CRUD EN MONGODB

CRUD - Update

- Para actualizar un documento, MongoDB proporciona operadores de actualización, como $set, para modificar los valores de los campos.
  - db.collection.updateOne(<filter>, <update>, <options>)
  - db.collection.updateMany(<filter>, <update>, <options>)
  - db.collection.replaceOne(<filter>, <update>, <options>)
- Para usar los operadores de actualización, el parámetro de los siguientes métodos de actualización debe ser un documento que cumpla lo siguiente:

```
{
  <update operator>: { <field1>: <value1>, ... },
  <update operator>: { <field2>: <value2>, ... },
  ...
}
```

- **Importante**: el operador $set creará el campo si este no existe.

CRUD - Update

db.collection.updateOne() actualiza el primer documento que cumpla la condición.

```
> db.library.updateOne(
      { item: "paper" },
      {
            $set: { "size.uom": "cm", status: "P" },
            $currentDate: { lastModified: true }
      }
  )
```

- Esta operación utiliza el operador $set para actualizar el valor del campo size.uom a "cm" y el valor del campo de status a "P".
- De igual forma, utiliza el operador $currentDate para actualizar el valor del campo lastModified a la fecha actual.
- Si el campo lastModified no existe, $currentDate creará el campo.

CRUD - Update

db.collection.updateMany() actualiza todos los documentos que cumplan la condición.

```
> db.library.updateMany(
      { "qty": { $lt: 50 } },
      {
            $set: { "size.uom": "in", status: "P" },
            $currentDate: { lastModified: true }
      }
  )
```

- Esta operación utiliza el operador $set para actualizar el valor del campo size.uom a "in" y el valor del campo de status a "P",
- Igual que el método anterior, el operador $currentDate actualiza el valor del campo lastModified a la fecha actual. Si el campo lastModified no existe, $currentDate creará el campo.

CRUD - Update

db.collection.replaceOne() reemplaza todo el contenido de un documento, excepto el campo **_id**, por un documento completamente nuevo indicado en el segundo argumento de esta función.

```
> db.library.replaceOne(
   { item: "paper" },
   { item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 40 } ] }    )
```

- Todas las operaciones de escritura en MongoDB son atómicas en el nivel de un solo documento.
- El documento de reemplazo puede tener diferentes campos del documento original.
- En el documento de reemplazo se puede omitir el campo _id ya que el campo _id es inmutable. Sin embargo, si incluye el campo _id, debe tener el mismo valor que el valor actual.
- Una vez establecido, no puede actualizar el valor del campo _id ni se puede reemplazar un documento existente con un documento de reemplazo que tenga un valor de campo _id diferente.
- El campo _id es siempre el primer campo en el documento.
- Las actualizaciones que incluyen el cambio de nombre de los campos pueden resultar en la reordenación de los campos en el documento.

CRUD - Update

db.collection.update () actualiza o reemplaza un solo documento que coincide con un filtro específico o actualiza todos los documentos que coinciden con un filtro específico.

```
> db.collection.update(
   <query>,
   <update>,
   {
     upsert: <boolean>,
     multi: <boolean>,
     writeConcern: <document>,
     collation: <document>,
     arrayFilters: [ <filterdocument1>, ... ],
     hint:  <document|string>        // Sólo en la version MongoDB 4.2
   }
)
```

- Por defecto, este método actualiza un solo documento. Para actualizar varios documentos, use la opción multi.

CRUD - Update

Opciones de db.collection.update ()

- query: criterio para realizar la operación de update.
- update: modificaciones a aplicar sobre el documento.
- upsert: si se establece en verdadero, crea un nuevo documento cuando ningún documento coincide con los criterios de consulta. El valor predeterminado es falso, que no inserta un nuevo documento cuando no se encuentra ninguna coincidencia.
- multi: si se establece en verdadero, actualiza varios documentos que cumplen con los criterios de consulta. Si se establece en falso, actualiza un documento. El valor predeterminado es falso.
- writeConcern: Opcional, un documento que expresa la preocupación de escritura. Omita utilizar la preocupación de escritura predeterminada w: 1.
- collation: La clasificación permite a los usuarios especificar reglas específicas del idioma para la comparación de cadenas, como las reglas para mayúsculas y minúsculas.
- arrayFilter: Opcional, un array de documentos de filtro que determinan qué elementos del array modificar para una operación de actualización en un campo de array.

