
2	6	2017-03-28	524431.2535000001	21	634561.8167350001	1
3	50	2017-03-14	378516.34800000006	21	458004.78108000004	15
4	27	2017-03-10	355809.31600000005	21	430529.27236000006	22

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

```
SELECT t.*, c.nombre, c.apellido, c.direccion, c.activo
FROM E01_TELEFONO t
     INNER JOIN E01_CLIENTE c ON t.nro_cliente = c.nro_cliente;
```

Resultados (4 de 198 tuplas):

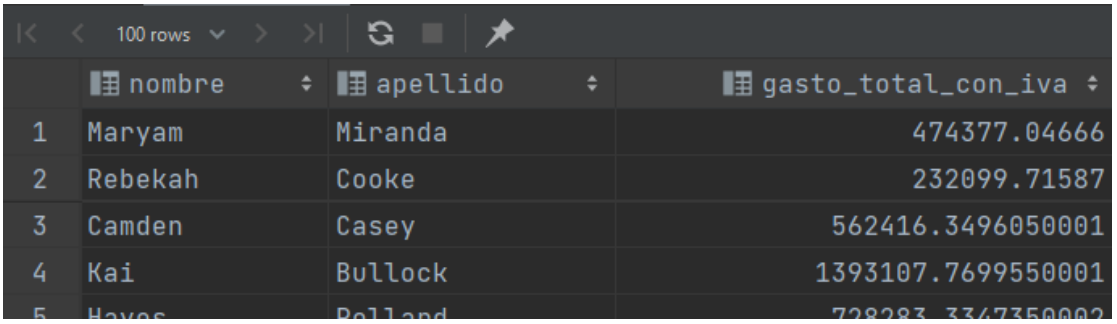


	codigo_area	nro_telefono	tipo	nro_cliente	nombre	apellido	direccion	activo
1	193	4625992	F	42	Linus	Potts	Apdo.:858-6434 Luctus. Calle	49
2	772	4299982	F	15	Clio	Phelps	Apdo.:139-2362 Fermentum C/	14
3	266	4938182	F	15	Clio	Phelps	Apdo.:139-2362 Fermentum C/	14
4	678	4316261	M	50	Charde	Kinney	7770 Pharetra. Avenida	39

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

```
SELECT c.nombre, c.apellido, SUM(f.total_con_iva) AS
gasto_total_con_iva
FROM E01_CLIENTE c
     LEFT JOIN E01_FACTURA f ON c.nro_cliente = f.nro_cliente
GROUP BY c.nombre, c.apellido;
```

Resultados (4 de 100 tuplas):



	nombre	apellido	gasto_total_con_iva
1	Maryam	Miranda	474377.04666
2	Rebekah	Cooke	232099.71587
3	Camden	Casey	562416.3496050001
4	Kai	Bullock	1393107.7699550001

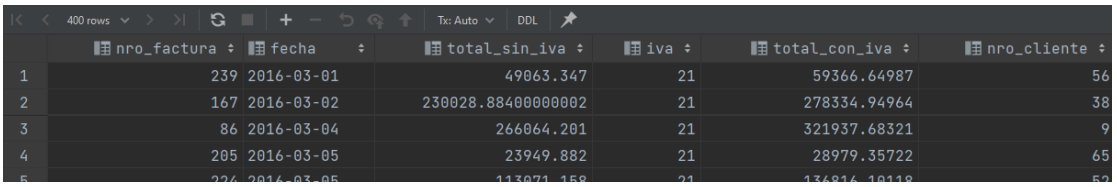
Vistas

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

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

SELECT * FROM FacturasOrdenadas;
```

Resultados (4 de 400 tuplas):



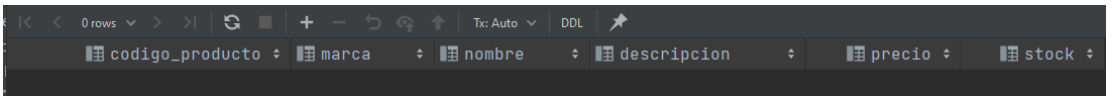
	nro_factura	fecha	total_sin_iva	iva	total_con_iva	nro_cliente
1	239	2016-03-01	49063.347	21	59366.64987	56
2	167	2016-03-02	230028.88400000002	21	278334.94964	38
3	86	2016-03-04	266064.201	21	321937.68321	9
4	205	2016-03-05	23949.882	21	28979.35722	65

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

