Administrar bases de datos con comandos No SQL en el gestor de base de datos MongoDB.

Karen Viviana Diaz Guevara

Wilson Martínez Saldarriaga
(Instructor)

Centro De Gestión Y Desarrollo Sostenible Sur Colombiano
Pitalito- Huila
2023

Tabla de Contenidos

Introducción

En el siguiente trabajo vamos a estar realizando unos ejercicios sobre las bases de datos no relacionales de varios casos con MongoDB, también estaremos realizando La base de datos de nuestro subproyecto Simon (sistema de monitoreo de café) con base de datos relacional y no relacional, y los modelos relacionales, lógico.

Actividad

1) **Crear el modelo documental por referencia y el modelo físico en una base de datos no relacional en MongoDB del siguiente caso de estudio de información policial.**

La Policía quiere crear una base de datos sobre la seguridad en algunas entidades bancarias. Para ello tiene en cuenta:

- Que cada entidad bancaria se caracteriza por un código y por el domicilio de su Central.
- Que cada entidad bancaria tiene más de una sucursal que también se caracteriza por un código y por el domicilio, así como por el número de empleados de dicha sucursal.
- Que cada sucursal contrata, según el día, algunos vigilantes, que se caracterizan por un código y su edad. Un vigilante puede ser contratado por diferentes sucursales (incluso de diferentes entidades), en distintas fechas y es un dato de interés dicha fecha, así como si se ha contratado con arma o no.
- Por otra parte, se quiere controlar a las personas que han sido detenidas por atracar las sucursales de dichas entidades. Estas personas se definen por una clave (código) y su nombre completo.
- Alguna de estas personas está integrada en algunas bandas organizadas y por ello se desea saber a qué banda pertenecen, sin ser de interés si la banda ha participado en el delito o no Dichas bandas se definen por un número de banda y por el número de miembros.
- Así mismo, es interesante saber en qué fecha ha atracado cada persona una sucursal.
- Evidentemente, una persona puede atracar varias sucursales en diferentes fechas, así como que una sucursal puede ser atracada por varias personas.
- Igualmente, se quiere saber qué Juez ha estado encargado del caso, sabiendo que un individuo, por diferentes delitos, puede ser juzgado por diferentes jueces. Es de interés saber, en cada delito, si la persona detenida ha sido condenada o no y de haberlo sido, cuánto tiempo pasará en la cárcel. Un Juez se caracteriza por una clave interna del juzgado, su nombre y los años de servicio.
  NOTA: En ningún caso interesa saber si un vigilante ha participado en la detención de un atracador.

Colección de Vigilantes

```
> db.vigilants.insertOne({
      Age: 'number'
  })
< {
    acknowledged: true,
    insertedId: ObjectId("656d433be6fc57672db01f27")
  }
```

Colección de Contrato

```
> db.hires.insertOne({
      Hire_date: 'date',
      branch: 'ObjectId',
      vigilant: 'ObjectId'
  })
< {
    acknowledged: true,
    insertedId: ObjectId("656d4395e6fc57672db01f28")
  }
```

Colección de Sucursales

```
> db.branchs.insertOne({
      N_employes: 'number',
      home: 'string',
      bank: 'ObjectId'
  })
< {
    acknowledged: true,
    insertedId: ObjectId("656d4429e6fc57672db01f29")
  }
```

Colección de Bancos

```
> db.banks.insertOne({
      Head_office: 'string'
  })
< {
    acknowledged: true,
    insertedId: ObjectId("656d46f0e6fc57672db01f2a")
  }
```

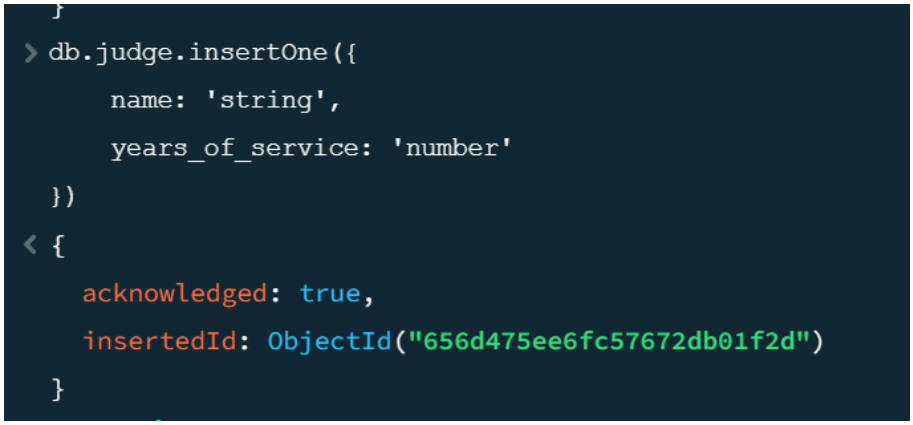
Colección de bandas

```
> db.bands.insertOne({
      n_members: 'number'
  })
< {
    acknowledged: true,
    insertedId: ObjectId("656d4729e6fc57672db01f2b")
  }
```

