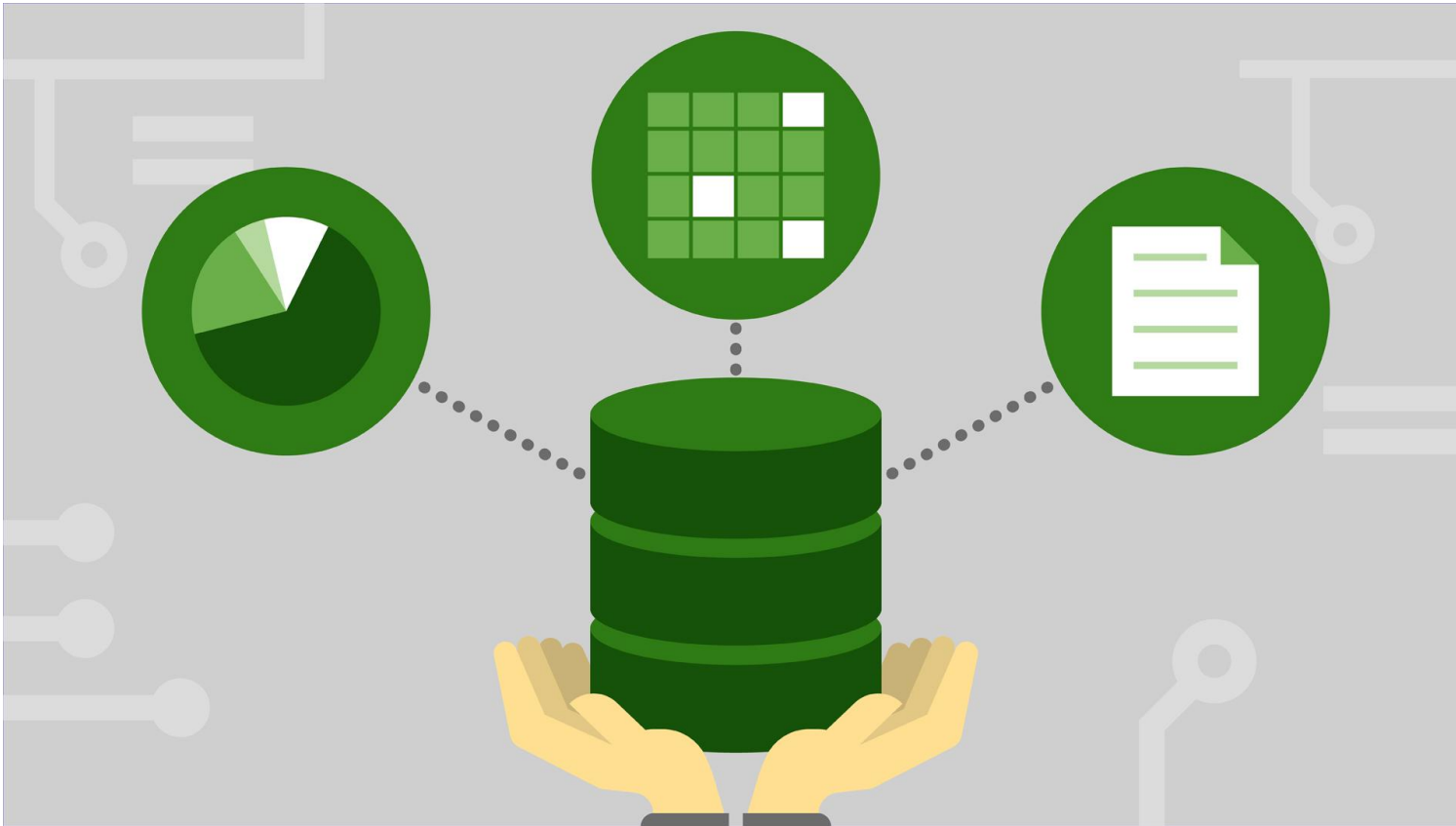


mongoDB

NOSQL: BASE DE DATOS ORIENTADA A DOCUMENTOS.