```
CREATE VIEW ProductosNoFacturados AS
SELECT *
FROM E01_PRODUCTO p
    WHERE p.codigo_producto NOT IN (
        SELECT codigo_producto FROM E01_DETALLE_FACTURA
    );

SELECT * FROM ProductosNoFacturados;
```

Resultados (0 tuplas):



codigo_producto	marca	nombre	descripcion	precio	stock

Parte No Relacional

Para esta sección del trabajo práctico desarrollamos las consultas en MongoDB, ejecutándolo desde un contenedor de Docker, creando y poblando las tablas con el esquema y los datos empleados en la sección relacional. Las consultas se probaron mediante `mongosh` en el contenedor de Docker.

Aquellas consultas que no explicitan campos particulares, se interpretaron como que requieren una devolución de todas las columnas. Todas las consultas pueden encontrarse en `no-relacional/queries.sql`.

Esquema

Decidimos representar los datos en base a los siguientes esquemas realizados a modo ilustrativo:

`E01_CLIENTE`

```
{
      _id: Number,
      nombre: String,
      apellido: String,
      direccion: String,
      activo: Number,
      telefonos: [
                    {
                       codigo_area: Number,
                       nro_telefono: Number,
                       tipo: String
                    },
                    {
                       ...
                    }
                ]
}
```

Tomamos la decisión de embeber los `teléfonos` dentro del objeto `cliente`. Asumimos que raramente se querrán saber teléfonos sin información sobre el cliente al que pertenecen.

E01_FACTURA

```
{
      _id: Number,
      fecha: Date,
      total_sin_iva: Number,
      iva: Number,
      total_con_iva: Number,
      nro_cliente: Number,
      detalles: [
                     {
                        nro_item: Number,
                        cantidad: Number,
                        codigo_producto: Number
                     },
                     {
                        ...
                     }
                ]
}
```

Similar al caso anterior, como cada detalle pertenece únicamente a una factura, se tomó la decisión de incluirlos dentro de la factura a la que corresponden.

E01_PRODUCTO

```
{
      _id: Number,
      marca: String,
      nombre: String,
      descripcion: String,
      precio: Number,
      stock: Number,
}
```

Como un producto puede estar en varias facturas, lo dejamos por sí mismo para evitar repetición de información.

Migración

Para migrar las tablas de PostgreSQL a MongoDB, se realizó un proceso con algunos prerrequisitos, si alguno de los prerrequisitos no se cumple, hay que alterar los comandos utilizados. Los prerrequisitos son:

- Correr PostgreSQL en Docker con las siguientes propiedades:
    - `POSTGRES_DB=postgres`
    - `POSTGRES_USER=postgres`
    - `POSTGRES_PASSWORD=docker`
    - Nombre del contenedor: `postgres-bd2`

Comando que usamos para PostgreSQL:

```
docker run --name postgres-bd2 -e POSTGRES_PASSWORD=docker -e
POSTGRES_USER=postgres -e POSTGRES_DB=postgres -v
postgres-bd2:/var/lib/postgresql/data -p 5432:5432 postgres
```

- Correr MongoDB en Docker con las siguientes propiedades:
    - `MONGO_INITDB_ROOT_USERNAME=mongo`
    - `MONGO_INITDB_ROOT_PASSWORD=docker`
    - Nombre de la base de datos: `admin` (default)
    - Nombre del contenedor: `mongo-bd2`