Colección de robos

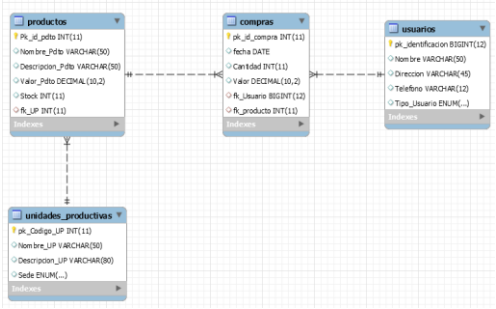
```
> db.robs.insertOne({
      robbery_date: 'date',
      branch: 'ObjectId',
      robber: 'ObjectId',
      judge: 'ObjectId'
  })
< {
    acknowledged: true,
    insertedId: ObjectId("656d4747e6fc57672db01f2c")
  }
```

Colección de juez



Nota: Para insertar datos en una colección se pueden utilizar de dos maneras cuando es un solo datos se realiza con insertOne y se vería de la siguiente manera:
db. nombreColección.insertOne({'dato1': valor1,'dato12': valor2});
Y en el caso de que sean muchos se realiza con el comando insertMany seguido de una coma por cada dato y se vería así:
db. NombreColección.insertMany(
[{valor1, valor2, valor3},
{valor1, valor2, valor3}]

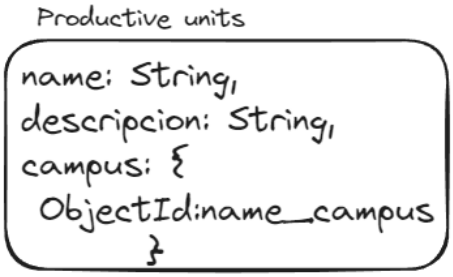
2) **Crear el modelo documental por referencia y el modelo físico en MongoDB**
   a) Crear el Modelo Físico del siguiente Modelo Relacional de Unidades Productivas de Sena Empresa



   b) Registrar las siguientes unidades productivas

Colección de Unidades Productivas:

codigo_up	Nombre_up	Descripción	Sede
1	Agrícola	Producción Productos del campo orgánicos	Yamboro
2	Agroindustria	Proceso de productos lácteos y cárnicos	Yamboro
3	Gastronomía	Venta de almuerzos especiales	Yamboro
4	Pecuaria	Producción de peces	Yamboro
5	Escuela Nacional de la Calidad del Café	Análisis del café	Yamboro
6	Ambiental – Recursos Naturales	Educación ambiental	Yamboro
7	Empresa de Servicios Públicos	Electricistas	Yamboro
8	Moda – Comercio y Servicios	Empresas	Centro



```
> db.campus.insertMany(
    [
        {
            name: "centro"
        },
        {
            name: "yamboro"
        }
    ]
)
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("656dda6db265acc6a85b46f3"),
    '1': ObjectId("656dda6db265acc6a85b46f4")
  }
}
```

