GA6-220501096-AA1-EV03 Y EV04

P.7 Una **base de datos NoSQL** (Not Only SQL) es un tipo de base de datos que **no utiliza el modelo relacional tradicional** basado en tablas y filas. En lugar de eso, las bases de datos NoSQL utilizan diferentes modelos de datos que permiten almacenar, recuperar y administrar datos de manera más **flexible y escalable**.

🔍 ¿Qué significa NoSQL?

El término **NoSQL** originalmente significaba *"No SQL"*, pero hoy se interpreta mejor como "**Not Only SQL**", es decir, "no solamente SQL", porque muchas bases de datos NoSQL pueden usar consultas similares a SQL o convivir con sistemas relacionales.

🔧 Características principales:

- **No usan esquemas fijos**: puedes guardar diferentes tipos de datos sin una estructura rígida.
- **Alta escalabilidad**: ideales para aplicaciones web y móviles con grandes volúmenes de datos.
- **Flexibilidad**: permiten trabajar con datos complejos como JSON, XML, imágenes, etc.
- **Distribuidas**: muchas están diseñadas para funcionar bien en sistemas distribuidos (varios servidores).

P.8 La **creación de objetos en bases de datos NoSQL** depende del tipo de base de datos que estés utilizando, ya que **NoSQL** no sigue un modelo relacional fijo como SQL. Existen varios tipos de bases NoSQL, y cada una maneja sus "objetos" o estructuras de datos de forma diferente.

PRINCIPALES DE BASES DE DATOS NoSQL

Tipo	Ejemplo de motor	Estructura principal
Documentales	MongoDB, CouchDB	Documentos JSON / BSON
Clave-Valor	Redis, Riak	Pares clave/valor
Columnar	Cassandra, HBase	Columnas y familias de columnas
Grafos	Neo4j, ArangoDB	Nodos y relaciones

1. **Documentales** (como MongoDB)
    - Almacenan documentos en formato JSON, BSON o XML.
    - Ejemplo:

```
{
 "nombre": "Juan",
 "edad": 30,
 "correo": "juan@example.com"
}
```

2. **Clave-valor** (como Redis)
    - Guardan los datos como pares clave-valor. Ejemplo: "usuario123" : "Juan"
3. **Columnares** (como Apache Cassandra)
    - Organizan los datos por columnas en lugar de filas.
4. **Basadas en grafos** (como Neo4j)
    - Excelentes para modelar relaciones complejas entre datos (como redes sociales).

✅ **¿Cuándo usar NoSQL?**

- Cuando tienes grandes cantidades de datos no estructurados o semi-estructurados.
- Cuando necesitas flexibilidad para cambiar el modelo de datos frecuentemente.
- Cuando requieres alto rendimiento y escalabilidad horizontal (más servidores).
- Para proyectos como aplicaciones móviles, big data, IoT, análisis en tiempo real, etc.

P.9 Cómo diferenciar NoSQL de SQL

Al utilizar el término NoSQL no se hace referencia a una base de datos o a un tipo de base de datos, sino que, más bien, es la definición de un conjunto de tipos de bases de datos que son diferentes a las bases de datos convencionales (relacionales). La principal diferencia entre las bases de datos de NoSQL y las bases de datos relacionales tradicionales es que las NoSQL emplean una forma de almacenamiento no estructurado o de estructuras diferentes a tablas. Simplificando la estructura de almacenamiento, es así como las NoSQL pueden concentrarse en procesar grandes volúmenes de datos.

A continuación, se muestra una comparación rápida de las funciones de NoSQL con las bases de datos relacionales.

Tabla 1

Comparación de las funciones de NoSQL

Característica	**Base de datos NoSQL**	**Bases de datos relacionales**
Desempeño	Alto	Bajo
Fiabilidad	Pobre	Buena
Disponibilidad	Buena	Buena
Consistencia	Pobre	Buena
Almacenamiento de datos	Optimizado para muchos datos	Mediano o grande
Estabilidad	Alta	Alta (pero más costosa)

P.10 En MongoDB, puedes pensar en la estructura así:

db → es la base de datos (el contenedor principal).

collection → es como una tabla en SQL, pero sin esquema fijo.

document → es como una fila en SQL, pero aquí es un objeto JSON/BSON con campos y valores.

Te lo explico con un ejemplo sencillo:

1️⃣ Base de datos (db)

Es el contenedor que agrupa colecciones relacionadas.

En MongoDB, para seleccionar o crear una base de datos usas:

use Restaurante

Aquí, Restaurante es el db. Si no existe, se crea cuando insertes algo.

2️⃣ Colección (collection)

Es un conjunto de documentos. No tiene un esquema rígido.

