mongoDB

This version of the manual is no longer supported.

SQL to MongoDB Mapping Chart

On this page

In addition to the charts that follow, you might want to consider the Frequently Asked Questions section for a selection of common questions about MongoDB.

Terminology and Concepts

The following table presents the various SQL terminology and concepts and the corresponding MongoDB terminology and concepts.

SQL Terms/Concepts	MongoDB Terms/Concepts
database	database
table	collection
row	document or BSON document
column	field
index	index
table joins	embedded documents and linking

SQL Terms/Concepts	MongoDB Terms/Concepts
primary key Specify any unique column or column combination as primary key.	primary key In MongoDB, the primary key is automatically set to the _id field.
aggregation (e.g. group by)	aggregation pipeline See the SQL to Aggregation Mapping Chart.

Executables

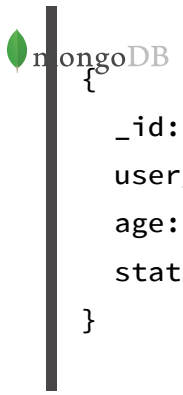
The following table presents some database executables and the corresponding MongoDB executables. This table is *not* meant to be exhaustive.

	MongoDB	MySQL	Oracle	Informix	DB2
Database Server	`mongod`	`mysqld`	`oracle`	`IDS`	`DB2 Server`
Database Client	`mongo`	`mysql`	`sqlplus`	`DB-Access`	`DB2 Client`

Examples

The following table presents the various SQL statements and the corresponding MongoDB statements. The examples in the table assume the following conditions:

- The SQL examples assume a table named `users`.
- The MongoDB examples assume a collection named `users` that contain documents of the following prototype:



```
{
  _id: ObjectId("509a8fb2f3f4948bd2f983a0"),
  user_id: "abc123",
  age: 55,
  status: 'A'
}
```

Create and Alter

The following table presents the various SQL statements related to table-level actions and the corresponding MongoDB statements.

SQL Schema Statements	MongoDB Schema Statements

`CREATE TABLE users ( id MEDIUMINT NOT NULL AUTO_INCREMENT, user_id Varchar(30), age Number, status char(1), PRIMARY KEY (id) )`	Implicitly created on first `insert()` operation. The primary key `_id` is automatically added if `_id` field is not specified.
	
	`db.users.insert( { user_id: "abc123", age: 55, status: "A" } )`
	
	However, you can also explicitly create a collection:
	
	`db.createCollection("users")`

SQL Schema Statements	MongoDB Schema Statements

`ALTER TABLE users`
`ADD join_date DATETIME`
	Collections do not describe or enforce the structure of its documents; i.e. there is no structural alteration at the collection level.
	
	However, at the document level, `update()` operations can add fields to existing documents using the `$set` operator.
	
	`db.users.update(`
	`{ },`
	`{ $set: { join_date: new Date() } },`
	`{ multi: true }`
	`)`

`ALTER TABLE users`
`DROP COLUMN join_date`
	Collections do not describe or enforce the structure of its documents; i.e. there is no structural alteration at the collection level.
	
	However, at the document level, `update()` operations can remove fields from documents using the `$unset` operator.
	
	`db.users.update(`
	`{ },`
	`{ $unset: { join_date: "" } },`
	`{ multi: true }`
	`)`

`CREATE INDEX idx_user_id_asc` `ON users(user_id)`	`db.users.ensureIndex( { user_id: 1 } )`

`CREATE INDEX`
`idx_user_id_asc_age_desc`
`ON users(user_id, age DESC)`
	`db.users.ensureIndex( { user_id: 1, age: -1 } )`

`DROP TABLE users`	`db.users.drop()`

For more information, see `db.collection.insert()`, `db.createCollection()`, `db.collection.update()`, `$set`, `$unset`, `db.collection.ensureIndex()`, indexes, `db.collection.drop()`, and Data Modeling Concepts.

Insert

The following table presents the various SQL statements related to inserting records into tables and the corresponding MongoDB statements.

SQL INSERT Statements	MongoDB insert() Statements
`INSERT INTO users(user_id, age, status) VALUES ("bcd001", 45, "A")`	`db.users.insert( { user_id: "bcd001", age: 45, status: "A" } )`

For more information, see `db.collection.insert()`.

Select

The following table presents the various SQL statements related to reading records from tables and the corresponding MongoDB statements.

