Integración e interacción con bases de datos relacionales y no relacionales (MongoDB)

Presentado por: Julian Felipe Latorre

30 de julio de 2024

Contenido

¿Qué es MongoDB?

- Base de datos NoSQL orientada a documentos
- Almacena datos en documentos similares a JSON
- Esquema flexible
- Diseñada para escalabilidad y rendimiento

Operaciones CRUD en MongoDB

- Create (Crear): Insertar nuevos documentos
- Read (Leer): Consultar documentos existentes
- Update (Actualizar): Modificar documentos existentes
- Delete (Eliminar): Eliminar documentos

Ejemplos de operaciones CRUD

```
// Crear
db.usuarios.insertOne({ nombre: "Ana", edad: 28 })

// Leer
db.usuarios.find({ edad: { $gt: 25 } })

// Actualizar
db.usuarios.updateOne(
    { nombre: "Ana" },
    { $set: { edad: 29 } }
)

// Eliminar
db.usuarios.deleteOne({ nombre: "Ana" })
```

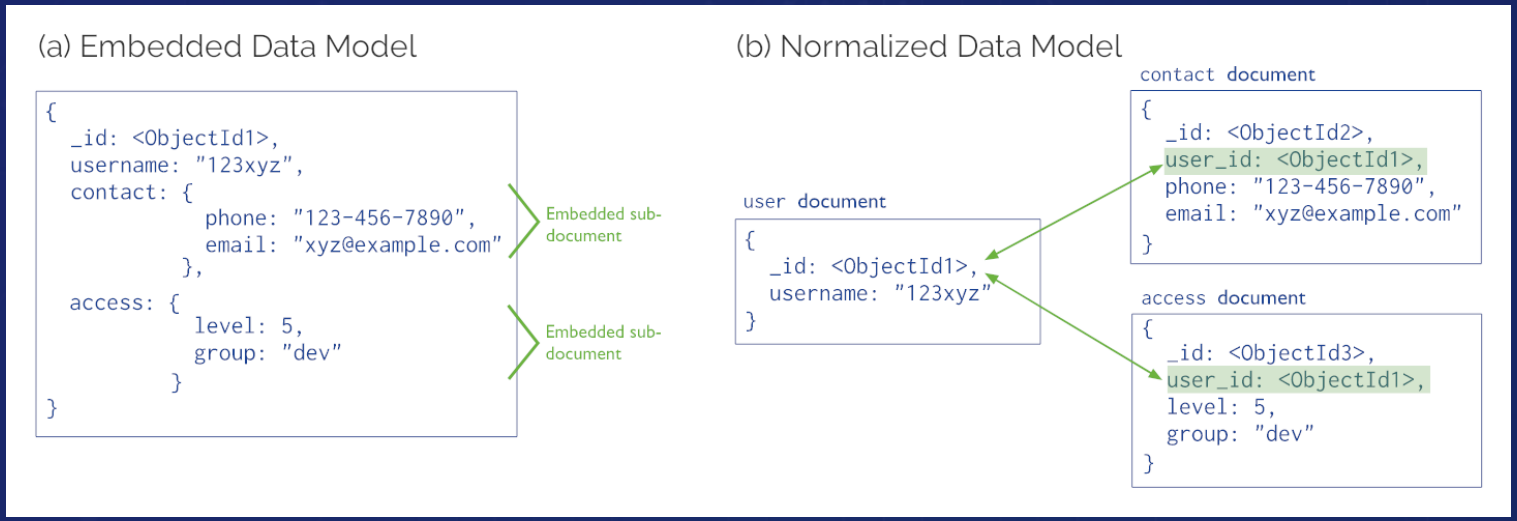
Documentos embebidos vs. Referencias

Documentos embebidos

- Datos relacionados juntos
- Relaciones one-to-few
- Mejor rendimiento en lecturas

Referencias

- Datos consultados por separado
- Relaciones one-to-many o many-to-many
- Mejor para datos compartidos



Ejemplo de documento embebido

```
{
    _id: ObjectId("5f8a7b"),
    nombre: "Juan Pérez",
    edad: 30,
    direcciones: [
        {
            tipo: "casa",
            calle: "Calle 123",
            ciudad: "Bogota"
        },
        {
            tipo: "trabajo",
            calle: "Calle 321",
            ciudad: "Bogota"
        }
    ]
}
```