Ejemplo: una colección llamada clientes dentro de Restaurante.

db.createCollection("clientes")

3️⃣ Documentos (document)

Es la unidad mínima de datos, guardada como JSON/BSON.

Ejemplo de documento en clientes:

```
db.clientes.insertOne({
   nombre: "Ana López",
   telefono: "3001234567",
   direccion: "Calle 10 #5-20",
   pedidos: 5
})
```

📌 Relación resumida:

db → collection → document

Restaurante → clientes → { nombre: "Ana López", ... }

P.11 MongoDB no existe un comando literal DESCRIBE como en SQL, pero sí hay formas de "describir" o listar lo que tienes en tu db, tus colecciones y tus documentos.

El paso seria asi:

1️⃣ Ver todas las bases de datos

show dbs

📌 Lista las bases de datos existentes en el servidor MongoDB.

2️⃣ Ver la base de datos actual

db

📌 Muestra el nombre de la base de datos en la que estás trabajando.

3️⃣ Ver todas las colecciones de la base de datos actual

show collections

📌 Muestra todas las colecciones dentro de la db seleccionada.

4️⃣ Ver un documento de ejemplo de una colección

db.nombreColeccion.findOne()

📌 Muestra un solo documento de la colección para que veas su estructura.

Ejemplo:

db.clientes.findOne()

5️⃣ Ver todos los documentos de una colección (limitando para no saturar)

db.clientes.find().limit(5)

📌 Lista los primeros 5 documentos.

6️⃣ Conocer las "claves" o campos de una colección (similar a describir columnas)

MongoDB no tiene esquema fijo, pero puedes inspeccionar un documento para ver sus campos:

db.clientes.findOne()

O usar $project para listar solo las llaves:

```
db.clientes.aggregate([
  { $project: { _id: 0, llaves: { $objectToArray: "$$ROOT" } } }
])
```

💡 Ejemplo en práctica:

```
use Restaurante          // Seleccionar base
show collections         // Ver colecciones
db.clientes.findOne()    // Ver estructura de un documento
```

P.14 **documento estructurado** cumpliendo con los elementos solicitados para un trabajo sobre **Bases de Datos NoSQL**, usando **MongoDB** como ejemplo (uno de los motores más comunes de bases de datos documentales NoSQL). Este formato es adecuado para ser entregado como actividad escrita.

📄 **PORTADA**

SENA - Servicio Nacional de Aprendizaje
Ficha: [Número de ficha]
Nombre del Aprendiz: [Tu nombre completo]
Instructor: [Nombre del instructor]
Programa: Tecnólogo en Análisis y Desarrollo de Software
Actividad: Creación de objetos en base de datos NoSQL
Fecha: [Fecha actual]

📘 **INTRODUCCIÓN**

Las bases de datos NoSQL han transformado la manera de almacenar y gestionar datos no estructurados o semiestructurados, ofreciendo mayor flexibilidad frente a las tradicionales bases relacionales. Este documento presenta el diseño y manipulación de una colección en una base de datos NoSQL documental, específicamente en MongoDB, donde se representan objetos de tipo "carro" y se ejecutan operaciones de inserción, consulta y actualización sobre los mismos.

🎯 **OBJETIVO**

Aplicar los conocimientos adquiridos en el componente formativo "Bases de Datos NoSQL" mediante la creación de objetos tipo JSON, colecciones y consultas sobre datos almacenados en una base de datos documental con el objetivo de modelar información referente a vehículos.

🛠️ **SENTENCIAS DE CREACIÓN Y MANIPULACIÓN DE DATOS EN NoSQL (MongoDB)**

✅ 1. Selección o creación de base de datos

use vehiculosDB

✅ 2. Creación de colección "carros"

En MongoDB, las colecciones se crean automáticamente cuando se insertan los primeros documentos. No obstante, se puede forzar la creación con:

db.createCollection("carros")

✅ 3. Sentencia para crear un objeto JSON (carro)

```
{
  "placa": "ABC123",
  "numero_serie": "SN123456789",
  "modelo": 2020,
  "marca": "Toyota",
  "kilometraje": 35000,
  "tipo": "Sedán"
}
```