```
docker run --name mongo-bd2 -e MONGO_INITDB_ROOT_USERNAME=mongo -e
MONGO_INITDB_ROOT_PASSWORD=docker -p 27017:27017 mongo:latest
```

Una vez hecho esto, para la migración se efectuaron los siguientes pasos:

1. Primero utilizamos las funciones `row_to_json` y `array_to_json` para convertir filas de una tabla y arrays en objetos JSON, la función `array_agg` para juntar los valores de una columna en un objeto JSON, y el comando `COPY (...) TO` para mover la respuesta de la consulta a un archivo. Con esto traducimos los datos de las tablas a archivos en formato JSON que se almacenaron en nuestro contenedor de Postgres en Docker (`postgres-bd2`).

    `Productos`

    ```
    COPY (
        SELECT array_to_json(array_agg(json))
        FROM (SELECT row_to_json(p) AS json
              FROM (SELECT codigo_producto AS _id,
    ```

```
                    marca, nombre, descripcion, precio, stock
               FROM e01_producto) p) j
    ) TO '/tmp/productos.json' WITH (FORMAT text, HEADER FALSE);
```

Clientes

```
COPY (
    SELECT array_to_json(array_agg(json))
    FROM (SELECT row_to_json(c) AS json
         FROM (SELECT nro_cliente AS _id,
                      nombre, apellido, direccion, activo,
                     (SELECT COALESCE(array_to_json(array_agg(t)),
                                      '[]'::json)
                      FROM (SELECT t.codigo_area, t.nro_telefono,
                                   t.tipo
                            FROM e01_telefono t
                            WHERE t.nro_cliente = c.nro_cliente) t
                     ) AS telefonos
               FROM e01_cliente c) c) j
    ) TO '/tmp/clientes.json' WITH (FORMAT text, HEADER FALSE);
```

Facturas

```
COPY (
    SELECT array_to_json(array_agg(json))
    FROM (SELECT row_to_json(f) AS json
         FROM (SELECT nro_factura AS _id,
                      fecha, total_sin_iva, iva, total_con_iva,
                      nro_cliente,
                     (SELECT COALESCE(array_to_json(array_agg(t)),
                                      '[]'::json)
                      FROM (SELECT df.codigo_producto,
                                   df.cantidad, df.nro_item
                            FROM e01_detalle_factura df
                            WHERE df.nro_factura = f.nro_factura) t
                     ) AS detalles
               FROM e01_factura f) f) j
    ) TO '/tmp/facturas.json' WITH (FORMAT text, HEADER FALSE);
```

2. Luego, copiamos los archivos generados en el contenedor `postgres-bd2` a `mongo-bd2`, copiándolos primero localmente para poder hacer la transición entre contenedores.

```
docker cp postgres-bd2:/tmp/productos.json ./productos.json
docker cp postgres-bd2:/tmp/clientes.json ./clientes.json
docker cp postgres-bd2:/tmp/facturas.json ./facturas.json

docker cp productos.json mongo-bd2:/tmp/productos.json
docker cp clientes.json mongo-bd2:/tmp/clientes.json
docker cp facturas.json mongo-bd2:/tmp/facturas.json
```

3. Por último, transformamos con `mongoimport` los archivos JSON en esquemas de Mongo:

```
docker exec mongo-bd2 mongoimport -h localhost:27017 -d admin -u
mongo -p docker --collection productos --jsonArray
/tmp/productos.json

docker exec mongo-bd2 mongoimport -h localhost:27017 -d admin -u
mongo -p docker --collection clientes --jsonArray
/tmp/clientes.json

docker exec mongo-bd2 mongoimport -h localhost:27017 -d admin -u
mongo -p docker --collection facturas --jsonArray
/tmp/facturas.json
```

4. En el caso particular de Facturas, una vez subido el esquema en MongoDB alteramos el atributo 'fecha' de *string* a *date*, por medio de MongoDB Shell (`mongosh`).