NoSQL



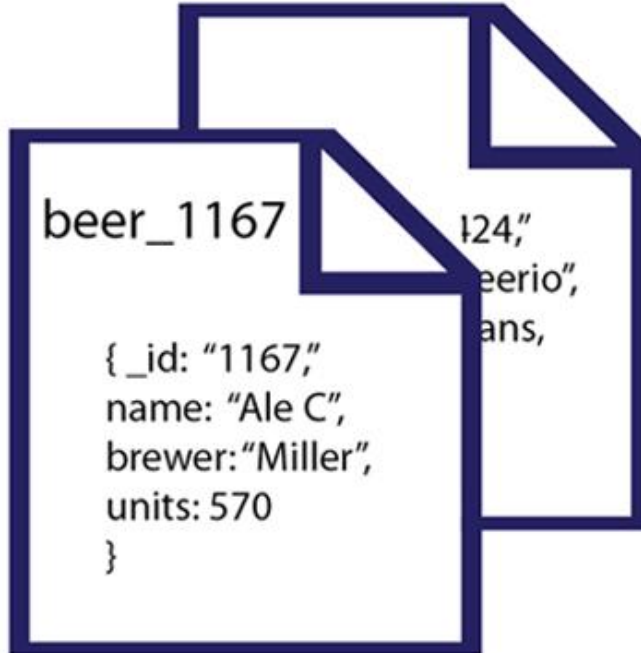
- 🗄️ ¿Porque surge?
- 🗄️ ¿Ventajas?
- 🗄️ ¿Desventajas?
- 🗄️ ¿Cuándo usar?

MongoDB

Colección

1167	Ale C	Miller	570
3424	Beerio	Ians	340
5612	Amstel	Amtel	121
2409	Colt's	BeerCo	98

Beer Documents



- Escrito en c++.
- Licenciado como GNU AGPL 3.0
- Funciona en Windows, Linux, os x y Solaris.
- Su origen se remonta a mediados del 2000, como intento de descifrar registros NoSQL.
- En 2009 sale como código abierto y actualmente su ultima versión es la 4.0

MongoDB

JSON

```
{  
  "array": [  
    1,  
    2,  
    3  
  ],  
  "boolean": true,  
  "null": null,  
  "number": 123,  
  "object": {  
    "a": "b",  
    "c": "d",  
    "e": "f"  
  },  
  "string": "Hello World"  
}
```

BSON

{ 01010100
 11101011
 10101110
 01010101 }





Principales características de MongoDB:

- Los datos se duplican para proteger el sistema en caso de falla del hardware.
- Admisión de MapReduce y herramientas de agregación
- Fácil administración en caso de fallas
- Está sin esquema escrito en C ++
- La combinación de MongoDB y JavaScript funciona bien ya que la base de datos usa el lenguaje en lugar de los procedimientos.




La consola de MongoDB se basa sobre el lenguaje JS.
Pero admite otros lenguajes.

