Evaluación comparativa de Base de Datos Relacionales y Base de Datos No Relacionales

Agustina Drocco
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
agustina.drocco@fing.edu.uy

Agustina Salmantón
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
agustina.salmanton@fing.edu.uy

***Resumen*—En el siguiente proyecto se presenta la base de datos Northwind, la cuál contiene los datos de ventas de Northwind Treaders, una empresa ficticia de importación y exportación de alimentos especiales. Se representará la misma en versión relacional y documental y se evaluará su performance en un conjunto de consultas.**

I. Introducción

La base de datos Northwind es una base utilizada por Microsoft para demostrar las características de sus productos, como son SQL Server y Microsoft Access.
La base de datos contiene datos de ventas de Northwind Tarders, una empresa ficticia de importación y exportación de alimentos especiales.
En este documento se diseña, se prueba y se compara la implementación de esta base de datos con dos modelos diferentes: relacional y documental. Para el caso no relacional se utilizará MongoDB[1] y en el caso relacional PostgreSQL[2].
El objetivo principal es comparar ambos diseños a partir de un conjunto de consultas de interés sobre los mismos datos de prueba.
Se utiliza la herramienta JMeter de Apache[3] como prueba de carga para analizar y medir el tiempo de ejecución de estas consultas, también se realizan pruebas unitarias con las operaciones provistas por las distintas bases de datos.

II. Diseño e implementación

A partir del esquema de la base de datos Northwind, se realiza un listado de las consultas de más interés, que podrían ser utilizadas para una análisis OLAP.
Luego se diseña la base de datos no relacional teniendo en cuenta las consultas prioritarias y abstrayéndonos del diseño relacional.

II-A. Modelo de la base de datos relacional

Para el modelo de la base de datos relacional se utilizó el siguiente diagrama de estructura de la base de datos Northwind.

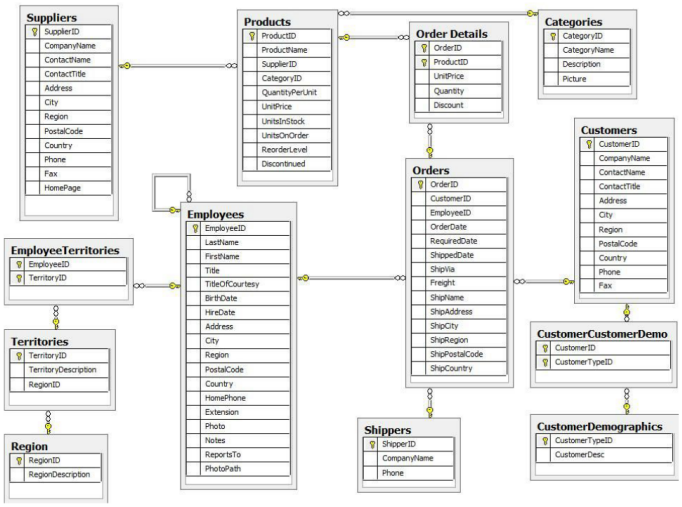

Figura 1: Estructura de la tabla de base de datos Northwind

A continuación se listan las entidades obtenidas:

- Suppliers
- EmployeeTerritories
- Territories
- Region
- Products
- Employees
- Order Details
- Orders
- Shippers
- Categories
- Customers
- CustomerCustomerDemo
- CustomerDemographics

Donde se tienen las siguientes dependencias:

- $\Pi_{RegionID}(Territories) \subseteq \Pi_{RegionID}(Region)$
- $\Pi_{TerritoryID}(EmployeeTerritories) \subseteq \Pi_{TerritoryID}(Territories)$
- $\Pi_{EmployeeID}(EmployeeTerritories) \subseteq \Pi_{EmployeeID}(Employees)$
- $\Pi_{EmployeeID}(Orders) \subseteq \Pi_{EmployeeID}(Employees)$
- $\Pi_{OrderID}(OrderDetails) \subseteq \Pi_{OrderID}(Orders)$
- $\Pi_{ShipVia}(Orders) \subseteq \Pi_{ShipperID}(Shippers)$

[1]https://www.mongodb.com/
[2]https://www.postgresql.org/
[3]https://jmeter.apache.org/