```
docker exec -i mongo-bd2 mongosh admin -u mongo -p docker << "EOM"
{
    db.facturas.find()
        .forEach(function(doc) {
            db.facturas.updateOne(
                {"_id": doc._id},
                {"$set": {"fecha": new Date(doc.fecha)}}
            );
        });
}
EOM
```

Consultas MongoDB

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

```
db.clientes.find(
    { nombre: "Wanda", apellido: "Baker" },
    { _id: 1,
      "telefonos.codigo_area": 1,
      "telefonos.nro_telefono": 1
    }
);
```

Resultado:

```
[
  {
    _id: 52,
    telefonos: [
      { codigo_area: 313, nro_telefono: 4578765 },
      { codigo_area: 593, nro_telefono: 4118559 }
    ]
  }
]
```

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

```
db.clientes.find(
    { _id: { $in: db.facturas.distinct("nro_cliente") } },
    { telefonos: 0 }
);
```

Resultados (3 de 98 elementos):

```
[
  {
    _id: 1,
    nombre: 'Xerxes',
    apellido: 'Hale',
    direccion: '129-5974 Suspendisse Ctra.',
    activo: 89
  },
  {
    _id: 2,
    nombre: 'Brent',
    apellido: 'Leblanc',
```

```
    direccion: 'Apartado núm.: 372, 5244 Nibh. ',
    activo: 58
  },
  {
    _id: 3,
    nombre: 'Kasper',
    apellido: 'Shannon',
    direccion: 'Apdo.:304-6908 Class Ctra.',
    activo: 38
  },
  ...
]
```

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

```
db.clientes.find(
    { _id: { $nin: db.facturas.distinct("nro_cliente") } },
    { telefonos: 0 }
);
```

Resultados (2 elementos):

```
[
  {
    _id: 58,
    nombre: 'Kaye',
    apellido: 'Stokes',
    direccion: 'Apdo.:227-9814 Natoque Carretera',
    activo: 27
  },
  {
    _id: 62,
    nombre: 'Ursula',
    apellido: 'Pollard',
    direccion: '971-7618 Tristique Carretera',
    activo: 118
  }
]
```

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

```
db.productos.find(
{ _id: { $in: db.facturas.distinct("detalles.codigo_producto") } }
);
```

Resultados (3 de 100 elementos):

```
[
  {
    _id: 1,
    marca: 'Dolor Sit Incorporated',
    nombre: 'return policy',
    descripcion: 'risus. In mi',
    precio: 443.88,
    stock: 404
  },
  {
    _id: 2,
    marca: 'A Felis Ullamcorper Company',
    nombre: 'sales tax',
    descripcion: 'Nunc mauris elit,',
    precio: 192.64,
    stock: 854
  },
  {
    _id: 3,
    marca: 'Elit A Corp.',
    nombre: 'household goods',
    descripcion: 'nec, euismod in,',
    precio: 554.79,
    stock: 123
  },
  ...
]
```

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

```
db.clientes.find({});
```

Resultados (3 de 100):

```
[
  {
    _id: 8,
    nombre: 'Mari',
    apellido: 'Savage',
    direccion: '2062 Id Avenida',
    activo: 49,
    telefonos: [
      { codigo_area: 629, nro_telefono: 4699941, tipo: 'F' },
      { codigo_area: 132, nro_telefono: 4962172, tipo: 'F' },
```

```
      { codigo_area: 212, nro_telefono: 4515936, tipo: 'F' }
    ]
  },
  {
    _id: 1,
    nombre: 'Xerxes',
    apellido: 'Hale',
    direccion: '129-5974 Suspendisse Ctra.',
    activo: 89,
    telefonos: [
      { codigo_area: 992, nro_telefono: 4241515, tipo: 'F' },
      { codigo_area: 513, nro_telefono: 4998612, tipo: 'M' }
    ]
  },
  {
    _id: 3,
    nombre: 'Kasper',
    apellido: 'Shannon',
    direccion: 'Apdo.:304-6908 Class Ctra.',
    activo: 38,
    telefonos: [
      { codigo_area: 915, nro_telefono: 4686947, tipo: 'F' }
    ]
  },
  ...
]
```

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