MongoDB

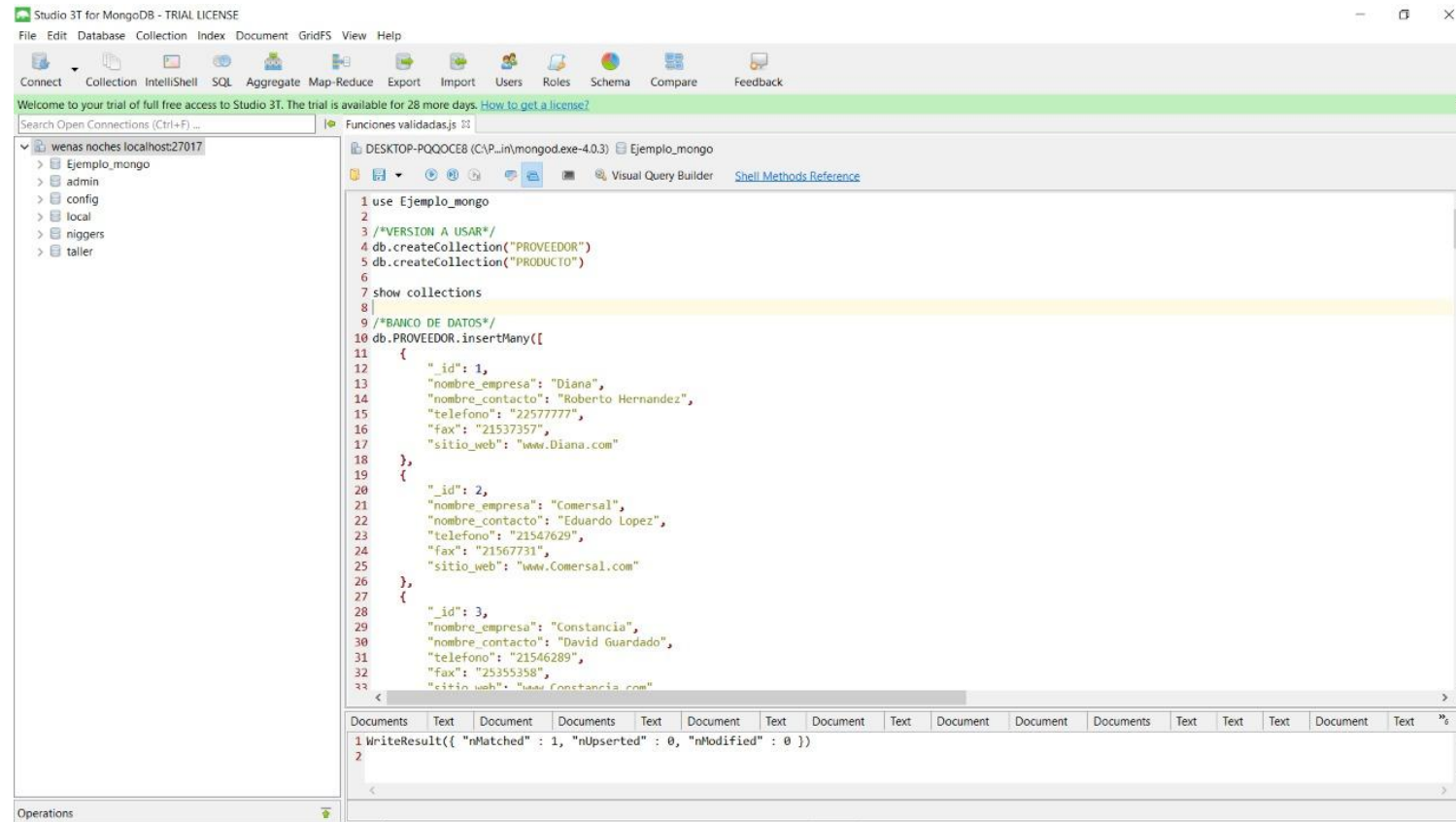
Ventajas:

-  Desarrollo rápido e iterativo.
-  Modelo de datos flexibles.
-  TCO reducido(Costo total de propiedad).
-  Conjunto de características integrado.

Desventajas:

-  La salida de datos es variable por lo que si necesitas una salida de datos con mas estructura no es conveniente usar mongoDB.
-  MongoDB no permite tener una estructura bien definida
-  Aun es una tecnología joven.

Studio 3T-IDLE alternativo a consola MongoDB



MongoDB

Empresas que usan MongoDB.