- $\Pi_{CustomerId}(CustomerCustomerDemo) \subseteq \Pi_{CustomerId}(Customers)$
- $\Pi_{CustomerTypeID}(CustomersCutomerDemo) \subseteq \Pi_{CustomerTypeID}(CustomerDemographics)$
- $\Pi_{SupplierID}(Products) \subseteq \Pi_{SupplierID}(Suppliers)$
- $\Pi_{ProductID}(OrderDetails) \subseteq \Pi_{ProductID}(Products)$
- $\Pi_{CategoryID}(Categories) \subseteq \Pi_{CategoryID}(Products)$

II-B. Modelado de la base de datos documental

Para el modelo de la base documental se diseñó agrupando según los siguientes criterios:

- Las colecciones correspondientes a Categories, Shippers, Suppliers quedan con los mismos atributos que en la base relacional.
- Las colecciones correspondientes a Employees, Orders, Products, Customers y Territories se modificaron de acuerdo a las consultas de interés:
  - Employees: se incluyen datos de Territories. Se observa el esquema en el Código 1.
  - Customers: se incluyen sus ordenes con el precio de las mismas. Se observa el esquema en el Código 2.
  - Orders: se incluyen datos relevantes de Customers, Employee, Shippers y Products. Se observa el esquema en el Código 3.
  - Products: se incluyen datos relevantes de Suppliers y Categories. Se observa el esquema en el Código 4.
  - Territories: se incluyen datos relevantes de Region. Se observa el esquema en el Código 5.

Código 1 Colección Employees

```
{
  EmployeeID:,
  LastName:,
  FirstName:,
  Title:,
  TitleOfCourtesy:,
  BirthDate:,
  HireDate:,
  Address:,
  City:,
  Region:,
  PostalCode:,
  Country:,
  HomePhone:,
  Extension:,
  Photo:,
  Notes:,
  PhotoPath:,
  Territories [{
     TerritoryID:,
     TerritoryDescription:,
  }]
}
```

Código 2 Colección Customers

```
{
   "CustomerID":,
   "CompanyName":,
   "ContactName":,
   "ContactTitle":,
   "Address":,
   "City":,
   "Region":,
   "PostalCode":,
   "Country":,
   "Phone":,
   "Fax":,
   "Orders": [{
      "OrderID":,
      "suma":,
   }]
}
```

Código 3 Colección Orders

```
{
   OrderID:,
   Curstomer {
      CustomerID:,
      CompanyName:,
      City:,
      Country:,
      Phone:,
      Address:,
      PostalCode:,
   }
   Employee {
      EmployeeID:,
      LastName:,
      FirstName:,
      Country:,
   }
   OrderDate:,
   OrderSales:,
   RequiredDate:,
   ShippedDate:,
   Shippers{
      ShipperID:,
      CompanyName:,
      Phone:,
   }
   Freight:,
   ShipName:,
   ShipAddress:,
   ShipCity:,
   ShipPostalCode:,
   ShipCountry:,
   Products {
      ProductID:,
      OrderDetails {
         UnitPrice:,
         Quantity:,
         Discount:,
      }
      ProductName:,
      CategoryID:,
      CategoryName:,
   }
}
```

Código 4 Colección Products

```
{
   ProductID:,
   ProductName:,
   QuantityPerUnit:,
   UnitPrice:,
   UnitsInStock:,
   UnitsOnOrder:,
   RecorderLevel:,
   Discontinued:,
   Suppliers{
      SupplierID:,
      CompanyName:,
      Country:,
   }
  Categories{
   CategoryID:,
   CategoryName:,
  }
}
```

Código 5 Colección Territories

```
{
   TerritoryID:,
   TerritoryDescription:,
   RegionID:,
   RegionDescription:,
}
```

III. Experimentación

Para la experimentación se seleccionó un conjunto de consultas a realizar en las bases de datos PostgreSql y MongoDB, las mismas implementadas para ambos modelos.
Los datos fuentes fueron obtenidos a través de archivos .csv, y por medio de la realización de scripts Python se obtienen los archivos JSON a ser cargados en MongoDB.
Luego mediante comandos y JMeter se obtienen los tiempos de ejecución para las consultas para luego ser analizados.

Tanto los scripts en Python, las consultas para PostgresSQL y MongoDB, los archivos JSON utilizados y el archivo .jmx de JMeter se encuentran disponibles en el repositorio Gitlab de la Facultad de Ingeniería de la UDELAR [4].

III-A. Datos de Prueba

La carga de la base de datos PostgreSQL se realizó a partir de un SQL publico en GIT [5].
A partir de la base PostgreSQL, se exportaron los archivos .csv que fueron utilizados. Los que no requerían transformación fueron cargados directamente en la base MongoDB, y otros fueron el archivo fuente para los scripts en Python con el objetivo de obtener documentos que siguieran los esquemas planteados anteriormente.