P.15 ✅ 4. Inserción de 5 documentos en la colección

```
db.carros.insertMany([
  {
    "placa": "ABC123",
    "numero_serie": "SN123456789",
    "modelo": 2020,
    "marca": "Toyota",
    "kilometraje": 35000,
    "tipo": "Sedán"
  },
  {
    "placa": "DEF456",
    "numero_serie": "SN987654321",
    "modelo": 2018,
```

```
    "marca": "Mazda",
    "kilometraje": 42000,
    "tipo": "Hatchback"
  },
  {
    "placa": "GHI789",
    "numero_serie": "SN112233445",
    "modelo": 2022,
    "marca": "Chevrolet",
    "kilometraje": 15000,
    "tipo": "SUV"
  },
  {
    "placa": "JKL012",
    "numero_serie": "SN556677889",
    "modelo": 2019,
    "marca": "Nissan",
    "kilometraje": 28000,
    "tipo": "Pickup"
  },
  {
    "placa": "MNO345",
    "numero_serie": "SN998877665",
    "modelo": 2021,
    "marca": "Hyundai",
    "kilometraje": 18000,
    "tipo": "Sedán"
  }
])
```

✅ **5. Consulta de todos los documentos en la colección**

db.carros.find()

✅ **6. Actualización de datos de un documento**

Por ejemplo, actualizar el kilometraje del carro con placa "DEF456":

db.carros.updateOne(

{ "placa": "DEF456" },

{ $set: { "kilometraje": 43000 } }

)

También se puede hacer una actualización múltiple:

db.carros.updateMany(

{ "tipo": "Sedán" },

{ $set: { "revisado": true } }

)

✅ **CONCLUSIONES**

La gestión de datos en bases NoSQL ofrece una gran flexibilidad y escalabilidad, facilitando el modelado de información compleja como objetos de tipo carro. MongoDB, como sistema documental, permite trabajar con estructuras JSON de forma natural, simplificando operaciones de inserción, consulta y actualización. La presente actividad demuestra la creación efectiva de estos objetos y su manipulación mediante sentencias específicas, fortaleciendo así los conocimientos prácticos del aprendiz.

P.21 El ejemplo paso a paso a ejecutar directamente en MongoDB, con sentencias completas y ordenadas.

una estructura de documento JSON para la colección parque (un registro de parqueadero de vehículos):

Evidencia 4

{

"placa": "ABC123",

"tipoVehiculo": "Carro",

"color": "Rojo",

"propietario": "Juan Pérez",

"horaEntrada": "2025-08-08T08:30:00",

"horaSalida": null

}

1️⃣ Crear la base de datos

use ParqueaderoDB

📌 Si no existe, se creará al insertar el primer documento.

2️⃣ Crear la colección parque

db.createCollection("parque")

3️⃣ Insertar 5 documentos

db.parque.insertMany([

{

placa: "ABC123",

tipoVehiculo: "Carro",

color: "Rojo",

propietario: "Juan Pérez",

horaEntrada: "2025-08-08T08:30:00",

horaSalida: null

},

{

placa: "XYZ789",

tipoVehiculo: "Moto",

color: "Negro",

propietario: "Ana López",

horaEntrada: "2025-08-08T09:00:00",

horaSalida: null

},

{

placa: "LMN456",

tipoVehiculo: "Camioneta",

color: "Blanco",

```
    propietario: "Carlos Gómez",
    horaEntrada: "2025-08-08T09:15:00",
    horaSalida: null
  },
  {
    placa: "PQR678",
    tipoVehiculo: "Carro",
    color: "Azul",
    propietario: "Laura Torres",
    horaEntrada: "2025-08-08T09:45:00",
    horaSalida: null
  },
  {
    placa: "JKL321",
    tipoVehiculo: "Moto",
    color: "Gris",
    propietario: "Pedro Ramírez",
    horaEntrada: "2025-08-08T10:00:00",
    horaSalida: null
  }
])
```

4 Actualizar el primer y último registro

Primer registro (placa "ABC123")

```
db.parque.updateOne(
  { placa: "ABC123" },
  { $set: { color: "Verde", propietario: "Juan Esteban Pérez" } }
)
```

Último registro (placa "JKL321")

```
db.parque.updateOne(
```

{ placa: "JKL321" },

{ $set: { color: "Negro", propietario: "Pedro Andrés Ramírez" } }

)

P.22 5 Listar toda la colección

db.parque.find().pretty()

6 Borrar el tercer documento

(Según nuestra inserción inicial, el tercero tiene placa "LMN456")

db.parque.deleteOne({ placa: "LMN456" })

7 Listar la colección nuevamente

db.parque.find().pretty()

8 Buscar por número de placa

Ejemplo: buscar placa "XYZ789".

db.parque.findOne({ pl

MongoDB Curso, Introducción Practica a NoSQL

https://www.youtube.com/watch?v=lWMemPN9t6Q

MongoDB EN 1 HORA – Aprende desde ceros

https://www.youtube.com/watch?v=YyfdOX-Clf4

Cómo Instalar MongoDB en Windows 11 (Guía TOTAL 2025)

https://www.youtube.com/watch?v=RNWdpWGNULw