```
db.clientes.aggregate([
    {
        $lookup: {
            from: "facturas",
            localField: "_id",
            foreignField: "nro_cliente",
            as: "facturas"
        }
    },
    {
        $project: {
            nombre: 1,
            apellido: 1,
            direccion: 1,
            activo: 1,
```

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

Resultados (3 de 100 elementos):

```
[
  {
    _id: 8,
    nombre: 'Mari',
    apellido: 'Savage',
    direccion: '2062 Id Avenida',
    activo: 49,
    cantidad_facturas: 5
  },
  {
    _id: 1,
    nombre: 'Xerxes',
    apellido: 'Hale',
    direccion: '129-5974 Suspendisse Ctra.',
    activo: 89,
    cantidad_facturas: 4
  },
  {
    _id: 3,
    nombre: 'Kasper',
    apellido: 'Shannon',
    direccion: 'Apdo.:304-6908 Class Ctra.',
    activo: 38,
    cantidad_facturas: 3
  },
  ...
]
```

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

```
db.facturas.find(
    { nro_cliente: db.clientes.findOne(
                    { nombre: "Pandora", apellido: "Tate" })._id },
    { detalles: 0 }
);
```

Resultados (7 elementos):

```
[
  {
    _id: 1,
    fecha: ISODate("2016-05-28T00:00:00.000Z"),
    total_sin_iva: 294369.60099999997,
    iva: 21,
    total_con_iva: 356187.21721,
    nro_cliente: 4
  },
  {
    _id: 17,
    fecha: ISODate("2016-06-01T00:00:00.000Z"),
    total_sin_iva: 265939.596,
    iva: 21,
    total_con_iva: 321786.91116,
    nro_cliente: 4
  },
  {
    _id: 60,
    fecha: ISODate("2017-02-05T00:00:00.000Z"),
    total_sin_iva: 108632.29500000001,
    iva: 21,
    total_con_iva: 131445.07695000002,
    nro_cliente: 4
  },
  {
    _id: 61,
    fecha: ISODate("2016-12-10T00:00:00.000Z"),
    total_sin_iva: 141156.963,
    iva: 21,
    total_con_iva: 170799.92523,
    nro_cliente: 4
  },
  {
    _id: 66,
    fecha: ISODate("2016-05-23T00:00:00.000Z"),
    total_sin_iva: 77261.163,
    iva: 21,
    total_con_iva: 93486.00723,
    nro_cliente: 4
  },
  {
    _id: 55,
    fecha: ISODate("2016-06-21T00:00:00.000Z"),
```

```
    total_sin_iva: 132705.16199999998,
    iva: 21,
    total_con_iva: 160573.24601999996,
    nro_cliente: 4
  },
  {
    _id: 52,
    fecha: ISODate("2016-07-01T00:00:00.000Z"),
    total_sin_iva: 187440.55400000003,
    iva: 21,
    total_con_iva: 226803.07034000003,
    nro_cliente: 4
  }
]
```

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

```
var productosIds = db.productos.find({ marca: "In Faucibus Inc."
}, { _id: 1 }).toArray().map(p => p._id);
db.facturas.find(
    { "detalles.codigo_producto": { $in: productosIds } },
    { detalles: 0 }
);
```

Resultados (3 de 27 elementos):

```
[
  {
    _id: 6,
    fecha: ISODate("2017-03-28T00:00:00.000Z"),
    total_sin_iva: 524431.2535000001,
    iva: 21,
    total_con_iva: 634561.8167350001,
    nro_cliente: 1
  },
  {
    _id: 5,
    fecha: ISODate("2016-12-29T00:00:00.000Z"),
    total_sin_iva: 127843.699,
    iva: 21,
    total_con_iva: 154690.87579,
    nro_cliente: 20
  },
  {
    _id: 27,
```