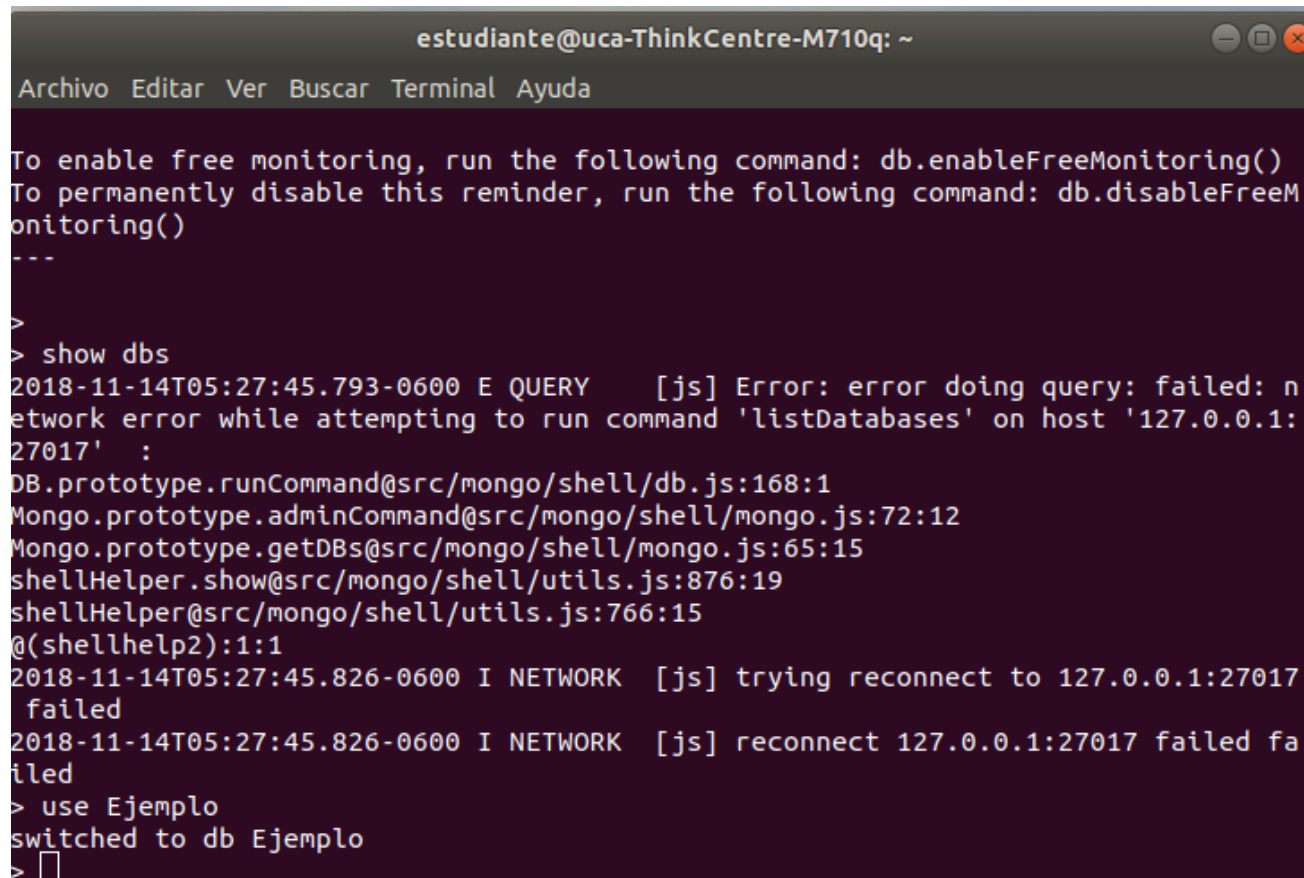
Telefónica

SEGA

 **Expedia**®

Forbes

Primera Parte: Consola (Linux)



```
estudiante@uca-ThinkCentre-M710q: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
> show dbs
2018-11-14T05:27:45.793-0600 E QUERY    [js] Error: error doing query: failed: network error while attempting to run command 'listDatabases' on host '127.0.0.1:27017' :
DB.prototype.runCommand@src/mongo/shell/db.js:168:1
Mongo.prototype.adminCommand@src/mongo/shell/mongo.js:72:12
Mongo.prototype.getDBs@src/mongo/shell/mongo.js:65:15
shellHelper.show@src/mongo/shell/utils.js:876:19
shellHelper@src/mongo/shell/utils.js:766:15
@(shellhelp2):1:1
2018-11-14T05:27:45.826-0600 I NETWORK  [js] trying reconnect to 127.0.0.1:27017 failed
2018-11-14T05:27:45.826-0600 I NETWORK  [js] reconnect 127.0.0.1:27017 failed failed
> use Ejemplo
switched to db Ejemplo
> 
```


Creación de colecciones en MongoDB

```
CREATE TABLE nombreTabla(  
    id_nombreTabla INT CONSTRAINT PK_id_nombreTabla PRIMARY KEY,  
    name VARCHAR2(50)  
);
```

```
db.createCollection('nombreTabla')
```

Insert SQL-MongoDB

```
INSERT INTO nombreTabla VALUES (1, 'Mongo') ;
```

```
db.nombreTabla.insert({"name" : "Mongo"})
```

Usuarios

Por defecto, es posible acceder al servidor de MongoDB sin necesidad de autenticarse y esto genera problemas de seguridad.



