Base de datos NoSQL

MongoDB

AGENDA

- Documento embebidos.
- Documentos embebidos operaciones
- Documentos embebidos relaciones uno a muchos
- Documentos embebidos relaciones uno a muchos – arreglos
- Documentos embebidos relaciones uno a muchos – arreglos – filtros y actualizaciones
- ObjectID
- Validación de documentos
- Índices
- Copias de respaldo
- Importación

MongoDB – Documentos Embebidos

```
persona
{ "_id": ObjetId(),
  "dni": "un dni"
  "apellido": "un apellido",
  "nombre": "un nombre",
}
```

```
Localidad
{ "_id": ObjectId(),
  "cod": "un código postal",
  "nombre": "una localidad",
  "prov": "una provincia"
}
```

```
persona
{ "_id": ObjetId(),
  "dni": "un dni"
  "apellido": "un apellido",
  "nombre": "un nombre",
  "id_loc": "un id localidad"
}
```



```
Localidad
{ "_id": ObjectId(),
  "cod": "un código postal",
  "nombre": "una localidad",
  "prov": "una provincia"
}
```

MongoDB – Documentos Embebidos

```
persona
{ "_id": ObjetId(),
  "dni": "un dni"
  "apellido": "un apellido",
  "nombre": "un nombre",

  "cod": "un código postal",
  "nombre": "una localidad",
  "prov": "una provincia"
}
```

```
persona
{ "_id": ObjetId(),
  "dni": "un dni"
  "apellido": "un apellido",
  "nombre": "un nombre",
  "localidad":{
    "cod": "un código postal",
    "nombre": "una localidad",
    "prov": "una provincia"
  }
}
```

MongoDB – Documentos Embebidos

- Documentos embebidos
  - Documentos que contienen otros documentos
  - Similar a las relaciones en el modelo relacional
- Ejemplo:
  - persona = {dni: “27707386”, apellido: “Marrero”, nombre: “Luciano”}
  - localidad = {cod: “7220”, nombre: “Monte”, prov: “Bs. As.” }
  - persona.localidad = localidad
  - db.personas.insert(persona)

mongoDB

MongoDB – Documentos Embebidos - Operaciones

- db.personas.find({“nombre”: “Luciano”}).pretty()
- db.personas.find({“localidad.prov”: “Buenos Aires”}).pretty()
- db.personas.find({“localidad.prov”: “Buenos Aires”}, {“localidad”: true}).pretty()
  - Retorna solo los datos del la localidad.
- db.personas.find({“localidad.prov”: “Buenos Aires”}, {“localidad.nombre”: true}).pretty()
  - Retorna solo el campo nombre de la localidad.

MongoDB – Documentos Embebidos - Operaciones

- db.personas.update({}, {$set: { “localidad.cod_postal”: 1900 } })
- db.personas.update({}, {$set: { “localidad.cod_postal”: 1900 } }, {multi: true})
- db.personas.remove({“localidad.cod_postal”: 1900})
- Probar:
  - fecha = {“dia”: 10, “mes”: 1, “año”: 1980}
  - db.personas.update({}, {$set: { fecha_nac: fecha }}, {multi: true})

mongoDB

MongoDB – Documentos Embebidos – Relación uno a muchos

```
persona
{ "_id": ObjetId(),
  "dni": "un dni"
  "apellido": "un apellido",
  "nombre": "un nombre",
  "localidad":{
    "cod": "un código postal",
    "nombre": "una localidad",
    "prov": "una provincia"
   }
   "teléfono_1": { "pref": "un prefijo 1", "num": "un número 1" }
   "teléfono_2": { "pref": "un prefijo 2", "num": "un número 2" }
   "teléfono_3": { "pref": "un prefijo 3", "num": "un número 3" }
}
```

MongoDB – Documentos Embebidos – Relación uno a muchos - Arreglos

```
persona
{ “_id”: ObjetId(),
  “dni”: “un dni”
  “apellido”: “un apellido”,
  “nombre”: “un nombre”,
  “localidad”:{
    “cod”: “un código postal”,
    “nombre”: “una localidad”,
    “prov”: “una provincia”
   }
   teléfonos:[
     { “pref”: “un prefijo 1”, “num”: “un número 1” },
     { “pref”: “un prefijo 2”, “num”: “un número 2” },
     { “pref”: “un prefijo 3”, “num”: “un número 3” }
   ]
}
```

MongoDB – Documentos Embebidos – Relación uno a muchos – Arreglos – Filtros

- db.personas.find({ “telefonos.pref”: “un perfijo”})
- db.personas.find({ telefonos: { $elemMatch: { pref: “un prefijo” }}})
- db.personas.find({ telefonos: { $elemMatch: { pref: “un prefijo”, num:” un número” }}})
- db.personas.find({ telefonos: { $elemMatch: { pref: “un prefijo”}}}, { _id: false, telefonos: true, dni: true})
- db.personas.find({ telefonos: { $elemMatch: { pref: “un prefijo”}}}, { _id: false, telefonos: {$elemMatch: { pref: “otro prefijo”}}, dni: true})
  - En este caso indicamos el sub documento con el que se quiere trabajar.
- db.personas.find({“telefonos.pref”: un prefijo }, { _id: false, “telefonos.$”: true})

MongoDB – Documentos Embebidos – Relación uno a muchos – Arreglos – Actualización