III-B. Consultas de prueba

Las consultas utilizadas para las pruebas son las siguientes:

- Consulta 1- Indicar cuál fue el producto más vendido por país.
- Consulta 2- Obtener la lista ordenada alfabéticamente de los productos junto con su categoría
- Consulta 3- Obtener información detallada de cada venta para poder emitir factura. (Orden, producto, cliente, empleado que lo vendió)
- Consulta 4- Dado una categoría y un año, devolver cuántas unidades se vendieron por semestre agrupado por país.
- Consulta 5- Clientes con sus gastos. Ordenado de forma descendente según el gasto.
- Consulta 6- Mostrar el empleado con mayor ventas por país. El resultado debe ser ordenado alfabéticamente por nombre de país.
- Consulta 7- Compañías con su cantidad de envíos por año.
- Consulta 8- Obtener los productos comprados por determinado cliente.

A continuación se muestran algunos ejemplos sobre su implementación en ambos modelos:

Código 6 Emitir factura - PostgreSQL

```
SELECT DISTINCT b.ship_name,
            b.ship_address,
            b.ship_city,
            b.ship_postal_code,
            b.ship_country,
            b.customer_ID,
            c.company_name,
            c.address,
            c.city,
            c.postal_code,
            c.country,
            concat(d.first_name, '␣', d.last_name)
               AS Salesperson,
            b.order_id,
            b.order_date,
            b.required_date,
            b.shipped_date,
            a.company_name,
            e.product_id,
            f.product_name,
            e.unit_price,
            e.quantity,
            e.discount,
            e.unit_price * e.quantity * (1 - e.discount)
               AS ExtendedPrice,
            b.freight
FROM shippers a
INNER JOIN orders b ON a.shipper_id = b.ship_via
INNER JOIN customers c ON c.customer_id = b.customer_id
INNER JOIN Employees d ON d.employee_id = b.employee_id
INNER JOIN Order_Details e ON b.order_id = e.order_id
INNER JOIN Products f ON f.product_id = e.product_id
ORDER BY b.ship_name;
```

Código 7 Emitir factura - MongoDB

```
db.orders.aggregate([
   {$project: {
     'ShipAddress': 1,
     'ShipCity': 1,
     'ShipPostalCode': 1,
     'ShipCountry': 1,
     'Customer.CustomerID': 1,
     'Customer.CompanyName': 1,
     'Customer.Address': 1,
     'Customer.City': 1,
     'Customer.PostalCode': 1,
     'Customer.Country': 1,
     'Employee.FirstName': 1,
     'Employee.LastName': 1,
     'OrderID':1,
     'OrderDate': 1,
     'RequiredDate': 1,
     'ShippedDate': 1,
     'Shippers.CompanyName': 1,
     'Products':1,
     'Freight': 1
   }}
])
```

Código 8 Mostrar el empleado con mayor ventas por país. El resultado debe ser ordenado alfabéticamente por nombre de país - PostgreSQL

```
WITH totals AS
  (SELECT e.country,
        e.first_name,
        e.last_name,
        SUM(unit_price * quantity * (1-discount)) AS Total
```

[4]https://gitlab.fing.edu.uy/agustina.salmanton/bdnr2021-group15/-/tree/master

[5]https://raw.githubusercontent.com/pthom/northwind_psql/master/northwind.sql

```
  FROM public.order_details od
  JOIN orders o ON od.order_id = o.order_id
  JOIN employees e ON e.employee_id = o.employee_id
  GROUP BY e.country,
         e.first_name,
         e.last_name),
   byRegion AS
 (SELECT *,
       ROW_NUMBER() OVER(PARTITION BY country
                   ORDER BY Total DESC) AS rn
  FROM totals)
SELECT country,
     first_name,
     last_name,
     total
FROM byRegion
WHERE rn = 1
ORDER BY country
```