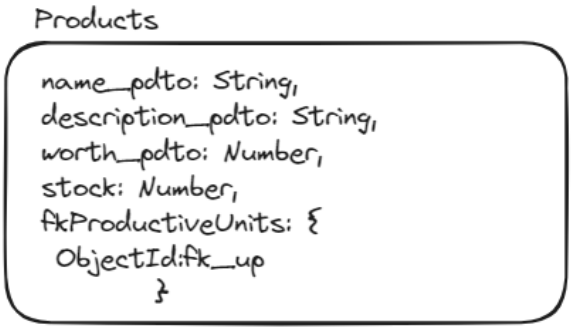
```
> db.productive_units.insertMany(
    [
        {
            name: "Agricola",
            descripcion: "Produccion productos del campo organicos",
            campus: ObjectId("656dda6db265acc6a85b46f4")
        },
        {
            name: "Agroindustrial",
            descripcion: "Proceso de productos lacteos y carnicos",
            campus: ObjectId("656dda6db265acc6a85b46f4")
        },
        {
            name: "Gastronomia",
            descripcion: "Venta de almuerzos especiales",
            campus:ObjectId("656dda6db265acc6a85b46f4")
        },
        {
            name: "Pecuaria",
            descripcion: "Produccion de peces",
```

```
            name: "Moda-comercio y servicios",
            descripcion: "Empresas",
            campus: ObjectId("656dda6db265acc6a85b46f3")
        }
    ]
)
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("656ddb57b265acc6a85b46f5"),
    '1': ObjectId("656ddb57b265acc6a85b46f6"),
    '2': ObjectId("656ddb57b265acc6a85b46f7"),
    '3': ObjectId("656ddb57b265acc6a85b46f8"),
    '4': ObjectId("656ddb57b265acc6a85b46f9"),
    '5': ObjectId("656ddb57b265acc6a85b46fa"),
    '6': ObjectId("656ddb57b265acc6a85b46fb"),
    '7': ObjectId("656ddb57b265acc6a85b46fc")
  }
}
```

c) Registrar los siguientes productos.

Colección de productos:

Pk_id_pdto	Nombre_Pdto	Descripcion_Pdto	Valor_Pdto	Stock	fk_UP
1	yogurt	yogurt con frutas	500	120	2
2	Chorizo	Chorizo de pollo	1000	500	2
3	Avena	Avena en vaso	2000	600	2
4	Cilantro	Hortalizas y verduras x Manojo	500	300	1
5	Cebolla	Cebolla Larga X Libra	800	100	1
6	Tomate	Tomate Cherry X Libra	1500	200	1
7	Almuerzos	Almuerzos especiales	5000	500	3
8	Cachama	Cachama Roja X Libra	4500	300	4
9	Trucha	Truca arreglada X Libra	6000	140	4
10	Café	Café especial X Libra	15000	700	5



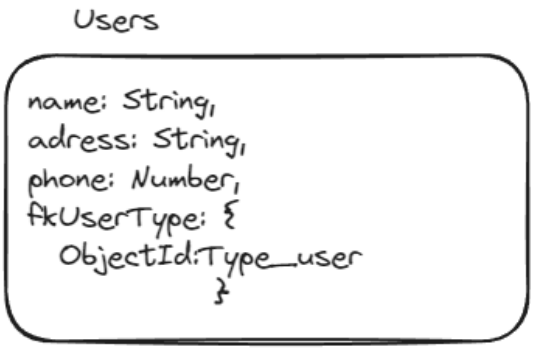

```
> db.products.insertMany(
    [
        {
            name_pdto: "yogurth",
            description_pdto: "yogurth con frutas",
            worth_pdto: 500,
            stock: 120,
            fkProductiveUnits: ObjectId("656ddb57b265acc6a85b46f6")
        },
        {
            name_pdto: "chorizo",
            description_pdto: "chorizo de pollo",
            worth_pdto: 1000,
            stock: 500,
            fkProductiveUnits: ObjectId("656ddb57b265acc6a85b46f6")
        },
        {
            name_pdto: "avena",
            description_pdto: "avena en vaso",
            worth_pdto: 2000,
```

```
            stock: 700,
            fkProductiveUnits: ObjectId("656ddb57b265acc6a85b46f9")
        },
    ]
)
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("656de060b265acc6a85b46fd"),
    '1': ObjectId("656de060b265acc6a85b46fe"),
    '2': ObjectId("656de060b265acc6a85b46ff"),
    '3': ObjectId("656de060b265acc6a85b4700"),
    '4': ObjectId("656de060b265acc6a85b4701"),
    '5': ObjectId("656de060b265acc6a85b4702"),
    '6': ObjectId("656de060b265acc6a85b4703"),
    '7': ObjectId("656de060b265acc6a85b4704"),
    '8': ObjectId("656de060b265acc6a85b4705"),
    '9': ObjectId("656de060b265acc6a85b4706")
  }
}
```

e) Registrar los siguientes Usuarios

Colección de Usuarios:

