


Departamento de Computación,
Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires

# Trabajo Práctico 2

## Bases de Datos

Primer Cuatrimestre de 2016



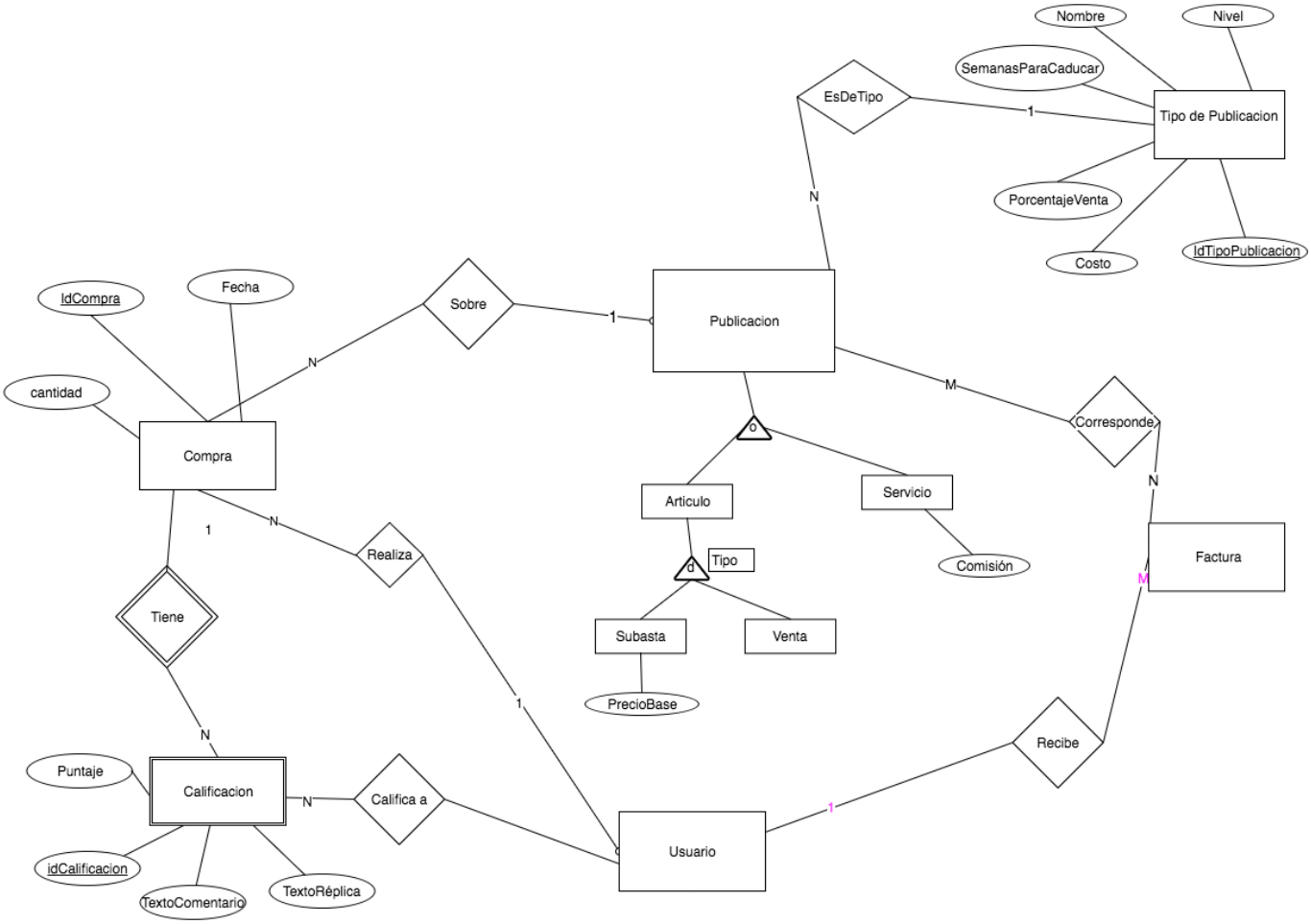
| Apellido y Nombre | LU | E-mail |
|---|---|---|
| Federico Hosen | 825/12 | federico.hosen@gmail.com |
| Martin 'Marto' Caravario | 470/12 | martin.caravario@gmail.com |
| Guido Rajngewerc | 379/12 | guido.raj@gmail.com |
| Christian Russo | 679/10 | christian.russo8@gmail.com |

# 1. Introducción

En el siguiente trabajo se desarrolla un modelo de Bases de Datos NoSQL con MongoDB teniendo en cuenta el problema descripto en el TP1. Para esto, se utilizó una sección del DER completo del TP1 y se diseñaron los documentos correspondientes para realizar las consultas pedidas en el enunciado del TP2.