```
    fecha: ISODate("2017-03-10T00:00:00.000Z"),
    total_sin_iva: 355809.31600000005,
    iva: 21,
    total_con_iva: 430529.27236000006,
    nro_cliente: 22
  },
  ...
]
```

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

```
db.clientes.aggregate([
    {
        $unwind: "$telefonos"
    },
    {
        $project: {
            _id: 1,
            nombre: 1,
            apellido: 1,
            direccion: 1,
            activo: 1,
            telefono: "$telefonos"
        }
    }
]);
```

Resultados (4 de 198 elementos):

```
[
  {
    _id: 8,
    nombre: 'Mari',
    apellido: 'Savage',
    direccion: '2062 Id Avenida',
    activo: 49,
    telefono:
            { codigo_area: 629, nro_telefono: 4699941, tipo: 'F' }
  },
  {
    _id: 8,
    nombre: 'Mari',
    apellido: 'Savage',
    direccion: '2062 Id Avenida',
    activo: 49,
```

```
    telefono:
            { codigo_area: 132, nro_telefono: 4962172, tipo: 'F' }
  },
  {
    _id: 8,
    nombre: 'Mari',
    apellido: 'Savage',
    direccion: '2062 Id Avenida',
    activo: 49,
    telefono:
            { codigo_area: 212, nro_telefono: 4515936, tipo: 'F' }
  },
  {
    _id: 1,
    nombre: 'Xerxes',
    apellido: 'Hale',
    direccion: '129-5974 Suspendisse Ctra.',
    activo: 89,
    telefono:
            { codigo_area: 992, nro_telefono: 4241515, tipo: 'F' }
  },
  ...
]
```

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

```
db.clientes.aggregate([
    {
        $lookup: {
            from: "facturas",
            let: { clienteId: "$_id" },
            pipeline: [
                {
                    $match: {
                        $expr: {
                            $eq: ["$nro_cliente", "$$clienteId"]
                        }
                    }
                }
            ],
            as: "facturas"
        }
    },
```

```
    {
        $unwind: {
           path: "$facturas",
           preserveNullAndEmptyArrays: true
        }
    },
    {
        $group: {
           _id: { _id: "$_id",
                  nombre: "$nombre", apellido: "$apellido" },
           gasto_total_con_iva: { $sum: { $ifNull:
                                ["$facturas.total_con_iva", 0] } }
        }
    },
    {
        $project: {
           _id: 0,
           nombre: "$_id.nombre",
           apellido: "$_id.apellido",
           gasto_total_con_iva: 1
        }
    }
]);
```

Resultados (3 de 100 elementos):

```
[
  {
    gasto_total_con_iva: 232099.71587,
    nombre: 'Rebekah',
    apellido: 'Cooke'
  },
  {
    gasto_total_con_iva: 351561.63042,
    nombre: 'Alexandra',
    apellido: 'Noel'
  },
  {
    gasto_total_con_iva: 259385.14205999998,
    nombre: 'Tanya',
    apellido: 'Reynolds'
  },
  ...
]
```

Vistas

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

```
db.createView("facturas_ordenadas", "facturas", [
    {
        $sort: { fecha: 1 }
    },
    {
        $project: { detalles: 0 }
    }
]);

db.facturas_ordenadas.find({});
```

Resultados (3 de 400 elementos):

```
[
  {
    _id: 239,
    fecha: ISODate("2016-03-01T00:00:00.000Z"),
    total_sin_iva: 49063.347,
    iva: 21,
    total_con_iva: 59366.64987,
    nro_cliente: 56
  },
  {
    _id: 167,
    fecha: ISODate("2016-03-02T00:00:00.000Z"),
    total_sin_iva: 230028.88400000002,
    iva: 21,
    total_con_iva: 278334.94964,
    nro_cliente: 38
  },
  {
    _id: 86,
    fecha: ISODate("2016-03-04T00:00:00.000Z"),
    total_sin_iva: 266064.201,
    iva: 21,
    total_con_iva: 321937.68321,
    nro_cliente: 9
  },
  ...
]
```

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