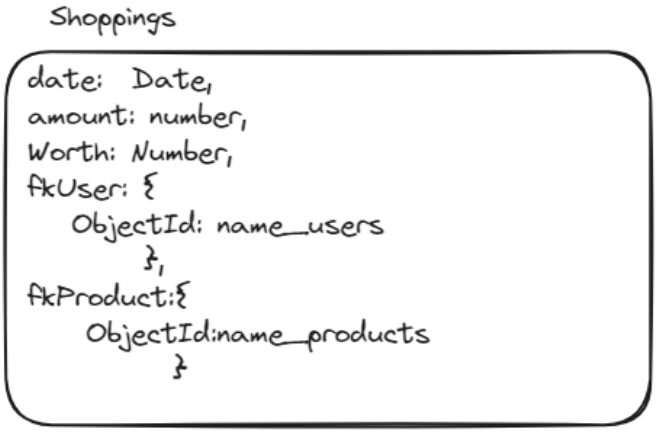
pk_identificacion	nombre	dirección	teléfono	Tipo_Usuario
100426973	ELIAN CANDIL			Aprendiz
119355841	LINA TATIANA SAMBONI			Aprendiz
1002337863	JERSON SMITH			Aprendiz
1004248797	LEIDY DAYANA INCHIMA			Aprendiz
1004269672	NATALIA ROJAS ROJAS			Aprendiz
1004402263	MANUEL CAMILO OME			Aprendiz
1004418839	OSWALDO SAMBONI BOLAÑOS			Aprendiz
1004492751	DANA ARTUNDUAGA			Aprendiz
1004492861	LAURA VANESSA			Aprendiz
1006410046	FERNANDO SARREAS			Aprendiz
1007269672	ARMANDO CUELLAR			Aprendiz
1007308252	JHONARY LOSADA			Aprendiz
1007308344	JERSON STERLING			Aprendiz
1007308354	DIEGO ALEGANDRO LOPEZ			Aprendiz
1007388140	KAREN DANIELA ROJAS			Aprendiz
96361787	WILSON MARTINEZ SALDARRIAGA	CRA 19-CLL2	3167512637	Instructor
125345343	JESUS DAVID CALDERON	CLL 3-CRA 12	3122874654	Instructor




```
> db.users.insertMany(
    [
        {
            name: "Elian Candil",
            direccion: "Cra 5 20-5",
            telefono: "31234553424",
            fkUserType: ObjectId("656607529e353c6ef6bf2667")
        },
        {
            name: "Lina Tatiana Samboni",
            direccion: "Cra 3 8-60",
            telefono: "31789002",
            fkUserType: ObjectId("656607529e353c6ef6bf2667")
        },
        {
            name: "Jerson Smith",
            direccion: "Cra 1 4-4",
            telefono: "3678992203",
            fkUserType: ObjectId("656607529e353c6ef6bf2667")
        },
```

```
>_MONGOSH
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("656de34cb265acc6a85b4709"),
    '1': ObjectId("656de34cb265acc6a85b470a"),
    '2': ObjectId("656de34cb265acc6a85b470b"),
    '3': ObjectId("656de34cb265acc6a85b470c"),
    '4': ObjectId("656de34cb265acc6a85b470d"),
    '5': ObjectId("656de34cb265acc6a85b470e"),
    '6': ObjectId("656de34cb265acc6a85b470f"),
    '7': ObjectId("656de34cb265acc6a85b4710"),
    '8': ObjectId("656de34cb265acc6a85b4711"),
    '9': ObjectId("656de34cb265acc6a85b4712"),
    '10': ObjectId("656de34cb265acc6a85b4713"),
    '11': ObjectId("656de34cb265acc6a85b4714"),
    '12': ObjectId("656de34cb265acc6a85b4715"),
    '13': ObjectId("656de34cb265acc6a85b4716"),
    '14': ObjectId("656de34cb265acc6a85b4717"),
    '15': ObjectId("656de34cb265acc6a85b4718"),
    '16': ObjectId("656de34cb265acc6a85b4719")
  }
```

Compras




```
> db.shoppings.insertMany(
    [
        {
            date: new Date('2023, 11, 22'),
            amount: 5,
            Worth: 2500,
            fkUser: ObjectId("656de34cb265acc6a85b4719"),
            fkProduct: ObjectId("656de060b265acc6a85b4706")
        },
        {
            date: new Date('2023, 05, 12'),
            amount: 10,
            worth: 10000,
            fkUser: ObjectId("656de34cb265acc6a85b4718"),
            fkProduct: ObjectId("656de060b265acc6a85b4705")
        },
        {
            date: new Date('2023, 08, 01'),
            amount: 5,
            worth: 10000,
```

```
            fkProduct: ObjectId("656de060b265acc6a85b4701")
        }
    ]
)
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("656deec9b265acc6a85b471a"),
    '1': ObjectId("656deec9b265acc6a85b471b"),
    '2': ObjectId("656deec9b265acc6a85b471c"),
    '3': ObjectId("656deec9b265acc6a85b471d"),
    '4': ObjectId("656deec9b265acc6a85b471e"),
    '5': ObjectId("656deec9b265acc6a85b471f"),
    '6': ObjectId("656deec9b265acc6a85b4720"),
    '7': ObjectId("656deec9b265acc6a85b4721"),
    '8': ObjectId("656deec9b265acc6a85b4722"),
    '9': ObjectId("656deec9b265acc6a85b4723")
  }
}
```