Documento con relaciones

```
// Documento de usuario
{
    _id: ObjectId("5f8a7b"),
    nombre: "Juan Pérez",
    edad: 30,
    direcciones: [ObjectId("5f8a7c"), ObjectId("5f8a7d")],
    pedidos: [ObjectId("5f8a7e"), ObjectId("5f8a7f")]
}

// Documentos de direcciones
{
    _id: ObjectId("5f8a7c"),
    usuario_id: ObjectId("5f8a7b"),
    tipo: "casa",
    direccion: "Calle 321 No. 123",
            ciudad: "Bogotá",
            codigoPostal: "266001"
}
{
    _id: ObjectId("5f8a7d"),
    usuario_id: ObjectId("5f8a7b"),
    tipo: "trabajo",
    direccion: "Calle 123 No. 321",
    ciudad: "Bogotá",
    codigoPostal: "266002"
}

// Documentos de pedidos
{
    _id: ObjectId("5f8a7e"),
```

Tipos de índices en MongoDB

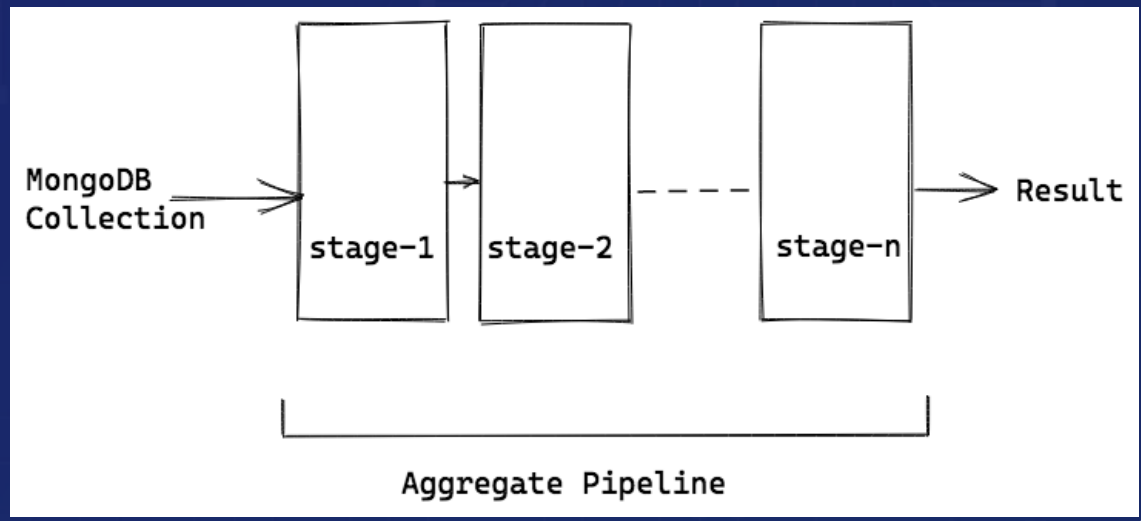
- Índices simples
- Índices compuestos
- Índices multikey
- Índices geoespaciales
- Índices de texto

Creación y análisis de índices

```
1 // Crear un índice
2 db.usuarios.createIndex({ "nombre": 1, "edad": -1 })
3
4 // Analizar una consulta
5 db.usuarios.find({ "edad": { $gt: 30 } })
6             .explain("executionStats")
```

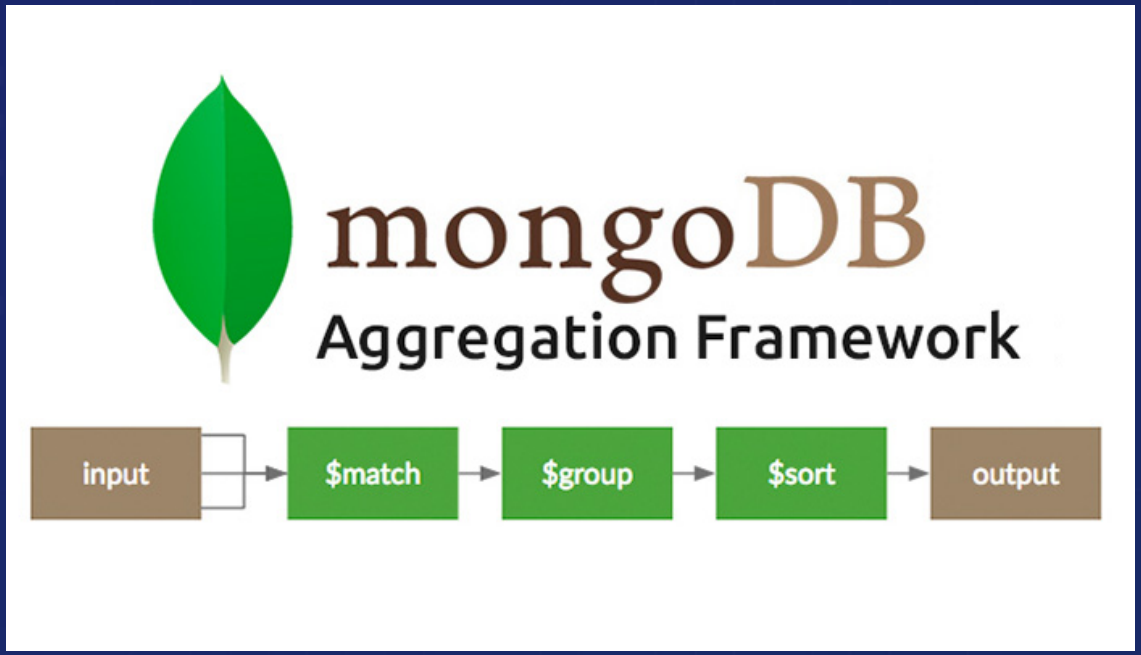
Pipeline de agregación

- $match: Filtrar documentos
- $group: Agrupar documentos
- $project: Dar forma a los documentos
- $sort: Ordenar documentos
- $limit y $skip: Paginar resultados
- $unwind: Descomponer arrays
- $lookup: Realizar "joins"



Pipeline de agregación

- $match: Filtrar documentos
- $group: Agrupar documentos
- $sort: Ordenar documentos

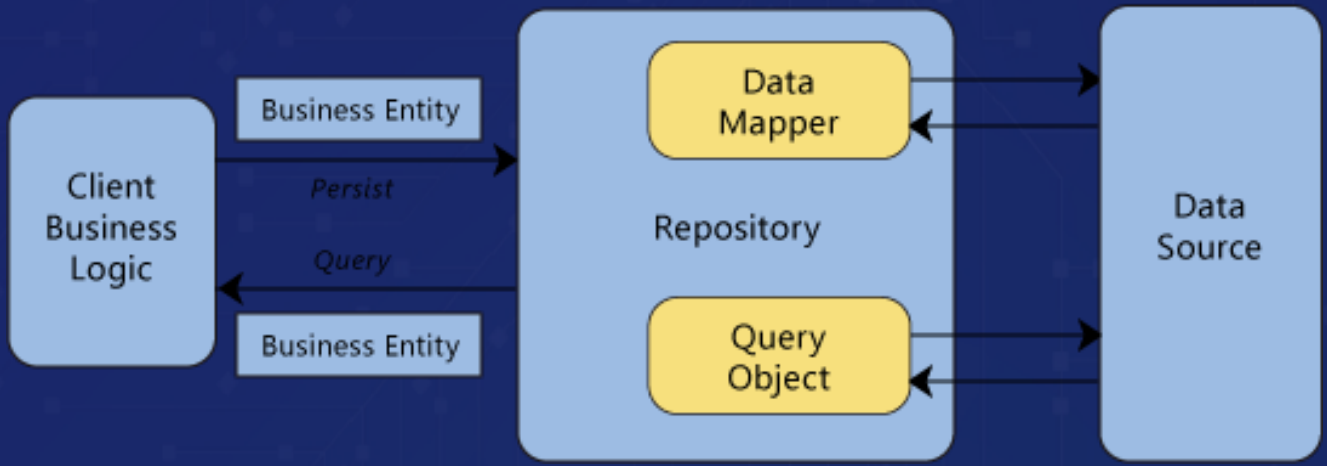


Ejemplo de agregación

```
db.ventas.aggregate([
    { $match: { fecha: { $gte: new Date("2023-01-01") } } },
    { $group: {
        _id: "$categoria",
        totalVentas: { $sum: "$monto" },
        promedioVenta: { $avg: "$monto" }
    }},
    { $sort: { totalVentas: -1 } }
])
```

Patrón Repository

- Abstrae la lógica de acceso a datos
- Permite cambiar fácilmente entre bases de datos
- Mejora la mantenibilidad y testabilidad



Implementación del Repository

```
class UsuarioRepository {
    async obtenerTodos() { ... }
    async obtenerPorId(id) { ... }
    async crear(usuario) { ... }
    async actualizar(id, usuario) { ... }
    async eliminar(id) { ... }
}

// Implementaciones específicas para MongoDB y MySQL
class MongoUsuarioRepository extends UsuarioRepository { ...
    }
class MySQLUsuarioRepository extends UsuarioRepository { ...
    }
```

Conclusiones

- MongoDB ofrece flexibilidad y escalabilidad
- El modelado de datos es crucial para el rendimiento
- La indexación y la agregación son herramientas poderosas
- El patrón Repository permite una fácil adaptación a diferentes bases de datos