AWD 1111

MongoDB CRUD Operations Guide

Create, Read, Update, Delete

CRUD Overview

CRUD stands for Create, Read, Update, and Delete — the four basic operations for working with data in any database. This guide covers each operation using the concert_demo database.

Operation	MongoDB Method	SQL Equivalent
Create	insertOne(), insertMany()	INSERT INTO
Read	find(), findOne()	SELECT
Update	updateOne(), updateMany()	UPDATE
Delete	deleteOne(), deleteMany()	DELETE

Database Used in Examples
All examples use the concert_demo database with collections: concerts, venues, reviews, and fans.

1. Create Operations (INSERT)

Create operations add new documents to a collection. MongoDB provides two methods for inserting documents.

1.1 insertOne()

Inserts a single document into a collection.

Syntax

```
db.collection.insertOne( { document } )
```

Example: Add a new fan

```
db.fans.insertOne({
name: "Sarah Connor",
email: "sarah@example.com",
membershipLevel: "Silver",
state: "CA"
})
```

Example: Add a new venue

```
db.venues.insertOne({
name: "Red Rocks Amphitheatre",
city: "Morrison",
state: "CO",
capacity: 9525,
location: {
type: "Point",
coordinates: [-105.2083, 39.6655]
}
})
```

Auto-generated _id
If you don't provide an _id field, MongoDB automatically generates a unique ObjectId for you.

1.2 insertMany()

Inserts multiple documents in a single operation. Pass an array of documents.

Syntax

```
db.collection.insertMany( [ { doc1 }, { doc2 }, ... ] )
```

Example: Add multiple fans

```
db.fans.insertMany([
{
name: "Mike Wilson",
email: "mike@example.com",
membershipLevel: "Gold",
state: "TX"
},
{
name: "Emily Chen",
email: "emily@example.com",
membershipLevel: "Platinum",
state: "WA"
}
])
```

Return Value
insertOne() returns the inserted _id. insertMany() returns an object with all inserted _ids.

1.3 Inserting Documents with References

When inserting documents that reference other collections, use ObjectId to create the reference.

Example: Add a review for a concert

```
// First, find the concert's _id
db.concerts.findOne({ artist: "Pearl Jam" }, { _id: 1 })

// Then insert the review with that _id
db.reviews.insertOne({
fan_name: "RockFan2024",
concert_id: ObjectId("5f8d04b3b54764421b7156c3"),
text: "Epic performance! Eddie Vedder still has it!",
date: new Date()
})
```

Example: Add a concert with venue references

```
db.concerts.insertOne({
tourName: "Summer Vibes Tour",
artist: "New Artist",
genres: ["Pop", "Electronic"],
releaseYear: 2025,
ticketPrice: 75,
attendance: 15000,
venue_ids: [
ObjectId("5f8d04b3b54764421b7156d1"), // The Pageant
ObjectId("5f8d04b3b54764421b7156d2") // Enterprise Center
],
setList: ["Song One", "Song Two", "Song Three"]
})
```

2. Read Operations (SELECT)

Read operations retrieve documents from a collection. These are your query operations.

2.1 find()

Returns all documents that match the query criteria.

Syntax

```
db.collection.find( { filter }, { projection } )
```

Example: Find all concerts

```
db.concerts.find()
```

Example: Find concerts by genre

```
db.concerts.find({ genres: "Rock" })
```

Example: Find concerts with tickets under $100

```
db.concerts.find({ ticketPrice: { $lt: 100 } })
```

Example: Find venues in a specific state

```
db.venues.find({ state: "MO" })
```

2.2 findOne()

Returns only the first document that matches the query. Useful when you expect a single result.

Example: Find a specific fan by email

```
db.fans.findOne({ email: "jim@ranken.edu" })
```

Example: Find a concert by artist

```
db.concerts.findOne({ artist: "Taylor Swift" })
```

2.3 Projection (Selecting Fields)

The second parameter controls which fields are returned. Use 1 to include, 0 to exclude.

Example: Return only name and email

```
db.fans.find({}, { name: 1, email: 1 })
```

Example: Exclude the setList field

```
db.concerts.find({}, { setList: 0 })
```

Example: Return only artist and ticketPrice, exclude _id

```
db.concerts.find({}, { _id: 0, artist: 1, ticketPrice: 1 })
```

Projection Rule
You cannot mix inclusion (1) and exclusion (0) in the same projection, except for _id which can always be excluded.

2.4 Query Operators

MongoDB provides operators for complex queries:

Operator	Description	Example
$eq	Equals	{ ticketPrice: { $eq: 100 } }
$ne	Not equals	{ state: { $ne: 'MO' } }
$gt	Greater than	{ attendance: { $gt: 50000 } }
$gte	Greater than or equal	{ ticketPrice: { $gte: 100 } }
$lt	Less than	{ capacity: { $lt: 10000 } }
$lte	Less than or equal	{ releaseYear: { $lte: 2020 } }
$in	Matches any in array	{ state: { $in: ['MO', 'IL'] } }
$nin	Not in array	{ genres: { $nin: ['Classical'] } }

Example: Multiple conditions (implicit AND)

```
db.concerts.find({
ticketPrice: { $gte: 50 },
releaseYear: 2024
})
```

Example: Using $or

```
db.concerts.find({
$or: [
{ genres: "Rock" },
{ genres: "Jazz" }
]
})
```

2.5 Sorting and Limiting

sort() - Order results

```
// Sort by ticketPrice ascending (1 = low to high)
db.concerts.find().sort({ ticketPrice: 1 })

// Sort by attendance descending (-1 = high to low)
db.concerts.find().sort({ attendance: -1 })
```

limit() - Restrict number of results

```
// Get only the first 3 results
db.concerts.find().limit(3)

// Get the 5 most expensive concerts
db.concerts.find().sort({ ticketPrice: -1 }).limit(5)
```

skip() - Pagination

```
// Skip first 2, get next 2 (page 2 with 2 per page)
db.concerts.find().skip(2).limit(2)
```

2.6 Counting Documents

```
// Count all documents in a collection
db.concerts.countDocuments()

// Count documents matching a filter
db.concerts.countDocuments({ genres: "Rock" })

// Count fans by membership level
db.fans.countDocuments({ membershipLevel: "Gold" })
```

3. Update Operations (UPDATE)

Update operations modify existing documents in a collection. MongoDB provides methods for updating one or many documents.

3.1 updateOne()

Updates the first document that matches the filter.

Syntax

```
db.collection.updateOne( { filter }, { update } )
```

> **Important!**
> Updates require an update operator like $set. Without it, you'll get an error or replace the entire document.

3.2 The $set Operator

$set replaces the value of a field. If the field doesn't exist, it creates it.

Example: Update a fan's membership level

```
db.fans.updateOne(
{ email: "jane@example.com" },
{ $set: { membershipLevel: "Gold" } }
)
```

Example: Update concert ticket price

```
db.concerts.updateOne(
{ artist: "Pearl Jam" },
{ $set: { ticketPrice: 95 } }
)
```

Example: Update multiple fields at once

```
db.venues.updateOne(
{ name: "The Pageant" },
{
$set: {
capacity: 2300,
city: "St. Louis"
}
}
)
```

Example: Update nested fields

```
db.venues.updateOne(
{ name: "Enterprise Center" },
{ $set: { "location.coordinates": [-90.2026, 38.6270] } }
)
```

3.3 The $push Operator

$push adds an element to an array field.

Example: Add a genre to a concert

```
db.concerts.updateOne(
{ artist: "Taylor Swift" },
{ $push: { genres: "Synth-pop" } }
)
```

Example: Add a song to a setList

```
db.concerts.updateOne(
{ artist: "Pearl Jam" },
{ $push: { setList: "Black" } }
)
```

Example: Add a venue to a concert tour

```
db.concerts.updateOne(
{ artist: "The Rolling Stones" },
{ $push: { venue_ids: ObjectId("5f8d04b3b54764421b7156d1") } }
)
```

3.4 The $pull Operator

$pull removes elements from an array that match a condition.

Example: Remove a genre from a concert

```
db.concerts.updateOne(
{ artist: "Taylor Swift" },
{ $pull: { genres: "Country" } }
)
```

Example: Remove a song from setList

```
db.concerts.updateOne(
{ artist: "Pearl Jam" },
{ $pull: { setList: "Jeremy" } }
)
```

3.5 The $inc Operator

$inc increments (or decrements) a numeric field by a specified value.

Example: Increase ticket price by $10

```
db.concerts.updateOne(
{ artist: "Taylor Swift" },
{ $inc: { ticketPrice: 10 } }
)
```

Example: Decrease attendance by 500

```
db.concerts.updateOne(
{ artist: "Pearl Jam" },
{ $inc: { attendance: -500 } }
)
```

Example: Increase venue capacity

```
db.venues.updateOne(
{ name: "The Pageant" },
{ $inc: { capacity: 100 } }
)
```

3.6 updateMany()

Updates all documents that match the filter.

Example: Give all Gold members a free upgrade

```
db.fans.updateMany(
{ membershipLevel: "Gold" },
{ $set: { membershipLevel: "Platinum" } }
)
```

Example: Increase all ticket prices by 5%

```
// Note: For percentage increase, you'd typically use aggregation
// This example increases all prices by $5
db.concerts.updateMany(
{},
{ $inc: { ticketPrice: 5 } }
)
```

3.7 Update Operators Summary

Operator	Description	Use Case
$set	Set field value	Change any field value
$unset	Remove a field	Delete a field from document
$inc	Increment number	Increase/decrease counters
$push	Add to array	Add item to list
$pull	Remove from array	Remove item from list
$addToSet	Add unique to array	Add only if not exists
$rename	Rename field	Change field name

Combining Operators
You can use multiple update operators in a single update: { $set: {...}, $push: {...}, $inc: {...} }

4. Delete Operations (DELETE)

Delete operations remove documents from a collection. Use with caution — deleted data cannot be recovered!

4.1 deleteOne()

Deletes the first document that matches the filter.

Syntax

```
db.collection.deleteOne( { filter } )
```

Example: Delete a fan by email

```
db.fans.deleteOne({ email: "mike@example.com" })
```

Example: Delete a specific review

```
db.reviews.deleteOne({ _id: ObjectId("5f8d04b3b54764421b7156e1") })
```

Example: Delete a venue by name

```
db.venues.deleteOne({ name: "Red Rocks Amphitheatre" })
```

Warning!
Always test your filter with find() first to make sure you're targeting the right documents before deleting.

4.2 deleteMany()

Deletes all documents that match the filter.

Example: Delete all Silver members

```
db.fans.deleteMany({ membershipLevel: "Silver" })
```

Example: Delete all reviews before a date

```
db.reviews.deleteMany({ date: { $lt: new Date("2024-01-01") } })
```

Example: Delete all concerts from a specific year

```
db.concerts.deleteMany({ releaseYear: 2010 })
```

4.3 Delete All Documents

To delete all documents in a collection, pass an empty filter:

```
// Delete ALL documents in a collection (use with extreme caution!)
db.reviews.deleteMany({})
```

> **Danger Zone!**
> deleteMany({}) with an empty filter deletes EVERYTHING in the collection. This cannot be undone!

4.4 drop() - Delete Entire Collection

To completely remove a collection (including all documents and indexes):

```
// Delete the entire collection
db.reviews.drop()
```

5. CRUD Quick Reference

Create

```
// Insert one document
db.collection.insertOne({ field: "value" })

// Insert multiple documents
db.collection.insertMany([{ doc1 }, { doc2 }])
```

Read

```
// Find all
db.collection.find()

// Find with filter
db.collection.find({ field: "value" })

// Find one
db.collection.findOne({ field: "value" })

// Find with projection
db.collection.find({}, { field1: 1, field2: 1 })

// Sort, limit, skip
db.collection.find().sort({ field: -1 }).limit(10).skip(5)
```

Update

```
// Update one document
db.collection.updateOne({ filter }, { $set: { field: "new value" } })

// Update many documents
db.collection.updateMany({ filter }, { $set: { field: "new value" } })

// Add to array
db.collection.updateOne({ filter }, { $push: { arrayField: "new item" } })

// Increment a number
db.collection.updateOne({ filter }, { $inc: { numField: 10 } })
```

Delete

```
// Delete one document
db.collection.deleteOne({ field: "value" })

// Delete many documents
db.collection.deleteMany({ field: "value" })

// Delete all documents in collection
db.collection.deleteMany({})
```

6. SQL to MongoDB Translation

For those familiar with SQL, here's how common operations translate:

SQL	MongoDB
INSERT INTO fans VALUES(...)	db.fans.insertOne({...})
SELECT * FROM concerts	db.concerts.find()
SELECT * FROM concerts WHERE year=2024	db.concerts.find({releaseYear: 2024})
SELECT artist, price FROM concerts	db.concerts.find({}, {artist:1, ticketPrice:1})
SELECT * FROM concerts ORDER BY price	db.concerts.find().sort({ticketPrice: 1})
SELECT TOP 5 * FROM concerts	db.concerts.find().limit(5)
UPDATE fans SET level='Gold' WHERE...	db.fans.updateOne({...}, {$set: {membershipLevel:'Gold'}})
DELETE FROM fans WHERE...	db.fans.deleteOne({...})

7. Reference Links

For more information, see the official MongoDB documentation:

- Insert Documents: https://www.mongodb.com/docs/manual/tutorial/insert-documents/
- Query Documents: https://www.mongodb.com/docs/manual/tutorial/query-documents/
- Update Documents: https://www.mongodb.com/docs/manual/tutorial/update-documents/
- Delete Documents: https://www.mongodb.com/docs/manual/tutorial/remove-documents/
- Update Operators: https://www.mongodb.com/docs/manual/reference/operator/update/