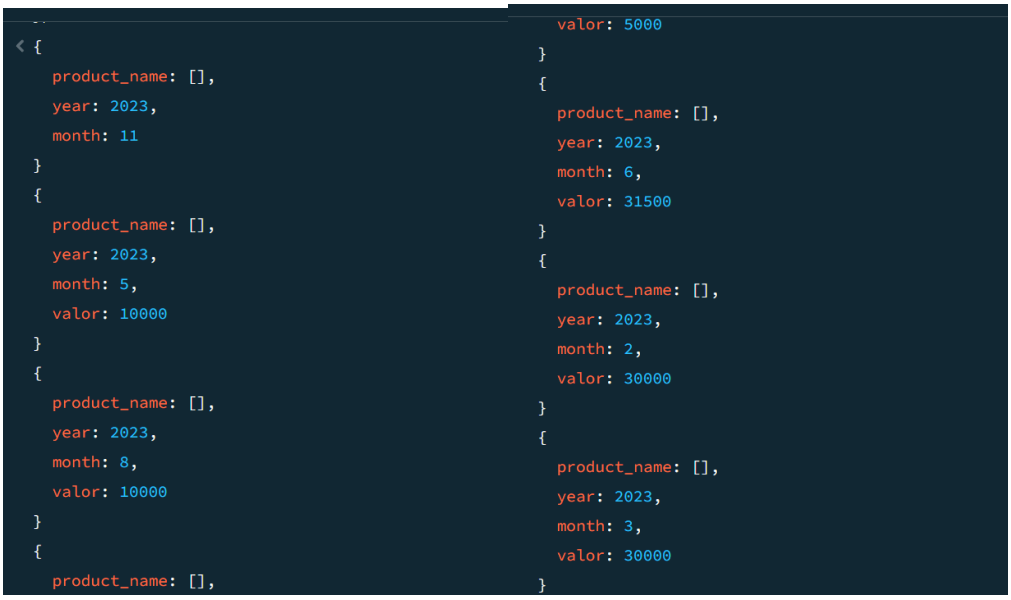
Nota: Para insertar datos en una colección se pueden utilizar de dos maneras cuando es un solo datos se realiza con insertOne y se vería de la siguiente manera:
db. nombreColección.insertOne({'dato1': valor1,'dato12': valor2});
Y en el caso de que sean muchos se realiza con el comando insertMany seguido de una coma por cada dato y se vería así:
db. NombreColección.insertMany(
[{valor1, valor2, valor3},
{valor1, valor2, valor3}]

a) Listar los nombres de los clientes que su nombre inicia con las letras L, D, G, A, F, J;

```
> db.users.find({ name: { $regex: /^[LDGAFJ]/ } }, { _id: 0, name: 1 });
< {
    name: 'Lina Tatiana Samboni'
  }
  {
    name: 'Jerson Smith'
  }
  {
    name: 'Leidy Dayana Inchima'
  }
  {
    name: 'Dana Artunduaga'
  }
  {
    name: 'Laura Vanesa'
  }
  {
    name: 'Fernando Sarreas'
  }
```

b) Listar datos estadísticos de compras, mostrar nombre del producto, año, mes, valor.