Código 9 Mostrar el empleado con mayor ventas por país. El resultado debe ser ordenado alfabéticamente por nombre de país - MongoDB

```
db.orders.aggregate([{$group: {
 _id: {
      "EmployeeID": "$Employee.EmployeeID",
      "FirstName": "$Employee.FirstName",
      "LastName": "$Employee.LastName",
      "Country": "$Employee.Country",
    },
 ventas: { "$sum" :"$Products.OrderDetails.OrderSales"}
}}, {$group:
{
 _id: "$_id.Country",
 info:{$push: {
       "id":"$_id.EmployeeID",
       "name": { $concat: ["$_id.FirstName",'␣',
            "$_id.LastName"]},
       "ventas": "$ventas"
      }
 },
 max_ventas: {
  "$max" : "$ventas"
 }
}}, {$project: {
 'max_ventas': 1,
 'employee': {
    '$setDifference': [
       { '$map': {
          'input': '$info',
          'as': 'info',
          'in': {
             '$cond': [
                { '$eq': [ '$$info.ventas',
                  '$max_ventas' ] },
                '$$info.name',
                false
             ]
          }
       }},
       [false]
    ]
 }
}}, {$sort: {
 _id: 1
}}])
```

Código 10 Compañias con su cantidad de envíos por año - PostgreSQL

```
SELECT EXTRACT(YEAR
           FROM o.shipped_date) AS yearAcc,
     s.company_name,
     count(DISTINCT order_id) AS cant_pedidos
FROM public.shippers s
JOIN public.orders o ON s.shipper_id=o.ship_via
WHERE NOT o.shipped_date ISNULL
GROUP BY yearAcc,
       s.company_name
ORDER BY yearacc ASC,
       cant_pedidos DESC
```

Código 11 Compañias con su cantidad de envíos por año - MongoDB

```
db.orders.aggregate([{$match: {
 "ShippedDate":{$ne:""}
}}, {$addFields: {
 year_shipped: {
  $year:{
   $dateFromString: {dateString: "$ShippedDate"}
  }
 }
}}, {$group: {
 _id: {
  'ShipperID':'$Shippers.CompanyName',
  'year': '$year_shipped'
 },
 ordenes_unicas:
  {$addToSet: "$OrderID"}
}}, {$project: {
 '_id' : 1,
 'cant_pedidos' : {$size: "$ordenes_unicas"}
}}, {$sort: {
 '_id.year' : 1,
 'cant_pedidos': -1,

 }}])
```

Código 12 Obtener los productos comprados por determinado cliente - PostgreSQL

```
SELECT o.order_id,
     order_date,
     product_name,
     quantity
FROM PUBLIC.orders o
     JOIN PUBLIC.order_details od
       ON o.order_id = od.order_id
     JOIN PUBLIC.products p
       ON od.product_id = p.product_id
WHERE customer_id = 'VINET'
ORDER BY order_date DESC
```

Código 13 Obtener los productos comprados por determinado cliente - MongoDB

```
db.customers.aggregate([{$match: {
 "CustomerID": "VINET",
}}, {$unwind:
{
 path: "$Orders"
}}, {$lookup: {
 from: 'orders',
 localField: 'Orders.OrderID',
 foreignField: 'OrderID',
 as: 'Orders'
}}, {$project: {
 "_id":0,
 "OrderID": {$arrayElemAt: [ "$Orders.OrderID", 0 ] },
 "OrderDate": {$arrayElemAt: [ "$Orders.OrderDate", 0 ] },
 "Products": {
  "$map": {
     "input": "$Orders",
     "as": "m",
     "in": {
        "ProductName": "$$m.Products.ProductName",
        "Quantity": "$$m.Products.OrderDetails.Quantity"
     }
  }
  }
}}, {$sort: {
 "OrderDate": -1
}}])
```

III-C. Configuración del ambiente de pruebas

Las pruebas de ambos modelos fueron realizadas desde una misma computadora para ser consistentes con los resultados. Se utilizó el siguiente ambiente:

- PostgreSQL versión 5.4 utilizando PgAdmin4 para la base de datos relacional.
- MongoDB versión 4.4.5 para la base de datos documental.
- JMeter versión 5.4.1 para las pruebas de ejecución.
- Especificaciones del sistema:
  - Sistema operativo: macOS Big Sur
  - Procesador: 2,3 GHz Intel Core i5 de cuatro núcleos
  - Disco duro: Macintosh HD 256 GB
  - RAM: 8 GB 2133 MHz LPDDR3

III-D. Pruebas

Se realizan dos tipos de pruebas diferentes:

- A través de comandos
- Utilizando la herramienta JMeter

Estas pruebas fueron utilizadas teniendo como objetivo evaluar el desempeño y simular múltiples peticiones concurrentes a cada base de datos, ya que posibilita evaluar el tiempo de respuesta de las mismas.
La comparación realizada se centra en el tiempo de ejecución en milisegundos (ms) de cada prueba, donde además se repitió su ejecución, verificando los resultados obtenidos.