# 2. Parte 1 - Diseño

## 2.1. Diagrama Entidad relacion




Las consultas pedidas solamente necesitan de usuarios, ventas, reputaciones, calificaciones, comisiones y facturas. Por lo tanto usamos:

- Para usuarios y ventas, las tablas de Usuarios, Publicaciones y Compra
- Para reputaciones de usuarios, las tablas de Calificación y Usuario
- Para comisiones, usamos las tablas de Publicación y Tipo de Publicación.
- Para monto de facturas, usamos las tabla de Facturas

## 2.2. Colecciones

A continuación, mostramos las colecciones que utilizamos para resolver las consultas propuestas en el enunciado

### 2.2.1. Publicaciones

```
{
  "Id": 65,
  "TipoPublicacion": "Servicio"
}
```

### 2.2.2. Compra

```
{
  "idCompra": 656897,
  "idUsuarioComprador": 65468,
  "Fecha": "2-06-2014",
  "Cantidad": 2,
  "Publicacion": {
    "idPublicacion": 2365,
    "idUsuario": 123,
    "Precio": 3655,
    "PorcentajeVenta": 20 //entre 0 y 100
  },
  "CalificacionDelVendedor": 8,
  "CalificacionDelComprador": 6
}
```

### 2.2.3. Factura

```
{
  "idFactura": 687468,
  "IdUsuario": 654,
  "Fecha": "16-9-2015",
  "TotalAPagar": 5624,
  "estoyPagando": [
    {
      "idPublicacion": 54,
      "TipoSuscripcion": "Rubi"
    },
    {
      "idPublicacion": 54,
      "TipoSuscripcion": "Libre"
     }
  ]
}
```

## 2.3. Migración datos SQL a Colecciones

Para migrar los datos de SQL (MySQLWorkwench) al formato de JSON que se necesita en MongoDB lo que hicimos fue exportar los datos directo desde MySQLWorkwench, pero estos se exportan en JSON plano, es decir no quedaba formateado de la forma que nosotros queriamos.

Para solucionar este problema, realizamos unos algoritmos en python para formatearlos a mano. Estos algoritmos reciben como input el JSON plano y devuelven un JSON formateado para MongoDB

### 2.3.1. Parser para el JSON de la compra

```
import json
from pprint import pprint

with open(raw_input()) as data_file:
    data = json.load(data_file)
for a in data:
  print "{"
  print '"idCompra": ' , a['idCompra'], ","
  print '"idUsuario": ', a['idUsuario'],","
  print '"fecha": ', '"',a['fecha'],'"',","
  print '"cantidad": ', a['cantidad'],","
  print '"Publicacion": {'
  print '"idPublicacion": ', a['idPublicacion'],","
  print '"idUsuario": ', a['idUsuario'],","
  print '"Precio": ', a['precio'],","
  print '"PorcentajeVenta": ', a['porcentajeVenta']
  print "}" ","
  print '"calificacionVendedor": ', a['CalificacionDelVendedor'], ','
  print '"calificacionComprador": ', a['CalificacionDelComprador'], ''
  print '}'
```

### 2.3.2. Parser para el JSON de la Factutra

```
import json
from pprint import pprint

with open('factura_sql.json') as data_file:
    data = json.load(data_file)

c = {}
for a in data:
  c.setdefault(a["idFactura"],[]).append(a)

for a in c:
  print '{'
  print '"idFactura": ', c[a][0]['idFactura'], ","
  print '"IdUsuario": ',c[a][0]['idUsuario'], ","
  print '"Fecha": ', '"',c[a][0]['fecha'],'"', ","
  total = 0
  for x in c[a]:
    total = total + c[a][c[a].index(x)]['totalAPagar']
  print '"TotalAPagar": ',total, ","
  print '"estoyPagando": ['
  i = 1
  for x in c[a]:
    print "{"
    print '"idPublicacion":', c[a][c[a].index(x)]['idPublicacion'], ","
    print '"TipoSuscripcion":','"',c[a][c[a].index(x)]['nombre'], '"'
    if i == len(c[a]):
      print "}"
    else:
      print "},"
    i = i+1
  print ']'
  print '}'
```