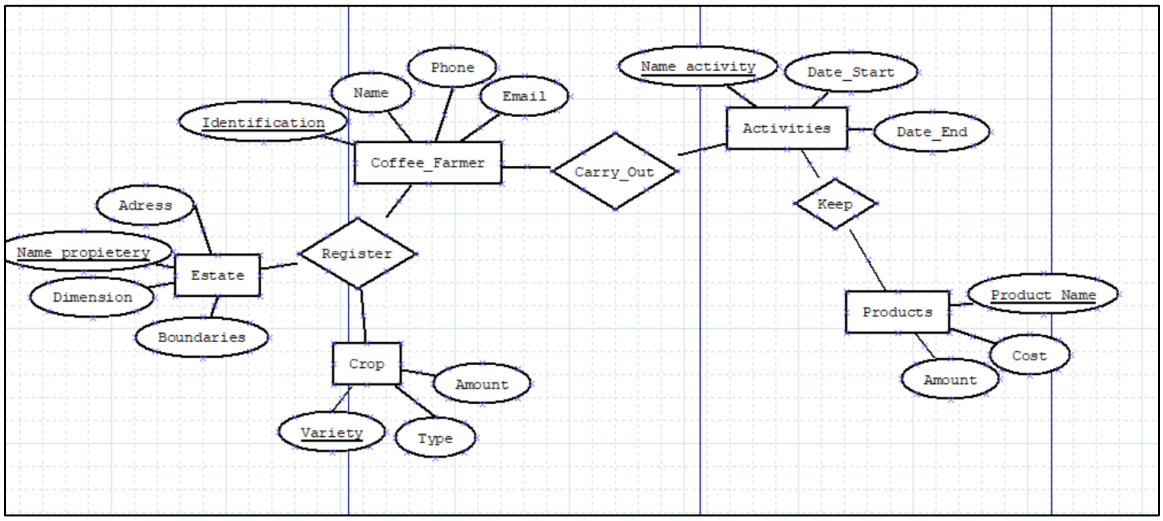
```
> db.shoppings.aggregate([
    {
      $lookup: {
        from: "products",
        localField: "fk_product",
        foreignField: "_id",
        as: "product"
      }
    },
    {
      $project: {
        _id: 0,
        product_name: "$product.name",
        year: { $year: "$date" },
        month: { $month: "$date" },
        valor: "$worth"
      }
    }
  ])
```



3) Crear la base de datos (Modelo entidad relación, Modelo Lógico, Modelo físico) en MySql del proyecto formativo.

Requerimiento Funcionales	**Descripción**	**Fuente**
Inicio de sesión	Al iniciar la aplicación se desplegará una ventana la cual mostrará el inicio de sesión con dos inputs de correo y contraseña	
Registrar	Este apartado se encontrará al momento de presionar un botón en inicio sesión donde lo direccionará a este apartado donde el usuario tendrá la posibilidad de registrarse para poder ingresar al aplicativo	
Dato Usuario	Este apartado se encontrará en un lado de la pantalla principal donde el usuario podrá revisar que los datos de el mismo sean correctos contará con información: • Identificación • Nombre • Correo • Dirección • Teléfono	
Registro de finca	Este apartado se encontrará disponible para el registro de lo que se necesite saber sobre la finca tendrá:	

	• Nombre finca • Dirección finca • Dimensiones de finca • Linderos de la finca	
Registro de actividades	Este apartado, aunque es automáticas las actividades el usuario tiene la posibilidad de modificar al igual que aumentar y eliminar: • Nombre actividad • Fecha de inicio • Fecha fin	
Información de cultivos	En este apartado se encontrará información sobre ayudas en el cuidado del cultivo, como información general de abonos, venenos, y plagas: • Variedad • Tipo • Cantidad	



Nota: ***Pasos para crear el modelo entidad relación:***

1) Encontrar entidades en cada uno de los requerimientos del usuario.
2) Identificar atributos de las entidades seleccionadas.
3) Buscar identificadores (dato único de la entidad).
4) Especificar las relaciones y cardinalidades entre las entidades.
5) Identificar entidades débiles.
6) Especializar y generalizar entidades donde sea posible