```
db.createView("productos_no_facturados", "productos", [
    {
        $lookup: {
            from: "facturas",
            localField: "_id",
            foreignField: "detalles.codigo_producto",
            as: "facturas"
        }
    },
    {
        $match: {
            facturas: { $eq: [] }
        }
    }
]);

db.productos_no_facturados.find({});
```

No hay resultados.

API

Se decidió diseñar la API REST HTTP creando una especificación OpenAPI. La API provee endpoints de clientes y productos con métodos para listar, crear, actualizar y eliminar los mismos. Se utilizó el formato JSON para transmitir los datos en el cuerpo de las peticiones y las respuestas.

Para implementar el servidor de la API se decidió utilizar ExpressJS desde Typescript y correrlo en el runtime NodeJS, ya que provee una manera rápida de construir servidores HTTP. Para conectarse a la base de datos se utilizaron los paquetes “pg” y “pg-pool” en caso de PostgreSQL y “mongodb” en el caso de MongoDB. Para parsear los cuerpos JSON de las peticiones, se utiliza “body-parser” y para validarlos se utilizó el paquete “express-openapi-validator” que permite reutilizar la especificación OpenAPI definida.

En el servidor que utiliza MongoDB, para seguir utilizando la numeración de clientes, productos y facturas predefinidas (un valor autoincremental), se tuvo que crear un documento para llevar la cuenta de estos contadores. Esto se logra con ejecutar el siguiente comando una única vez:

```
docker exec -i mongo-bd2 mongosh admin -u mongo -p docker << "EOM"
{
    db.createCollection("counters");
    let maxClientId = db.clientes.aggregate({
        $group: {
            _id: '',
            last: {
                $max: "$_id"
            }
        }
    }).toArray()[0].last;
    db.counters.insertOne({_id: "clientes", seq: maxClientId});

    let maxProductId = db.productos.aggregate({
        $group: {
            _id: '',
            last: {
                $max: "$_id"
            }
        }
    }).toArray()[0].last;
    db.counters.insertOne({_id: "productos", seq: maxProductId});

    let maxBillId = db.facturas.aggregate({
```

```
        $group: {
            _id: '',
            last: {
                $max: "$_id"
            }
        }
    }).toArray()[0].last;
    db.counters.insertOne({_id: "facturas", seq: maxBillId});
}
EOM
```

Luego, estas colecciones se utilizan para generar nuevos ID para las entidades que se creen.

Las instrucciones de como correr los servidores se puede conseguir en el `README.md` en el repositorio del proyecto.

Conclusión

A lo largo del trabajo práctico se desarrollaron las mismas consultas sobre bases de datos equivalentes, en modelo relacional y modelo no relacional. En base al modelado que se le dió a los esquemas, cada uno tuvo momentos de ventaja por sobre el otro.

Por ejemplo, al embeber los teléfonos en los usuarios para las consultas de MongoDB, seleccionar los datos de los clientes junto con sus teléfonos (Consulta 5) fue una tarea extremadamente sencilla. Este mismo procedimiento no podría realizarse en una base de datos relacional como PostgreSQL, ya que se guardarían tuplas con toda la información del usuario repetida por cada teléfono diferente que dicho usuario tenga, y por ello se prefiere tenerlo en tablas separadas.

Por otro lado, el poder de las juntas en una base de datos relacional hace que consultas como mostrar nombre y apellido de cada cliente junto con lo que gastó en total (Consulta 10) sea más simple de expresar en PostgreSQL que en MongoDB.

Al final del día, la decisión de qué base de datos utilizar dependerá del contexto de la aplicación y sus casos de uso, basándose en las necesidades específicas de cada proyecto.