Gestión de usuarios

Para agregar un usuario, MongoDB proporciona el metodo [db.createUser\(\)](#). Cuando se agregan usuarios, podemos asignarles roles o privilegios.

Roles

Nombre rol	Privilegio
Read	changeStream, collStats ,dbHash ,dbStats ,find ,killCursors ,listIndexes ,listCollections
readWrite	convertToCapped, createCollection ,dbHash ,dbStats ,dropCollection,createIndex,dropIndex,find ,insert ,killCursors ,listIndexes,listCollections, remove, Update
dbAdmin	collStats,dbHash,dbStats,find,killCursors,listIndexes,list Collections
dbOwner	Privilegios de read, readWrite, dbAdmin
userAdmin	changeCustomData,changePassword,createRole,createUser,dropRole,dropUser,grantRole,revokeRole,setAuthenticationRestriction,viewRole,viewUser

Creación de Usuarios

```
db.createUser(  
  {  
    user: "<name>",  
    pwd: "<cleartext password>",  
    roles: [  
      { role: "<role>", db: "<database>" } | "<role>",  
      ...  
    ]  
  }  
)
```

Comparación SQL-MongoDB

```
/*Crear usuario*/  
CREATE USER Admin  
IDENTIFIED BY 1234;
```

```
/*Asignarle privilegios al usuario*/  
GRANT CREATE SESSION TO Admin;  
GRANT CONNECT, RESOURCE, DBA Admin;
```

Habilitar control de acceso

MongoDB no habilita el control de acceso por defecto. Puede habilitar la autorización utilizando la configuración `--auth` o `security.authorization`. La habilitación de la autenticación interna también habilita la autorización del cliente.

Habilitar autenticación

Ahora que ya disponemos con dos usuarios con los privilegios para admi y un visitor

Imponemos la autenticación y requiere que los usuarios se identifiquen. Los usuarios solo pueden realizar acciones según lo determinado por sus roles.

Usuario Administrador

Con el control de acceso habilitado, hay que tener un usuario con el rol `userAdmin` o `userAdminAnyDatabase` en la base de datos de administración.

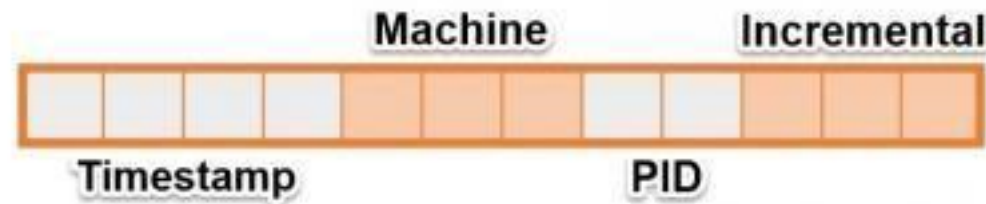


Comparación con SQL-MongoDB

```
INSERT INTO CATEGORIA_PRODUCTO VALUES (1, 'Carne y Embutidos');  
  
SELECT * FROM CATEGORIA_PRODUCTO;
```

¿Qué es un ObjectId?