III-D1. Pruebas con comandos: Para las consultas sobre la base PostgreSQL se utiliza el comando **EXPLAIN ANALYSE**[6], este devuelve información sobre como se ejecuta la consulta por el Query Planner y además dos valores: Planning time y Execution time.
Estos dos valores sumados son utilizados para poder obtener el tiempo total de ejecución.

Para las consultas sobre la base MongoDB se utiliza el comando **explain('executionStats')**[7] que devuelve información similar al comando explicado anteriormente, el valor executionTimeMillis devuelve el tiempo total de la consulta (Planning y Execution).
En el cuadro 1 se muestra los tiempos obtenidos de cada consulta:

Cuadro I: Tiempos de ejecución pruebas con comandos

Consultas	***PostgreSQL***	***MongoDB***
Consulta 1	17.567	5
Consulta 2	4.362	6
Consulta 3	43.788	6
Consulta 4	9.3	5
Consulta 5	9.633	5
Consulta 6	12.025	110
Consulta 7	4.583	14
Consulta 8	13.105	8

[6]https://www.postgresql.org/docs/current/sql-explain.html
[7]https://docs.mongodb.com/manual/reference/explain-results/#mongodb-data-explain.executionStats

III-D2. Pruebas en JMeter: Se realiza un plan de prueba en JMeter para cada ambiente utilizando las siguientes configuraciones:

- 30 usuarios concurrentes
- Una petición por usuario
- Peticiones en un periodo de dos segundos

A continuación se muestran los resultados obtenidos a través de JMeter para las 300 peticiones de los usuarios, evaluando el desempeño sobre PostgreSQL y MongoDB. Se realizan pruebas sobre las consultas 3,6, 7 y 8 donde las mismas se ejecutan 4 veces.
El análisis se basa en el tiempo promedio obtenido. El mismo representa el tiempo promedio de ejecución y esta expresado en milisegundos. A su vez se muestra cual fue el tiempo de ejecución mínimo y máximo para brindar mas información.

De los cuadros II al IX se muestran los resultados obtenidos para las consultas sin índices tanto en PostgreSQL como MongoDB.

Cuadro II: Consulta 3 en PostgreSQL sin índices

Consulta 3	***Average***	***Min***	***Max***
Ejecución 1	246	43	670
Ejecución 2	209	50	433
Ejecución 3	241	50	589
Ejecución 4	200	46	335

Cuadro III: Consulta 3 en MongoDB sin índices

Consulta 3	***Average***	***Min***	***Max***
Ejecución 1	7	4	34
Ejecución 2	6	5	34
Ejecución 3	6	3	37
Ejecución 4	9	5	34

Cuadro IV: Consulta 6 en PostgreSQL sin índices

Consulta 6	***Average***	***Min***	***Max***
Ejecución 1	9	2	239
Ejecución 2	4	1	28
Ejecución 3	3	1	37
Ejecución 4	4	1	37

Cuadro V: Consulta 6 en MongoDB sin índices

Consulta 6	***Average***	***Min***	***Max***
Ejecución 1	7	5	31
Ejecución 2	6	4	42
Ejecución 3	6	4	32
Ejecución 4	9	4	109

Cuadro VI: Consulta 7 en PostgreSQL sin índices

Consulta 7	***Average***	***Min***	***Max***
Ejecución 1	5	0	185
Ejecución 2	3	0	104
Ejecución 3	2	0	37
Ejecución 4	3	0	25

Cuadro VII: Consulta 7 en MongoDB sin índices

Consulta 7	***Average***	***Min***	***Max***
Ejecución 1	92	24	233
Ejecución 2	74	22	125
Ejecución 3	71	18	155
Ejecución 4	81	21	172

Cuadro VIII: Consulta 8 en PostgreSQL sin índices

Consulta 8	***Average***	***Min***	***Max***
Ejecución 1	11	1	34
Ejecución 2	3	0	27
Ejecución 3	2	0	37
Ejecución 4	3	0	25

Cuadro IX: Consulta 8 en MongoDB sin índices

Consulta 8	***Average***	***Min***	***Max***
Ejecución 1	0	0	46
Ejecución 2	0	0	32
Ejecución 3	0	3	36
Ejecución 4	0	0	30

Luego se realizan las mismas pruebas agregando los siguientes índices:
Para PostgreSQL:

- En la tabla *order_details* se agregan los índices por *order_id* y *product_id*
- En la tabla *customers* se agrega el índice por *customer_id*.
- En la tabla *orders* se agregan los índices por *order_id* y *customer_id*
- En la tabla *products* se agrega el índice por *product_id*.