**Nota:** para la colección de publicaciones no tuvimos que hacer ningún algoritmo.

## 2.4. Consultas SQL

A continuación listamos las consultas realizadas para conseguir los datos necesarios para nuestras colecciones:

### 2.4.1. SQL de Publicaciones

```
select p.idPublicacion,
CASE
  when s.idPublicacion is not null and
  (v.idPublicacion is not null or su.idPublicacion is not null) then "mixto"
  when s.idPublicacion is not null then "servicio"
    when v.idPublicacion is not null then "articulo"
    when su.idPublicacion is not null then "articulo"
END
as TipoPublicacion
from publicacion p
left join servicio s on s.idPublicacion = p.idPublicacion
left join venta v on v.idPublicacion = p.idPublicacion
left join subasta su on su.idPublicacion = p.idPublicacion
```

### 2.4.2. SQL de Compras

```
select c.idCompra, c.idUsuario as idUsarioComprador,p.idUsuario as idUsuarioVendedor,
c.fecha, c.cantidad, c.idPublicacion, p.precio,
tp.costo, tp.porcentajeVenta, tp.nombre, ca.puntaje
from compra c
join publicacion p

on c.idPublicacion = p.idPublicacion
join tipo_de_publicacion tp
on p.idTipoDePublicacion = tp.idTipoPublicacion
join calificacion ca
on c.idCompra = ca.idCompra
```

### 2.4.3. SQL de Facturas

```
select f.idFactura, p.idUsuario, f.fecha, f.totalAPagar, p.idPublicacion, tp.nombre

from factura f

inner join corresponde c
on c.idFactura = f.idFactura

inner join publicacion p
on c.idPublicacion = p.idPublicacion

inner join tipo_de_publicacion tp
on tp.idTipoPublicacion = p.idTipoDePublicacion
```

### 2.5. Implementación MongoDB

Para implementar la base de datos de documentos en mongoDB utilizamos los JSON de las colecciones descriptos anteriormente y los importamos de la siguiente forma:

```
mongoimport --db tp2 --collection compras --drop --file ./compra_mongo.json
mongoimport --db tp2 --collection facturas --drop --file ./factura_mongo.json
mongoimport --db tp2 --collection publicacion --drop --file ./publicacion_mongo.json
```

## 3. Parte 2 - Map Reduce

### 3.1. Ejercicio 1

El importe total de ventas por usuario.

```
var ej1_m = function(){
  emit(this.Publicacion.idUsuario, this.Publicacion.Precio * this.cantidad)
}

var ej1_r = function(k, vs){
  return Array.sum(vs);
}
```

### 3.2. Ejercicio 2

La reputación histórica de cada usuario según la calificación.

```
var ej2_m = function(){
  if(this.calificacionVendedor != null) emit(this.idUsuario, this.calificacionVendedor);
  if(this.calificacionComprador != null) emit(this.Publicacion.idUsuario, this.calificacionComprador);
}

var ej2_r = function(k, vs){
  return ( Array.sum(vs) / vs.length);
}
```

### 3.3. Ejercicio 3

Las operaciones con comisión más alta.

```
var ej3_m = function(){
  var getComision = function(compra){
   var subs = compra.Publicacion;
    return (subs.Precio * compra.cantidad) * (subs.PorcentajeVenta / 100);
  };

  emit("todos", {
  'idCompra': this.idCompra,
  'comision': getComision(this)});
};

var ej3_r = function(k, vs){
  var max = Math.max(...vs.map(function(v){
      return v.comision;
  }));
  return {"resultado": vs.filter(function(v){
    return v.comision >= max;
  }), "max": max};
};
```

## 3.4. Ejercicio 4

El monto total facturado por año.

```
var ej4_m = function(){
  var getAno = function(fecha){
    return fecha.trim().substring(0,4);
  }
  emit(getAno(this.Fecha), this.TotalAPagar)
}

var ej4_r = function(k, vs){
  return Array.sum(vs);
}
```

