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

Trabajo Práctico 2

Bases de Datos

Primer Cuatrimestre de 2016

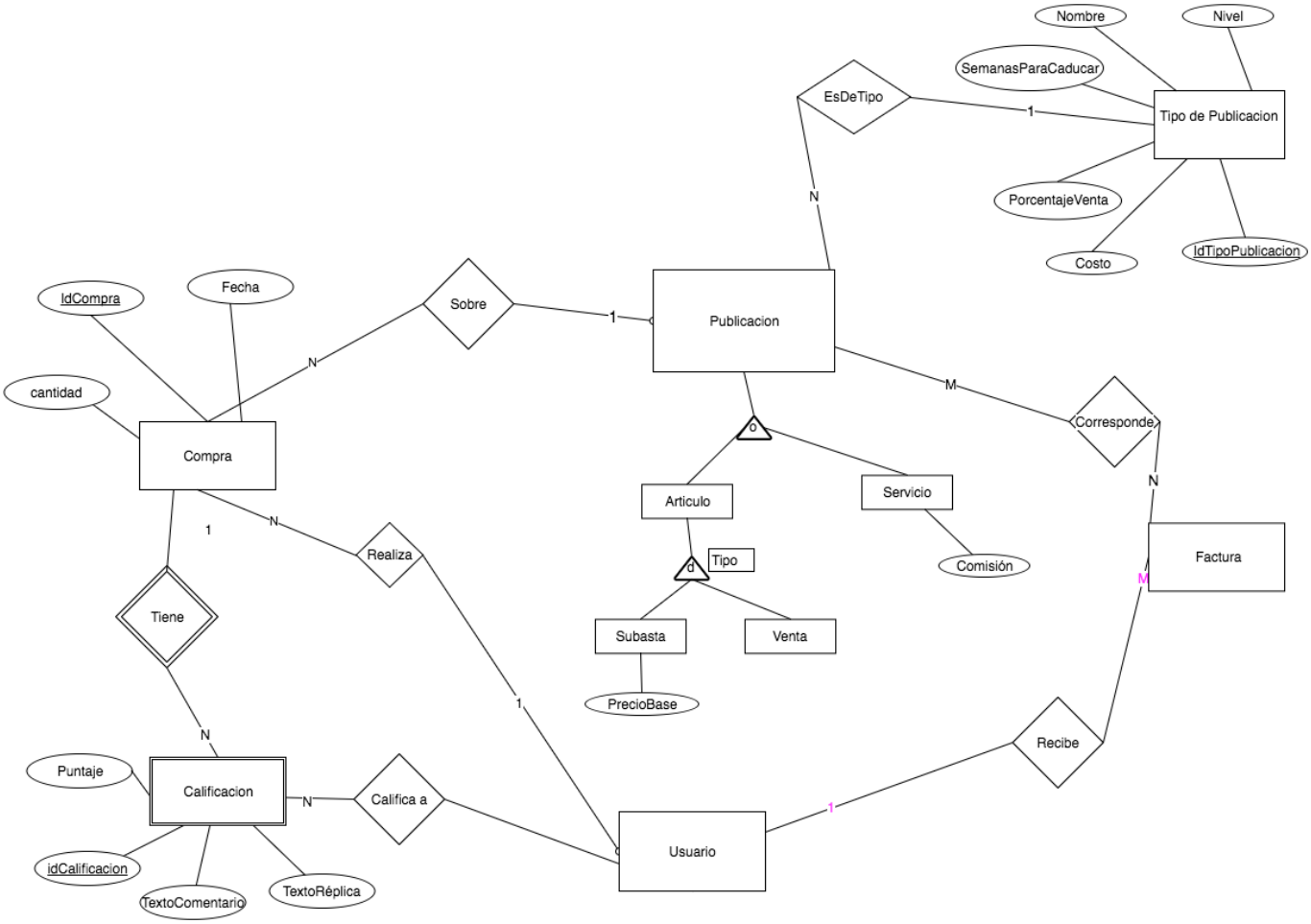
Apellido y Nombre	LU	E-mail
Federico Hosen	825/12	federico.hosen@gmail.com
Martin 'Marto' Caravario	470/12	martin.caravario@gmail.com
Guido Rajngewerc	379/12	guido.raj@gmail.com
Christian Russo	679/10	christian.russo8@gmail.com

1. Introducción

En el siguiente trabajo se desarrolla un modelo de Bases de Datos NoSQL con MongoDB teniendo el cuenta el problema descripto en el TP1. Para esto se utilizo una seccion 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 Calificacion y Usuario
- Para comisiones, usamos las tablas de Publicacion y Tipo de Publicacion.
- Para monto de facturas, usamos las tabla de Facturas

2.2. Colecciones

A continuacion 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. Migracion 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 Facutra

```
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 coleccion de publicaciones no tuvimos que hacer ningun algoritmo.

2.4. Consultas SQL

A continuacion 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 nu
  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.c
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. Implementacion 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. Inserccion 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. Nota: A la coleccion de publicaciones le agregamos un atributo extra ïdUsuarioVendedor"para poder usarlo como atributo clave a la hora de ahcer experimentos de sharding.

El pseudocodigo para la generacion de los 500000 registros:

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


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

A continuacion, 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

4.2. Resultados

poner que devuelve db.publicaciones.getShardDistribution() y sh.status()

4.3. Creacion de Shards

5. Conclusiones