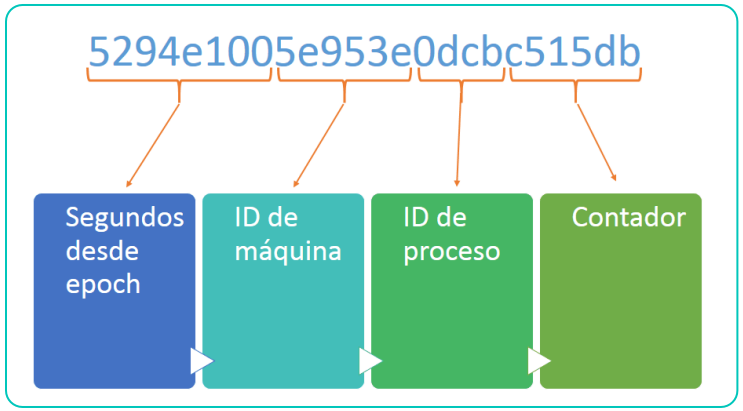
- db.personas.update({ “telefonos.pref”: “un prefijo”}, { $set: { “apellido”: “otro apellido” } })
- db.personas.update({ “telefonos.pref”: “un prefijo”}, { $set: { “telefonos.$”: { pref: “otro prefijo”, num: “ un numero”} } })
- db.personas.update({ “telefonos.pref”: “un prefijo”}, { $set: { “telefonos.$.num”: “un numero” } })

MongoDB – ObjectId

- Si el documento a insertar no tiene un "_id" definido, entonces MongoDB lo crea, caso contrario el documento tendrá el "_id" definido.
  - persona = {"_id" : "27707386", apellido: "Marrero", nombre: "Luciano" }
  - db.personas.insert(persona)
  - db.personas.find().pretty()
  - persona = {"_id" : 27707386, apellido: "Suarez", nombre: "Mariana" }
  - db.personas.insert(persona)
  - db.personas.find().pretty()

MongoDB – ObjectId

- MongoDB permite crear un Object ID
- new ObjectId()
  - Un ObjectId es tipo JSON binario (BSON) de 12 bytes que posee la siguiente estructura:
    - Los primeros 4 bytes son una marca de tiempo que indica el número de segundos transcurridos desde la fecha epoch de UNIX (el 1 de Enero de 1970). Esto nos da una referencia temporal de cada inserción y hace que los registros se guarden en disco aproximadamente en el orden de inserción.
    - Los 3 siguientes bytes son un identificador único de la máquina que ha generado el ObjectId. Por regla general es un hash obtenido a partir del nombre de red de la máquina. De esta forma varias máquinas de un sistema distribuido generan valores distintos para esta parte del identificador.
    - Los 2 siguientes son el identificador del proceso que genera el ObjectId. De este modo en una misma máquina si hay varios procesos ejecutándose a la vez (lo habitual) cada uno tendrá un valor diferente para esta parte.
    - Los 3 últimos bytes son un contador incremental. De esta forma se asegura un valor único para cada segundo (primer grupo) dentro de una misma máquina y un mismo proceso. Como son tres bytes eso nos da para almacenar hasta 256^3 valores diferentes (casi 17 millones) por cada proceso en un mismo segundo.



MongoDB – Colecciones – Validando documentos

- db.createCollection("mi coleccion")
- db.createCollection("mi coleccion", { validator: { $and: [{ nombre: { $type: "string"} }] } })
- db.createCollection("mi coleccion", { validator: { $and: [{ nombre: { $type: "string"} }, { genero: { $in: ["M", "F"] } }] } })
- db.createCollection("mi coleccion", { validator: { $and: [{ nombre: { $type: "string"} }, { genero: { $in: ["M", "F"] }}, { email: { $regex: /@/ } }] } })
- db.createCollection("mi coleccion", { validator: { $and: [{ nombre: { $type: "string"} }, { genero: { $in: ["M", "F"] }}, { email: { $regex: /@/ } }, {edad: { $exists: false }}] } })
- db.createCollection("mi coleccion", { validator: { $and: [{ "telefonos.num": { $type: "string"} }] } })

MongoDB - Índices

- Información sobre la búsqueda
  - db.personas.find({ apellido: “un apellido”}).explain(“executionStats”)
  - db.personas.find({ apellido: “un apellido”}).explain(“executionStats”).executionStats.executionTimeMillis
    - Retorna el tiempo de la consulta
- db.personas.createIndex({ apellido: 1})
  - Crea un índice por apellido de forma ascendente.

MongoDB – Copias de respaldo

- En la terminal:
  - mongodump --db nombre_base
  - mongorestore --db nombre_base_a_restaurar path
    - "path" es la ruta en donde están los archivos (BJSON) de la copia de respaldo.
  - mongodump --collection nombre_colección --db nombre_base
  - mongorestore --collection nombre_colección --db nombre_base path
    - "path" es la ruta en donde está el archivo BJSON de la copia de respaldo

MongoDB – Importar documentos

- mongoexport --db nombre_base --collection nombre_collection --out nombre_archivo.ext (csv, json, etc.)
- mongoimport --db nombre_base --collection nombre_collection --type csv --headerline --file nombre_archivo.csv
- mongoimport --db nombre_base --collection nombre_collection --file nombre_archivo.ext(por ejemplo un json)

GRACIAS