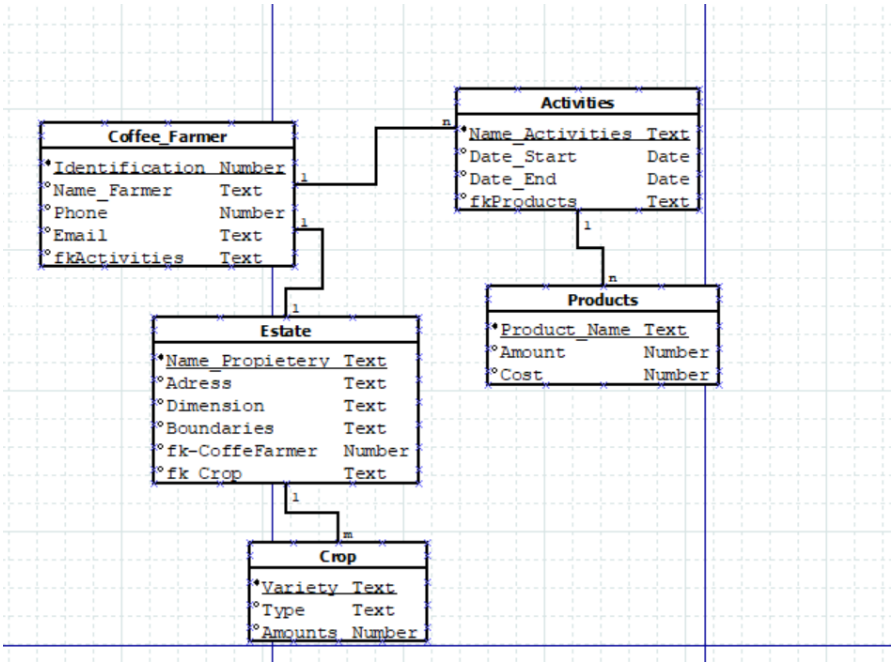


Tabla del Caficultor

```
MariaDB [simon]> describe coffeeFarmer;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| identification | int(11)     | NO   | PRI | NULL    |       |
| nameFarmer     | varchar(50) | NO   |     | NULL    |       |
| phone          | int(11)     | YES  |     | NULL    |       |
| email          | varchar(50) | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
4 rows in set (0.060 sec)
```

Tabla de Finca

```
MariaDB [simon]> describe estate
    -> ;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| namePropiertery | varchar(50) | NO   | PRI | NULL    |       |
| adress          | varchar(50) | NO   |     | NULL    |       |
| dimension       | varchar(50) | NO   |     | NULL    |       |
| boundaries      | varchar(50) | NO   |     | NULL    |       |
| fkCoffeefarmer  | int(11)     | NO   |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
5 rows in set (0.837 sec)
```

Tabla de Cultivo

```
MariaDB [simon]> describe crop;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| variety | varchar(50) | NO   |     | NULL    |       |
| type    | varchar(50) | NO   |     | NULL    |       |
| amounts | int(11)     | NO   |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
3 rows in set (0.017 sec)
```

Tabla de Actividades

```
MariaDB [simon]> describe activities;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| nameActivities | varchar(50) | NO   |     | NULL    |       |
| dateStart      | date        | NO   |     | NULL    |       |
| dateEnd        | date        | NO   |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
3 rows in set (0.048 sec)
```

Tabla de productos

```
MariaDB [simon]> describe products;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| productName  | varchar(50) | NO   |     | NULL    |       |
| amount       | int(11)     | NO   |     | NULL    |       |
| cost         | int(11)     | NO   |     | NULL    |       |
| fkActivities | varchar(50) | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.011 sec)
```