CRUD - Update

db.collection.update ()

```
> db.books.update(
   { _id: 1 },
   {
     $inc: { stock: 5 },
     $set: {
       item: "ABC123",
       "info.publisher": "2222",
       tags: [ "software" ],
       "ratings.1": { by: "xyz", rating: 3 }
     }
   }
)
```

```
UPDATE books
SET    stock = stock + 5
       item = "ABC123"
       publisher = 2222
       pages = 430
       tags = "software"
       rating_authors = "ijk,xyz"
       rating_values = "4,3"
WHERE  _id = 1
```

CRUD - Update

db.collection.update ()

```
> db.books.update(
    { item: "ZZZ135" },
    {
      item: "ZZZ135",
      stock: 5,
      tags: [ "database" ]
    },
    { upsert: true }
)
```

```
WriteResult({
  "nMatched" : 0,
  "nUpserted" : 1,
  "nModified" : 0,
  "_id" : ObjectId("5da78973835b2f1c75347a83")
 })
```

CRUD - Update

db.collection.save() este método usa el comando **insert** o el comando **update** y tiene en cuenta la preocupación de escritura de forma predeterminada. Para especificar una preocupación de escritura diferente, incluya la preocupación de escritura en el parámetro de opciones.

Insert

- Si el documento no contiene un campo _id, entonces este método llama al método **insert**(). Durante la operación, el shell mongo creará un ObjectId y lo asignará al campo _id.

Update

- Si el documento contiene un campo _id, entonces este método es equivalente a una actualización con la opción upsert establecida en true y el predicado de consulta en el campo _id.

```
> db.products.save( { _id: 100, item: "water", qty: 30 } )
```

```
{ "_id" : 100, "item" : "water", "qty" : 30 }
```

```
> db.products.save( { _id : 100, item : "juice" } )
```

➡ `{ "_id" : 100, "item" : "juice" }`

CRUD - Delete

- Para eliminar todos los documentos de una colección, pase un documento de filtro vacío {} al método db.<collection>.deleteMany().
- El método devuelve un documento con el estado de la operación.
- Puede especificar criterios o filtros que identifiquen los documentos a eliminar. Los filtros usan la misma sintaxis que las operaciones de lectura.

```
> db.products.deleteMany( { status : "A" } )
```

- Para eliminar como máximo un solo documento que coincida con un filtro especifico (varios documentos pueden coincidir con el filtro indicado), utilice el método db.<collection>.deleteOne().
- Las operaciones de eliminación no eliminan los índices, incluso si se eliminan todos los documentos de una colección.
- Todas las operaciones de escritura en MongoDB son atómicas en el nivel de un solo documento.

CRUD - Delete

- El método db.<collection>.remove() elimina todos los documentos que coinciden con la expresión de consulta.
- Se puede utilizar la opción justOne para limitar la operación a la eliminación de un solo documento.
- Al eliminar varios documentos, la operación de eliminación puede intercalar con otras operaciones de lectura y / o escritura en la colección.
- Para eliminar todos los documentos de una colección, llame al método remove con un documento de consulta vacío {}.
- Esta operación no es equivalente al método db.<collection>.drop().
- Para eliminar todos los documentos de una colección, puede ser más eficiente usar el método drop() para borrar toda la colección, incluidos sus índices, y luego volver a crear la colección y reconstruir los índices.

```
> db.products.remove( { } )
```

```
> db.products.remove( { qty: { $gt: 20 } } )
```

```
> db.products.remove( { qty: { $gt: 20 } }, true )
```

CRUD - Delete

- El método db.<collection>.drop() elimina una colección o vista de la base de datos. El método también elimina los índices asociados a la colección borrada.
- Este método emplea un bloqueo exclusivo **en la colección especificada** durante la duración de la operación.
- Todas las operaciones posteriores en la colección deben esperar hasta que este libere el bloqueo.
- En versiones inferiores de MongoDB 4.2, db.collection.drop () aplica un bloqueo exclusivo **en la base de datos principal**, bloqueando todas las operaciones en la base de datos y todas sus colecciones hasta que se completó la operación.

```
> db.products.drop()
```

```
> db.dropDatabase()
```

- El método db.dropDatabase() elimina la base de datos actual o en uso y todos sus ficheros asociados.

Muchas gracias por tu atención