Para MongoDB:

- En la colección orders se agregaron índices por OrderID, CustomerID y ProductID.
- En la colección customers se agrego un índice por CustomerID.
- En la colección products se agrego un índice por ProductID.

De los cuadros X al XVII se muestran los resultados obtenidos.

Cuadro X: Consulta 3 en PostgreSQL con índices

Consulta 3	***Average***	***Min***	***Max***
Ejecución 1	362	67	954
Ejecución 2	232	51	564
Ejecución 3	214	50	409
Ejecución 4	210	46	408

Cuadro XI: Consulta 3 en MongoDB con índices

Consulta 3	***Average***	***Min***	***Max***
Ejecución 1	6	3	30
Ejecución 2	7	3	37
Ejecución 3	6	3	37
Ejecución 4	7	3	42

Cuadro XII: Consulta 6 en PostgreSQL con índices

Consulta 6	***Average***	***Min***	***Max***
Ejecución 1	11	12	268
Ejecución 2	4	1	35
Ejecución 3	4	1	26
Ejecución 4	4	1	27

Cuadro XIII: Consulta 6 en MongoDB con índices

Consulta 6	***Average***	***Min***	***Max***
Ejecución 1	6	3	30
Ejecución 2	6	4	42
Ejecución 3	6	4	44
Ejecución 4	5	4	34

Cuadro XIV: Consulta 7 en PostgreSQL con índices

Consulta 7	***Average***	***Min***	***Max***
Ejecución 1	3	0	49
Ejecución 2	2	0	31
Ejecución 3	2	0	37
Ejecución 4	2	0	26

Cuadro XV: Consulta 7 en MongoDB con índices

Consulta 7	***Average***	***Min***	***Max***
Ejecución 1	2	0	35
Ejecución 2	2	0	31
Ejecución 3	2	19	149
Ejecución 4	3	22	180

Cuadro XVI: Consulta 8 en PostgreSQL con índices

Consulta 8	***Average***	***Min***	***Max***
Ejecución 1	2	0	28
Ejecución 2	3	0	104
Ejecución 3	2	0	37
Ejecución 4	3	0	25

Cuadro XVII: Consulta 8 en MongoDB con índices

Consulta 8	***Average***	***Min***	***Max***
Ejecución 1	5	0	187
Ejecución 2	2	0	37
Ejecución 3	0	3	36
Ejecución 4	0	0	24

IV. Conclusiones y trabajo futuro

Primero se observa que los índices creados en ambas bases no variaron el tiempo de ejecución, esto puede deberse a que las tablas contienen pocos registros y la búsqueda se realiza de forma rápida tanto haciendo un Scan secuencial o por índice.

Para realizar la evaluación comparativa, se basó en los resultados obtenidos en las pruebas. Se observa que para las mismas consultas MongoDB tiene mejor desempeño en la mayoría de los casos pero no por una gran diferencia respecto a la base PostgreSQL. Esto puede deberse a que la base de datos original de NorthWind no contiene una gran cantidad de datos, y por lo tanto para notar una mayor diferencia se debería tener una base de datos más realista de ventas de productos pertenecientes a una empresa.

La diferencia de performance más grande puede reflejarse en la consulta 3, que corresponde a 'Emitir Factura'. Suponiendo que esta consulta es una de las más recurrentes para la realidad, MongoDB responde mejor a los tiempos de performance dado que contiene toda la información necesaria en el mismo documento, mientras que en la base de datos relacional se debe realizar JOINS entre todas las tablas para unificar los datos.

Respecto a las medidas tomadas en JMeter para la base documental, al usar Apache Groovy para las consultas en MongoDB no se puede asegurar que no haya un mínimo overhead de operaciones que puedan impactar en el tiempo de la consulta.

Respecto al espacio que ocupa la base, MongoDB al tener los objetos desnormalizados ocupa más espacio que una base de datos relacional.

Con los resultados obtenidos, aunque se observa una mínima diferencia con MongoDB, esta no es tan notoria como para poder concluir cual modelo es el mejor para esta realidad.

Como trabajo a futuro se plantea realizar estas pruebas de desempeño en una computadora con mayores recursos que se asemejen a lo que podría ser un servidor real para esta base de datos, y fundamentalmente para obtener medidas representativas se haría sobre una base con grandes volúmenes de datos, donde pudiera apreciarse una mayor diferencia entre ambos modelos.