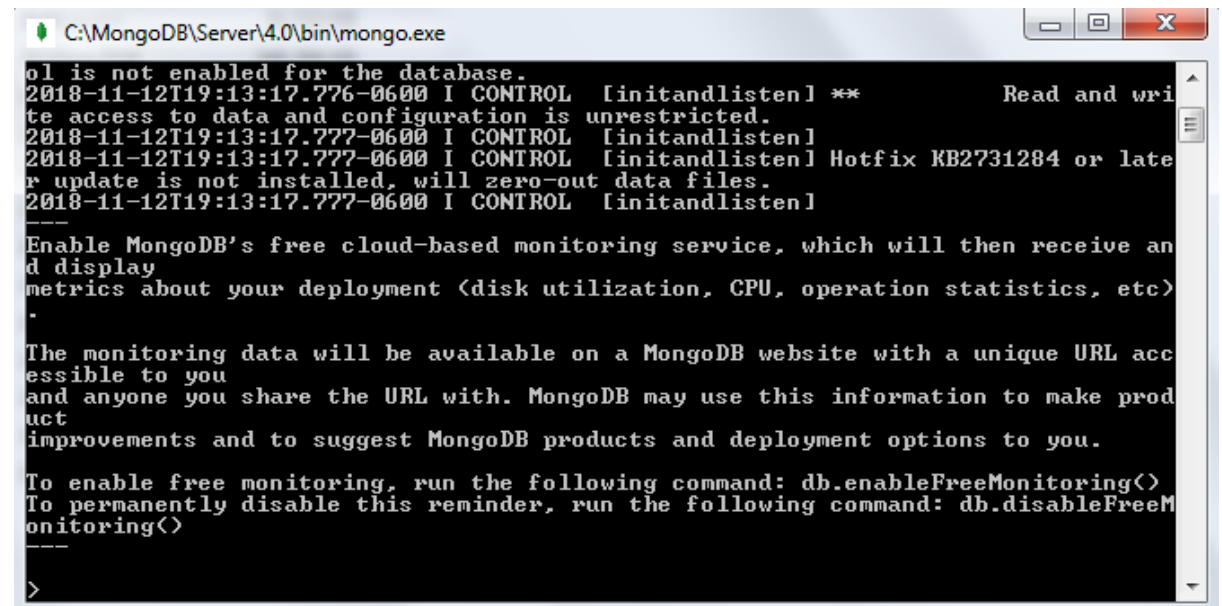
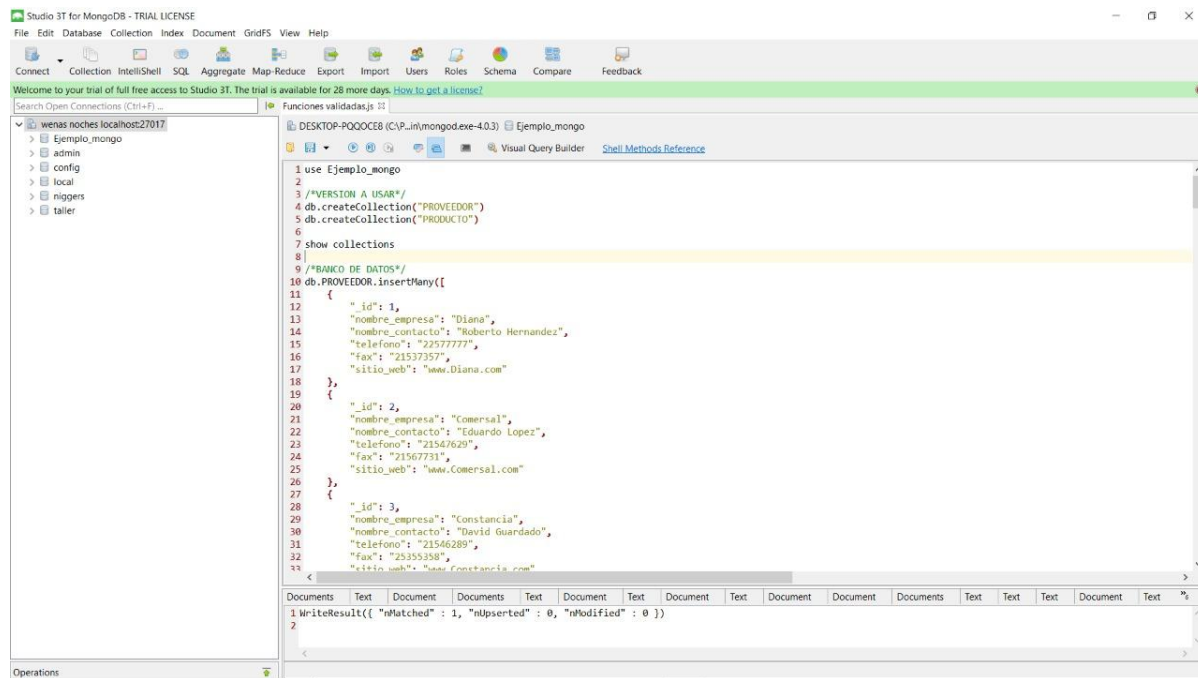
El campo está compuesto por 12 bytes



Los cuatro primeros bytes son un timestamp con la fecha y hora del momento en que se creó el proceso; los tres siguientes bytes representan el identificador único de la máquina; los dos siguientes el identificador del proceso; y para finalizar los últimos tres bytes, son un campo incremental.