SQL SELECT Statements	MongoDB find() Statements
`SELECT * FROM users`	`db.users.find()`
`SELECT id, user_id, status FROM users`	`db.users.find( { }, { user_id: 1, status: 1 } )`
`SELECT user_id, status FROM users`	`db.users.find( { }, { user_id: 1, status: 1, _id: 0 } )`

SQL SELECT Statements	MongoDB find() Statements

`SELECT *`
`FROM users`
`WHERE status = "A"`
	`db.users.find(`
	`  { status: "A" }`
	`)`

`SELECT user_id, status`
`FROM users`
`WHERE status = "A"`
	`db.users.find(`
	`  { status: "A" },`
	`  { user_id: 1, status: 1, _id: 0 }`
	`)`

`SELECT *`
`FROM users`
`WHERE status != "A"`
	`db.users.find(`
	`  { status: { $ne: "A" } }`
	`)`

`SELECT *`
`FROM users`
`WHERE status = "A"`
`AND age = 50`
	`db.users.find(`
	`  { status: "A",`
	`  age: 50 }`
	`)`

`SELECT *`
`FROM users`
`WHERE status = "A"`
`OR age = 50`
	`db.users.find(`
	`  { $or: [ { status: "A" } ,`
	`  { age: 50 } ] }`
	`)`

`SELECT *`
`FROM users`
`WHERE age > 25`
	`db.users.find(`
	`  { age: { $gt: 25 } }`
	`)`

`SELECT *`
`FROM users`
`WHERE age < 25`
	`db.users.find(`
	`  { age: { $lt: 25 } }`
	`)`

`SELECT *`
`FROM users`
`WHERE age > 25`
`AND age <= 50`
	`db.users.find(`
	`  { age: { $gt: 25, $lte: 50 } }`
	`)`

SQL SELECT Statements	MongoDB find() Statements
`SELECT * FROM users WHERE user_id like "%bc%"`	`db.users.find( { user_id: /bc/ } )`
`SELECT * FROM users WHERE user_id like "bc%"`	`db.users.find( { user_id: /^bc/ } )`
`SELECT * FROM users WHERE status = "A" ORDER BY user_id ASC`	`db.users.find( { status: "A" } ).sort( { user_id: 1 } )`
`SELECT * FROM users WHERE status = "A" ORDER BY user_id DESC`	`db.users.find( { status: "A" } ).sort( { user_id: -1 } )`

`SELECT COUNT(*) FROM users`	`db.users.count()`
	or
	`db.users.find().count()`

`SELECT COUNT(user_id) FROM users`	`db.users.count( { user_id: { $exists: true } } )`
	or
	`db.users.find( { user_id: { $exists: true } } ).count()`

`SELECT COUNT(*) FROM users WHERE age > 30`	`db.users.count( { age: { $gt: 30 } } )`
	or
	`db.users.find( { age: { $gt: 30 } } ).count()`

SQL SELECT Statements	MongoDB find() Statements
`SELECT DISTINCT(status) FROM users`	`db.users.distinct( "status" )`

`SELECT * FROM users LIMIT 1`	`db.users.findOne()`
	or
	`db.users.find().limit(1)`

`SELECT * FROM users LIMIT 5 SKIP 10`	`db.users.find().limit(5).skip(10)`
`EXPLAIN SELECT * FROM users WHERE status = "A"`	`db.users.find( { status: "A" } ).explain()`

For more information, see `db.collection.find()`, `db.collection.distinct()`, `db.collection.findOne()`, `$ne` `$and`, `$or`, `$gt`, `$lt`, `$exists`, `$lte`, `$regex`, `limit()`, `skip()`, `explain()`, `sort()`, and `count()`.

Update Records

The following table presents the various SQL statements related to updating existing records in tables and the corresponding MongoDB statements.

SQL Update Statements	MongoDB update() Statements
`UPDATE users SET status = "C" WHERE age > 25`	`db.users.update( { age: { $gt: 25 } }, { $set: { status: "C" } }, { multi: true } )`

SQL Update Statements	MongoDB update() Statements
`UPDATE users SET age = age + 3 WHERE status = "A"`	`db.users.update( { status: "A" }, { $inc: { age: 3 } }, { multi: true } )`

For more information, see `db.collection.update()`, `$set`, `$inc`, and `$gt`.

Delete Records

The following table presents the various SQL statements related to deleting records from tables and the corresponding MongoDB statements.

SQL Delete Statements	MongoDB remove() Statements
`DELETE FROM users WHERE status = "D"`	`db.users.remove( { status: "D" } )`
`DELETE FROM users`	`db.users.remove({})`

For more information, see `db.collection.remove()`.

Additional Resources

- Transitioning from SQL to MongoDB (Presentation)
- Best Practices for Migrating from RDBMS to MongoDB (Webinar)
- RDBMS to MongoDB Migration Guide
- SQL vs. MongoDB Day 1-2
- SQL vs. MongoDB Day 3-5
- MongoDB vs. SQL Day 18
- MongoDB and MySQL Compared
- MongoDB Database Modernization Consulting Package