Relaciones entre tablas

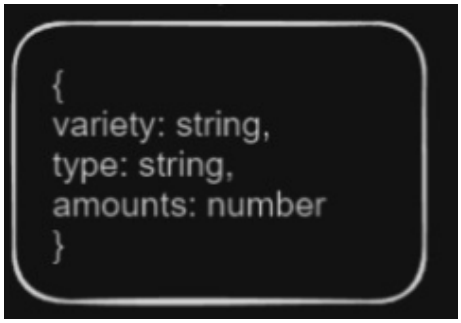
```
MariaDB [simon]> alter table estate add constraint register foreign key(fkCoffeefarmer)
    -> references coffeefarmer (identification)
    -> ;
Query OK, 0 rows affected (0.453 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [simon]> alter table estate add constraint tener foreign key (fkCrop) references crop (variety);
Query OK, 0 rows affected (0.561 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [simon]>  alter table coffeefarmer add constraint CarryOut foreign key (fkActivities) references activities (nameActivities);
Query OK, 0 rows affected (0.331 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [simon]> alter table activities add constraint keep foreign key(fkProducts) references products (productName);
Query OK, 0 rows affected (0.365 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

4) Crear la base de datos (Modelo documental por referencia, Modelo físico) en MongoDB del proyecto formativo.

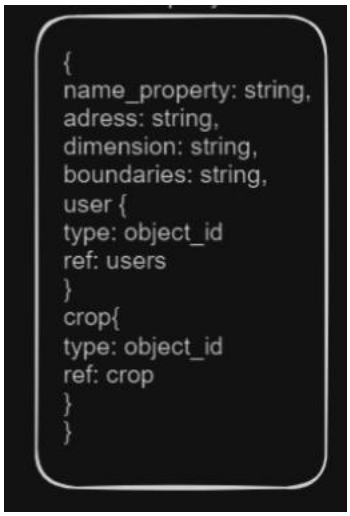
Requerimiento Funcionales	Descripción	Fuente
Inicio de sesión	Al iniciar la aplicación se desplegará una ventana la cual mostrará el inicio de sesión con dos inputs de correo y contraseña	
Registrar	Este apartado se encontrará al momento de presionar un botón en inicio sesión donde lo direccionará a este apartado donde el usuario tendrá la posibilidad de registrarse para poder ingresar al aplicativo	
Dato Usuario	Este apartado se encontrará en un lado de la pantalla principal donde el usuario podrá revisar que los datos de el mismo sean correctos contará con información: • Identificación • Nombre • Correo • Dirección • Teléfono	
Registro de finca	Este apartado se encontrará disponible para el registro de lo que se necesite saber sobre la finca tendrá: • Nombre finca • Dirección finca • Dimensiones de finca • Linderos de la finca	
Registro de actividades	Este apartado, aunque es automáticas las actividades el usuario tiene la posibilidad de modificar al igual que aumentar y eliminar: • Nombre actividad • Fecha de inicio • Fecha fin	
Registro de costos	Este apartado se utilizará para ver los costos de materiales que se utilizaron en un	

	determinado tiempo: • Costo de Producto • Nombre producto • Descripción del producto • Fecha de gastos	
Información de cultivos	En este apartado se encontrará información sobre ayudas en el cuidado del cultivo, como información general de abonos, venenos, y plagas: • Variedad • Tipo • Cantidad	

Crop

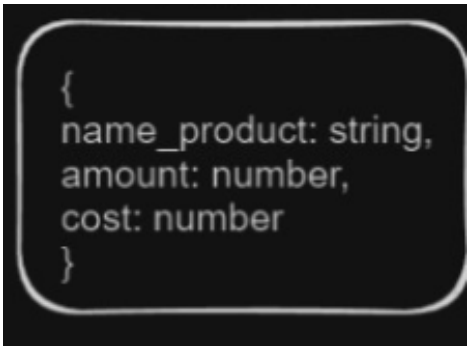


Estate



Activities

Products



Coffee Farmer

Colecciones

```
> use Simon
< switched to db Simon
> db.createCollection.crop
> db.createCollection.estate
> db.createCollection.activities
> db.createCollection.products
```

Colección de Caficultor

```
> db.coffeFarmer.insertOne({
    identification: 'Number',
    nameFarmer: 'String',
    phone: 'Number',
    Email:'String'
})
< {
  acknowledged: true,
  insertedId: ObjectId("65762eb8bd3a0652f7f25bd2")
}
```

Colección de Cultivo

```
> db.crop.insertOne({
    variety:'Sting',
    type:'String',
    amounts: 'Number'
})
< {
  acknowledged: true,
  insertedId: ObjectId("65762ed2bd3a0652f7f25bd3")
}
```

Colección de finca

```
> db.estate.insertOne({
    namePropiertery:'String',
    adress:'String',
    dimesion:'String',
    boundaries:'String'
  })
< {
    acknowledged: true,
    insertedId: ObjectId("65762f91bd3a0652f7f25bd4")
  }
```

Colección de Actividades

```
> db.activities.insertOne({
     nameActivies:'String',
     dateStar:'Date',
     dateEnd:'Date'
  })
< {
    acknowledged: true,
    insertedId: ObjectId("65762fa4bd3a0652f7f25bd5")
  }
```

Colección de productos

```
> db.products.insertOne({
      productName : 'String',
      amount:'Number',
      cost:'Number'
  })
< {
    acknowledged: true,
    insertedId: ObjectId("65762fc6bd3a0652f7f25bd6")
  }
```

Nota: Para insertar datos en una colección se pueden utilizar de dos maneras cuando es un solo datos se realiza con insertOne y se vería de la siguiente manera:
db. nombreColección.insertOne({'dato1': valor1,'dato12': valor2});
Y en el caso de que sean muchos se realiza con el comando insertMany seguido de una coma por cada dato y se vería así:
db. NombreColección.insertMany(
[{valor1, valor2, valor3},
{valor1, valor2, valor3}]

Conclusión

En el anterior trabajo se pudo evidenciar unos ejercicios sobre las bases de datos no relacionales de varios casos con MongoDB, también estaremos realizando La base de datos de nuestro subproyecto Simon (sistema de monitoreo de café) con base de datos relacional y no relacional, y los modelos relacionales, lógico.