Los tres últimos bytes, nos garantizan que cada segundo podemos insertar $2^{24} = 16,777,216$ documentos con un identificador distinto.

Segunda Parte: IDLE y Consola (Windows)



Operaciones en MongoDB

`db.collection.insertOne()`
`db.collection.insertMany()`




INSERT

`db.collection.find()`




SELECT

`db.collection.updateOne()`
`db.collection.updateMany()`
`db.collection.replaceOne()`



UPDATE

`db.collection.deleteOne()`
`db.collection.deleteMany()`



DELETE

```
db.users.insertOne(  ← collection
{
  name: "sue",      ← field: value
  age: 26,          ← field: value
  status: "pending" ← field: value } document
}
```

)

```
db.users.find(
  { age: { $gt: 18 } }, ← collection
  { name: 1, address: 1 } ← query criteria
).limit(5)              ← projection
                        ← cursor modifier
```

```
db.users.updateMany( ← collection
  { age: { $lt: 18 } }, ← update filter
  { $set: { status: "reject" } } ← update action
)
```

```
db.users.deleteMany( ← collection
  { status: "reject" } ← delete filter
)
```

Alternativa de relaciones en MongoDB

Relación incrustada

```
db.PRODUCTO2.insert(  
  {  
    "_id": 1,  
    "nombre": "Quesitos",  
    "fecha_llegada": "10/02/2018",  
    "fecha_vencimiento": "10/05/2018",  
    "precio_compra": 20.00,  
    "precio_venta": 0.50,  
    "stock": 43,  
    "descripcion": "Hecho con harina",  
    "categoria": "Aperitivos",  
    "proveedor": [  
      {  
        "nombre_empresa": "Diana",  
        "nombre_contacto": "Roberto Hernandez",  
        "telefono": "22577777",  
        "fax": "21537357",  
        "sitio_web": "www.Diana.com"  
      }  
    ]  
  }  
)
```

Relación referenciada

```
db.PROVEEDOR.insert(  
  {  
    "_id": 1,  
    "nombre_empresa": "Diana",  
    "nombre_contacto": "Roberto Hernandez",  
    "telefono": "22577777",  
    "fax": "21537357",  
    "sitio_web": "www.Diana.com"  
  }  
)  
  
db.PRODUCTO2.insert(  
  {  
    "_id": 1,  
    "nombre": "Quesitos",  
    "fecha_llegada": "10/02/2018",  
    "fecha_vencimiento": "10/05/2018",  
    "precio_compra": 20.00,  
    "precio_venta": 0.50,  
    "stock": 43,  
    "descripcion": "Hecho con harina",  
    "categoria": "Aperitivos",  
    "proveedor": 1  
  }  
)
```


Vistas

Así como SQL, en mongoDB existe algo llamado view, que posee la misma función, que es guardar una colección o un conjunto de colecciones, que en nuestro caso nos ayudara a unir la relación de nuestro ejemplo.

```
db.createView()
```

```
db.createView(<view>, <source>, <pipeline>, <options>)
```

```
db.nombreVista.find()
```

Cursores en SQL

```
DECLARE
    CURSOR cursorEjemplo is
        SELECT * FROM PRODUCTO;
    variables cursorEjemplo%ROWTYPE;
BEGIN
    OPEN cursorEjemplo;
    LOOP
        FETCH cursorEjemplo INTO variables;
        EXIT WHEN cursorEjemplo%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE (variables.nombre);
    END LOOP;
    CLOSE cursorEjemplo;
END;
```

Cursores MongoDB

Los cursores son usados para iterar o recorrer secuencialmente los resultados devueltos por una determinada consulta. Por defecto, en el shell de MongoDB, se muestra el resultado de la consulta por lo que se puede añadir una sentencia null para evitar que salga por pantalla toda esa información:

```
cur = db.PRODUCTO.find() ; null;
```