## 3.5. Ejercicio 5

El monto total facturado por año de las operaciones pertenecientes a usuarios con suscripciones Rubi Oriente.

```
var ej5_m1 = function(){

  var esRuby = function(e, i, arr){
    return e.TipoSuscripcion.trim() == "RubiDeOriente"
  };

  var getAno = function(fecha){
    return fecha.trim().substring(0,4);
  }

  emit(
    {
      "ano": getAno(this.Fecha),
      "usuario": this.IdUsuario
    },
    {
      "totalAPagar": this.TotalAPagar,
      "esRuby": this.estoyPagando.some(esRuby)
    });
};

var ej5_r1 = function(k, vs){
  if (vs.some(function(a){
    return a.esRuby;
  }))
  {
    return Array.sum(vs.map(function(e){return e.totalAPagar}));
  }
}


var ej5_m2 = function(){
  if (this.value.esRuby)
  {
    emit(this._id.ano, this.value.totalAPagar);
  }
}

var ej5_r2 = function(k, vs){
  return Array.sum(vs);
}
```

## 3.6. Ejercicio 6

El total de publicaciones por tipo de publicación (productos, servicios o mixtas).

```
var ej6_m = function(){
  emit(this.TipoPublicacion, 1);
}

var ej6_r = function(k,vs){
  return  vs.length;
};
```

Listing 1: Ejercicio 6

# 4. Parte 3 - Sharding

## 4.1. Introducción

Utilizamos 5 shards para hacer las pruebas y a la colección de publicaciones le agregamos un atributo extra ïdUsuarioVendedor"para poder usarlo como atributo clave a la hora de hacer experimentos de sharding. Cada 200000 insert hacemos un output en un archivo de

db.publicaciones.getShardDistribution() y sh.status() y con estos datos hicimos los gráficos.

## 4.2. Inserción de datos en la base de datos

Para insertar 500000 registros en la base de datos lo que hicimos fue generar un script en Python que nos devuelva 500000 publicaciones.

El pseudocodigo para la generación de los 500000 registros:

```
import json
from pprint import pprint
import random
from random import randint


foo = ['articulo', 'servicio', 'mixto']
print 'var pubs = ['
for x in range(1,500000):
  print '{'
  print '"idPublicacion":', x,","
  print '"TipoPublicacion":', ",",random.choice(foo),","
  print '"idUsuarioVendedor":',randint(0,8000)
  print '},'
print ']'
```

A continuación, para insertar estos datos en la base de datos de mongo, hicimos un scrpit de JavaScript para correrlo desde el Shell de mongo y que pueda importar todos estos datos en los distintos shards

```
var collection = "publicaciones";

pubs.forEach(function(e, i){
    if (i % 20000 == 0){
        db[collection].getShardDistribution();
    }
    db[collection].insert(e);
});
```

Listing 2: Ejercicio 6

# 5. Índice Simple

## 5.1. Creación de Shards

Para la creacion de los Shards con Indice Simple corrimos los siguientes scripts:

Creamos los shards:

```
mkdir -p ./data/localhost10000
mongod --rest --shardsvr --port 10000 --dbpath data/localhost10000
--logpath data/localhost10000/log

mkdir -p ./data/localhost10001
mongod --rest --shardsvr --port 10001 --dbpath data/localhost10001
--logpath data/localhost10001/log

mkdir -p ./data/localhost10002
mongod --rest --shardsvr --port 10002 --dbpath data/localhost10002
--logpath data/localhost10002/log

mkdir -p ./data/localhost10003
mongod --rest --shardsvr --port 10003 --dbpath data/localhost10003
--logpath data/localhost10003/log

mkdir -p ./data/localhost10004
mongod --rest --shardsvr --port 10004 --dbpath data/localhost10004
--logpath data/localhost10004/log
```

Creamos el config server

```
mkdir -p ./data/localhost10005
mongod --rest --configsvr --port 10005 --dbpath data/localhost10005
--logpath data/localhost10005/log
```

Creaamos routing service

```
 mongos --port 10006 --configdb localhost:10005 > run_routing_service_log
```

Agregamos los shards

```
mongo localhost:10006
use admin
db.runCommand({addshard:"localhost:10000", name:"shard10000"});
db.runCommand({addshard:"localhost:10001", name:"shard10001"});
db.runCommand({addshard:"localhost:10002", name:"shard10002"});
db.runCommand({addshard:"localhost:10003", name:"shard10003"});
db.runCommand({addshard:"localhost:10004", name:"shard10004"});
```

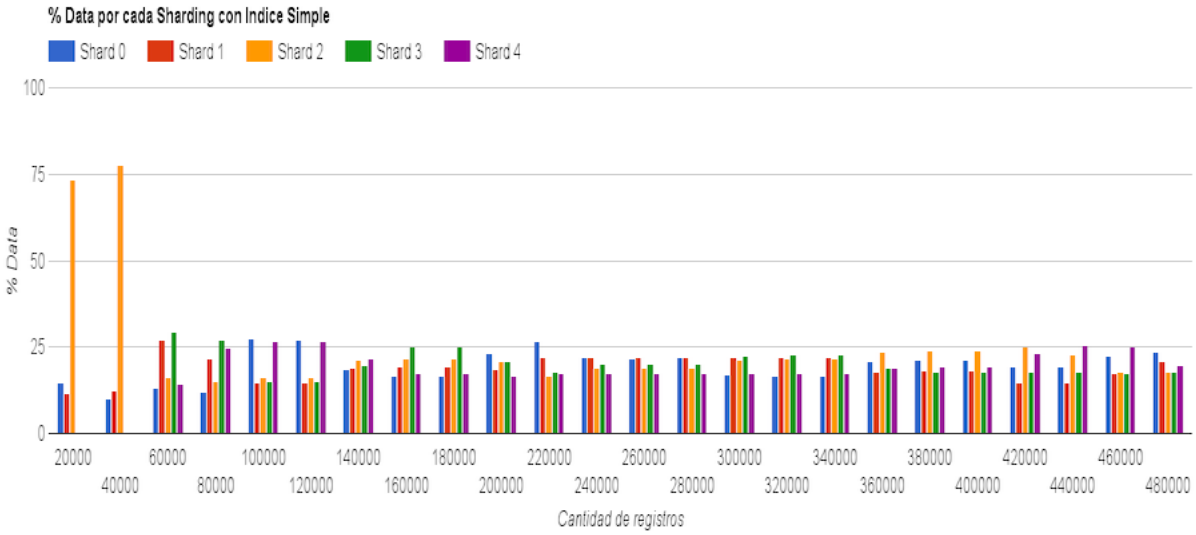
Configuramos los Shards:

```
use test_sharding
sh.enableSharding("test_sharding")
db.people.ensureIndex({"zip": 1})
```

## 5.2. Evolucion de Shards

A continuación se muestra un cuadro donde se encuentran los datos obtenidos del análisis de los Shards. Lo que hicimos fue ver el getShardDistribution() cada 20000 inserciones y mostrar el porcentaje de distrubución de los datos en los distintos Shards. Como ya fue explicado, utilizamos 5 Shards y en el cuadro se ve el porcentaje de datos que contiene cada Shard cada 20000 insert desde 0 hasta 500000 registros. Para este análisis utilizamos índices simples sobre la columna ïdUsuarioVendedor"del documento de Publicaciones.

| | Shard 0 | Shard 1 | Shard 2 | Shard 3 | Shard 4 |
|---|---|---|---|---|---|
| 20000 | 14,86 | 11,66 | 73,47 | 0 | 0 |
| 40000 | 10,2 | 12,32 | 77,47 | 0 | 0 |
| 60000 | 13,06 | 27,03 | 16,22 | 29,27 | 14,39 |
| 80000 | 11,95 | 21,61 | 14,91 | 26,88 | 24,64 |
| 100000 | 27,26 | 14,59 | 16,24 | 15,05 | 26,83 |
| 120000 | 27,17 | 14,61 | 16,32 | 15,12 | 26,76 |
| 140000 | 18,4 | 18,91 | 21,16 | 19,88 | 21,63 |
| 160000 | 16,65 | 19,31 | 21,61 | 25,05 | 17,35 |
| 180000 | 16,62 | 19,31 | 21,66 | 25,01 | 17,37 |
| 200000 | 23,06 | 18,63 | 20,9 | 20,69 | 16,68 |
| 220000 | 26,48 | 21,88 | 16,59 | 17,68 | 17,34 |
| 240000 | 21,83 | 21,92 | 18,91 | 20,04 | 17,28 |
| 260000 | 21,81 | 21,92 | 18,89 | 20,05 | 17,3 |
| 280000 | 21,82 | 21,91 | 18,88 | 20,06 | 17,3 |
| 300000 | 16,95 | 21,84 | 21,41 | 22,52 | 17,26 |
| 320000 | 16,6 | 21,93 | 21,51 | 22,62 | 17,32 |
| 340000 | 16,6 | 21,94 | 21,49 | 22,63 | 17,31 |
| 360000 | 20,85 | 17,71 | 23,5 | 18,87 | 19,04 |
| 380000 | 21,15 | 17,96 | 23,84 | 17,7 | 19,32 |
| 400000 | 21,15 | 17,96 | 23,82 | 17,69 | 19,35 |
| 420000 | 19,19 | 14,78 | 25,07 | 17,68 | 23,26 |
| 440000 | 19,21 | 14,69 | 22,84 | 17,65 | 25,58 |
| 460000 | 22,33 | 17,52 | 17,72 | 17,3 | 25,1 |
| 480000 | 23,7 | 20,92 | 17,72 | 17,83 | 19,8 |




En el grafico se puede ver que ....

# 6. Indice Hash

## 6.1. Creacion de Shards

Para la creacion de los Shards con Indice Hash corrimos los siguientes scripts:

Creamos los shards:

```
mkdir -p ./data/localhost10000
mongod --rest --shardsvr --port 10000 --dbpath data/localhost10000
--logpath data/localhost10000/log

mkdir -p ./data/localhost10001
mongod --rest --shardsvr --port 10001 --dbpath data/localhost10001
--logpath data/localhost10001/log

mkdir -p ./data/localhost10002
mongod --rest --shardsvr --port 10002 --dbpath data/localhost10002
--logpath data/localhost10002/log

mkdir -p ./data/localhost10003
mongod --rest --shardsvr --port 10003 --dbpath data/localhost10003
--logpath data/localhost10003/log

mkdir -p ./data/localhost10004
mongod --rest --shardsvr --port 10004 --dbpath data/localhost10004
--logpath data/localhost10004/log
```

Creamos el config server

```
mkdir -p ./data/localhost10005
mongod --rest --configsvr --port 10005 --dbpath data/localhost10005
--logpath data/localhost10005/log
```

Creaamos routing service

```
 mongos --port 10006 --configdb localhost:10005 > run_routing_service_log
```

Agregamos los shards

```
mongo localhost:10006
use admin
db.runCommand({addshard:"localhost:10000", name:"shard10000"});
db.runCommand({addshard:"localhost:10001", name:"shard10001"});
db.runCommand({addshard:"localhost:10002", name:"shard10002"});
db.runCommand({addshard:"localhost:10003", name:"shard10003"});
db.runCommand({addshard:"localhost:10004", name:"shard10004"});
```

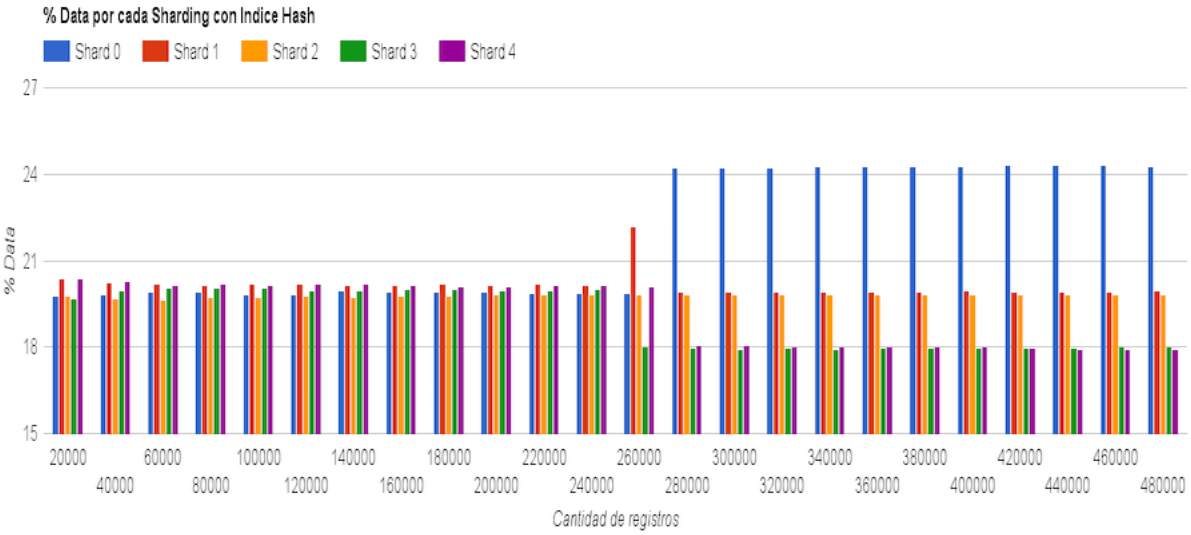
Configuramos los Shards:

```
use test_sharding
sh.enableSharding("test_sharding")
db.people.ensureIndex({"_id": "hashed"})
```

## 6.2. Evolución de Shards

A continuación se muestra un cuadro donde se encuentran los datos obtenidos del análisis de los Shards. Lo que hicimos, fue ver el getShardDistribution() cada 20000 inserciones y mostrar el porcentaje de distrubución de los datos en los distintos Shards. Como ya fue explicado, utilizamos 5 Shards y en el cuadro se ve el porcentaje de datos que contiene cada Shard cada 20000 insert desde 0 hasta 500000 registros. Para este análisis utilizamos índices hash sobre la columna ïdUsuarioVendedor"del documento de Publicaciones.

| | Shard 0 | Shard 1 | Shard 2 | Shard 3 | Shard 4 |
|---|---|---|---|---|---|
| 20000 | 19,77 | 20,38 | 19,77 | 19,68 | 20,38 |
| 40000 | 19,82 | 20,22 | 19,68 | 19,98 | 20,27 |
| 60000 | 19,93 | 20,19 | 19,64 | 20,06 | 20,15 |
| 80000 | 19,9 | 20,12 | 19,71 | 20,05 | 20,2 |
| 100000 | 19,84 | 20,21 | 19,74 | 20,05 | 20,14 |
| 120000 | 19,84 | 20,18 | 19,78 | 19,98 | 20,2 |
| 140000 | 19,94 | 20,16 | 19,74 | 19,96 | 20,17 |
| 160000 | 19,93 | 20,12 | 19,77 | 19,99 | 20,16 |
| 180000 | 19,92 | 20,19 | 19,76 | 19,99 | 20,11 |
| 200000 | 19,92 | 20,16 | 19,84 | 19,95 | 20,11 |
| 220000 | 19,87 | 20,17 | 19,83 | 19,98 | 20,12 |
| 240000 | 19,87 | 20,15 | 19,83 | 20,01 | 20,12 |
| 260000 | 19,86 | 22,16 | 19,83 | 18,02 | 20,1 |
| 280000 | 24,21 | 19,93 | 19,84 | 17,95 | 18,04 |
| 300000 | 24,22 | 19,93 | 19,83 | 17,94 | 18,06 |
| 320000 | 24,23 | 19,92 | 19,83 | 17,96 | 18,03 |
| 340000 | 24,26 | 19,93 | 19,82 | 17,94 | 18,03 |
| 360000 | 24,27 | 19,93 | 19,82 | 17,97 | 17,99 |
| 380000 | 24,27 | 19,92 | 19,82 | 17,97 | 17,99 |
| 400000 | 24,28 | 19,94 | 19,82 | 17,95 | 17,99 |
| 420000 | 24,31 | 19,92 | 19,81 | 17,97 | 17,96 |
| 440000 | 24,31 | 19,93 | 19,82 | 17,97 | 17,94 |
| 460000 | 24,3 | 19,92 | 19,82 | 18 | 17,94 |
| 480000 | 24,28 | 19,94 | 19,81 | 18 | 17,94 |




En el gráfico se puede ver que ....

# 7. Indice Simple vs Indice Hash

aca escribir la conclusion