Operaciones básicas en MongoDB (CRUD)

Objetivo:
Aprender a **crear**, **consultar**, **modificar** y **eliminar** documentos en MongoDB usando Compass y Mongo Shell. Entender cada parte del comando, comparando con SQL para ayudar a la transición.

¿Qué es CRUD?

CRUD es una sigla que viene del inglés:

- **Create** → Crear
- **Read** → Leer (consultar)
- **Update** → Actualizar
- **Delete** → Eliminar

Son las 4 acciones básicas que se pueden hacer sobre cualquier conjunto de datos, ya sea en una base relacional como SQL Server o en MongoDB.

Antes de empezar:
En MongoDB, muchas operaciones existen en dos versiones: **One** y **Many**. Se utilizan en distintas ocasiones:

One: Realiza la acción sobre el **primer documento** que cumple con el filtro. Si hay más de uno, los demás son ignorados. Es útil cuando sabés que hay un solo documento y hace más eficiente la operación.

Many: Realiza la acción sobre **todos** los documentos que cumplan con el filtro.

Acción	Versión "One"	Versión "Many"
Crear	`insertOne()`	`insertMany()`
Leer	`find()`, `findOne()`	-
Actualizar	`updateOne()`	`updateMany()`
Eliminar	`deleteOne()`	`deleteMany()`

¿Cuál usar?

Si trabajás con un **_id**, usá siempre la versión **One**. Es único, por lo tanto no tiene sentido modificar o eliminar más de uno.

Si querés aplicar una operación a **varios documentos** que cumplen una condición, usá **Many**.

Crear base de datos y colección

Antes de insertar documentos, es útil saber cómo crear una base de datos y una colección.

Crear una base de datos:

MongoDB cambia de base de datos automáticamente con el comando `use`:

```
use empresa
```

Esto cambia a la base de datos llamada `empresa`. Si no existe, se creará en cuanto agregues datos.

Luego de ejecutar este comando, usaremos `db` para referirnos a esa base de datos seleccionada.

Para visualizar qué base de datos está actualmente en uso, podemos ejecutar el siguiente comando:

```
db
```

Esto nos muestra la base de datos a la que se está haciendo referencia. En este caso, nos debería mostrar `empresa`.

Crear una colección:

Las colecciones se crean automáticamente al insertar el primer documento. Sin embargo, también podés crearla explícitamente:

```
db.createCollection("empleados")
```

Luego de crearla, utilizaremos `db.empleados` para referirnos a la colección llamada `empleados` y de esta forma podemos realizar las acciones necesarias sobre esta colección.

Crear documentos – insertOne() y insertMany()

`insertOne()`

Esta operación agrega un **nuevo documento** en una colección. El argumento recibido es un objeto (documento).

```
db.empleados.insertOne({
  nombre: "Juan",
  apellido: "Pérez",
  edad: 32,
  puesto: "Desarrollador"
})
```

Explicación:

- `db.empleados` → accedemos a la colección **empleados**.
- `.insertOne({...})` → insertamos un documento con sus campos y valores.
- Mongo genera automáticamente un campo **_id** que identifica al documento.

`insertMany()`

Esta operación agrega **varios documentos** a la colección, recibiendo un **array de objetos**.

```
db.empleados.insertMany([
  { nombre: "Ana", apellido: "Gómez", edad: 28, puesto: "Diseñadora"
},
  { nombre: "Carlos", apellido: "López", edad: 40, puesto: "Gerente"
}
])
```

Explicación:

- Se usa un array `[ ]` con varios objetos `{}`.
- Ideal para **cargar múltiples registros** de una sola vez.

Respuesta esperada:

```
{
  acknowledged: true,
  insertedIds: {
    "0": ObjectId(...),
    "1": ObjectId(...)
  }
}
```

Leer documentos – find() y findOne()

La operación `find()` permite **consultar documentos** dentro de una colección.

Leer todos los documentos:

```
db.empleados.find()
```

Devuelve **todos los documentos** de la colección.

Para verlos de forma legible:

```
db.empleados.find().pretty()
```

Esto muestra cada documento en varias líneas, facilitando su lectura.

Aplicar filtros:

```
db.empleados.find({ nombre: "Juan" })
```

Filtra los documentos cuyo campo `nombre` sea exactamente "Juan".

Leer un solo documento:

Si solo querés obtener el **primer resultado**:

```
db.empleados.findOne({ nombre: "Juan" })
```

`findOne()` está pensado para cuando sabés que solo te interesa un documento. Te lo devuelve directamente. En cambio, `find()` siempre devuelve una lista de posibles coincidencias, aunque sea una sola.

Filtrar por _id

MongoDB guarda los `_id` como **objetos especiales** (`ObjectId`). No se puede comparar directamente con un string.

Forma correcta:

```
db.empleados.find({ _id: ObjectId("6651bd7e87f9aefad6432a67") })
```

Forma incorrecta:

```
db.empleados.find({ _id: "6651bd7e87f9aefad6432a67" })
```

Regla práctica: Siempre que filtres por `_id`, usá `ObjectId(...)`.

Actualizar documentos – updateOne() y updateMany()

`updateOne()`

Modifica el **primer documento** que cumpla con el filtro.

```
db.empleados.updateOne(
  { nombre: "Juan" },
  { $set: { puesto: "Líder Técnico" } }
)
```

Explicación:

- El primer argumento es el **filtro de búsqueda**.
- `$set` indica los **campos a modificar** y su nuevo valor.

Modificar varios campos:

```
db.empleados.updateOne(
  { nombre: "Juan" },
  { $set: { puesto: "Líder Técnico", edad: 33 } }
)
```

`updateMany()`

Modifica **todos los documentos** que cumplan con el filtro.

```
db.empleados.updateMany(
  { puesto: "Diseñadora" },
  { $set: { puesto: "Diseñadora UX" } }
)
```

Respuesta esperada:

```
{
  acknowledged: true,
  matchedCount: 2,
  modifiedCount: 2
}
```

Consejo: Si usás `_id`, siempre aplicá `updateOne()`.

Eliminar documentos – deleteOne() y deleteMany()

`deleteOne()`

Elimina el **primer documento** que cumpla con el filtro.

```
db.empleados.deleteOne({ apellido: "López" })
```

Eliminar por _id:

```
db.empleados.deleteOne({_id: ObjectId("6651bd7e87f9aefad6432a67")})
```

Este es el **método más preciso** para eliminar un documento específico.

`deleteMany()`

Elimina **todos los documentos** que cumplan con el filtro.

```
db.empleados.deleteMany({ edad: 60 })
```

Cuidado:

```
db.empleados.deleteMany({})
```

❌ Esto elimina **todos los documentos** de la colección. Solo usar en pruebas o con mucho cuidado.

Respuesta esperada:

```
{
  acknowledged: true,
  deletedCount: 3
}
```

Resumen:

Acción	Método	Resultado esperado
Crear	`insertOne()`	Inserta un documento, devuelve `insertedId`
Crear	`insertMany()`	Inserta varios, devuelve `insertedIds`
Leer	`find()`	Devuelve documentos con o sin filtros
Leer	`findOne()`	Devuelve un solo documento
Actualizar	`updateOne()`	Modifica el primero que cumpla el filtro
Actualizar	`updateMany()`	Modifica todos los que cumplan el filtro
Eliminar	`deleteOne()`	Borra un documento
Eliminar	`deleteMany()`	Borra todos los que cumplan el filtro

Errores comunes a evitar:

- Usar `_id` como string en lugar de `ObjectId()`
- Olvidar `$set` y sobrescribir todo el documento
- No usar filtro en `deleteMany()` y eliminar toda la colección
- Asumir que `find()` devuelve un documento en vez de un cursor (si querés uno solo: `findOne()`)

Actividad práctica

1. **Insertá 3 empleados** distintos en Compass.
2. **Consultá uno de ellos por nombre** desde Mongo Shell.
3. **Modificá dos campos** (por ejemplo: edad y puesto) con `updateOne()`.
4. **Insertá varios empleados** con `insertMany()` y luego hacé un `updateMany()`.
5. **Eliminá uno por su `_id` y otro por edad con** `deleteMany()`.