Mostrar

```
DECLARE
    CURSOR cursorEjemplo is
        SELECT * FROM PRODUCTO;
    variables cursorEjemplo%ROWTYPE;
BEGIN
    OPEN cursorEjemplo;
    LOOP
        FETCH cursorEjemplo INTO variables;
        EXIT WHEN cursorEjemplo%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE (variables.nombre);
    END LOOP;
    CLOSE cursorEjemplo;
END;
```

```
while (cur.hasNext()) printjson(cur.next())
```

Limitar cursores en MongoDB

Sólo se podrán consultar los primeros N documentos

```
cur.limit(1) ;
```

Ordenar cursores en MongoDB

Sort:

“1”: ascendente

“-1”: descendente

```
cur.sort({ name: -1 });
```

Ignorar o saltar documentos en cursores en MongoDB

Skip: permite ignorar los primeros N documentos encontrados y comenzar desde el documento N+1 como si fuese el primero.

```
cur.sort({ name: -1 }).limit(2).skip(1);
```


Funciones en MongoDB

```
//Bloque de funciones en MongoDB

db.system.js.save({
  _id: "Nombre_funcion",
  value: function(Parametros que recibira){
    .
    ..
    ...
    "Bloque se codigo javascript"
  })

//Bloque para compilar una funcion

db.eval("<Nombre de la funcion>(Parametros)")|
```

Operadores de comparación

Este tipo de operadores de expresión se utilizan para comparar valores y devolver un resultado. Los operadores disponibles son los siguientes:

- `$cmp`: compara dos valores y devuelve un número entero como resultado. Devuelve -1 si el primer valor es menor que el segundo, 0 si son iguales y 1 si el primer valor es mayor que el segundo.
- `$eq`: compara dos valores y devuelve true si son equivalentes.
- `$gt`: compara dos valores y devuelve true si el primero es más grande que el segundo.
- `$gte`: compara dos valores y devuelve true si el primero es igual o más grande que el segundo.
- `$lt`: compara dos valores y devuelve true si el primero es menor que el segundo.
- `$lte`: compara dos valores y devuelve true si el primero es igual o menor que el segundo.
- `$ne`: compara dos valores y devuelve true si los valores no son equivalentes.

Los operadores `$gt`, `$gte`, `$lt`, `$lte`, `$ne` se utilizan de la misma manera que en las consultas find normales.

Aggregate Framework

Apareció con la versión 2.2 de **MongoDB** para poder realizar cálculos de agregación de forma parecida a los que hacemos en las bases de datos relacionales.

Operadores pipelines

- ❑ \$project : se utiliza para modificar el conjunto de datos de entrada, añadiendo, eliminando o recalculando campos para que la salida sea diferente.
- ❑ \$match: filtra la entrada para reducir el número de documentos, dejando solo los que cumplan las condiciones establecidas.
- ❑ \$limit: restringe el número de resultados al número indicado.
- ❑ \$skip: ignora un número determinado de registros, devolviendo los siguientes.
- ❑ \$unwind: convierte un array para devolverlo separado en documentos.
- ❑ \$group: agrupa documentos según una determinada condición.
- ❑ \$sort: ordena un conjunto de documentos según el campo especificado.
- ❑ \$geoNear: utilizado con datos geoespaciales, devuelve los documentos ordenados por proximidad según un punto geoespacial.

Aggregate Framework

```
db.nombre_coleccion.aggregate(  
{  
  $group: {  
    _id: '$campo',  
    totalAlgo: { $sum: '$Algo' },  
    cont: { $sum: 1 }  
  }  
}) ;
```

MapReduce

es un framework creado por Google, y pensado para realizar operaciones de forma paralela sobre grandes colecciones de datos. Este framework está compuesto de dos funciones principales: la función **Map** y la función **Reduce**. De ahí ese nombre tan original.

MapReduce

```
db.nombre_coleccion.mapReduce (mapp, reduce, {out: 'map_reduce_result'}) ;  
db.map_reduce_result.find() ;
```



mongo
DB

MUCHAS GRACIAS C: