aponxi / **sql-mongo_comparison.md**

Last active 3 days ago • Report abuse

MongoDb Cheat Sheets

`sql-mongo_comparison.md`

SQL to MongoDB Mapping Chart

SQL to MongoDB Mapping Chart¶

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.

Executables¶

The following table presents the MySQL/Oracle executables and the corresponding MongoDB executables.

	MySQL/Oracle	MongoDB
Database Server	mysqld/oracle	*mongod*
Database Client	mysql/sqlplus	*mongo*

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
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 framework See the *SQL to Aggregation Framework Mapping Chart*.

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	Reference

`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")`
		See `insert()` and `createCollection()` for more information.

`ALTER TABLE users` `ADD join_date DATETIME`	Collections do not describe or enforce the structure of the constituent documents. See the Schema Design wiki page for more information.	See `update()` and `$set` for more information on changing the structure of documents in a collection.
`ALTER TABLE users` `DROP COLUMN join_date`	Collections do not describe or enforce the structure of the constituent documents. See the Schema Design wiki page for more information.	See `update()` and `$set` for more information on changing the structure of documents in a collection.
`CREATE INDEX idx_user_id_asc` `ON users(user_id)`	`db.users.ensureIndex( { user_id: 1 } )`	See `ensureIndex()` and *indexes* for more information.

SQL Schema Statements	MongoDB Schema Statements	Reference
`CREATE INDEX idx_user_id_asc_age_desc ON users(user_id, age DESC)`	`db.users.ensureIndex( { user_id: 1, age: -1 } )`	See `ensureIndex()` and *indexes* for more information.
`DROP TABLE users`	`db.users.drop()`	See `drop()` for more information.

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	Reference
`INSERT INTO users(user_id, age, status) VALUES ("bcd001", 45, "A")`	`db.users.insert( { user_id: "bcd001", age: 45, status: "A" } )`	See `insert()` for more information.

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	Reference
`SELECT * FROM users`	`db.users.find()`	See `find()` for more information.
`SELECT id, user_id, status FROM users`	`db.users.find( { }, { user_id: 1, status: 1 } )`	See `find()` for more information.
`SELECT user_id, status FROM users`	`db.users.find( { }, { user_id: 1, status: 1, _id: 0 } )`	See `find()` for more information.
`SELECT * FROM users WHERE status = "A"`	`db.users.find( { status: "A" } )`	See `find()` for more information.

SQL SELECT Statements	MongoDB find() Statements	Reference
`SELECT user_id, status FROM users WHERE status = "A"`	`db.users.find( { status: "A" }, { user_id: 1, status: 1, _id: 0 } )`	See `find()` for more information.
`SELECT * FROM users WHERE status != "A"`	`db.users.find( { status: { $ne: "A" } } )`	See `find()` and `$ne` for more information.
`SELECT * FROM users WHERE status = "A" AND age = 50`	`db.users.find( { status: "A", age: 50 } )`	See `find()` and `$and` for more information.
`SELECT * FROM users WHERE status = "A" OR age = 50`	`db.users.find( { $or: [ { status: "A" } , { age: 50 } ] } )`	See `find()` and `$or` for more information.
`SELECT * FROM users WHERE age > 25`	`db.users.find( { age: { $gt: 25 } } )`	See `find()` and `$gt` for more information.
`SELECT * FROM users WHERE age < 25`	`db.users.find( { age: { $lt: 25 } } )`	See `find()` and `$lt` for more information.
`SELECT * FROM users WHERE age > 25 AND age <= 50`	`db.users.find( { age: { $gt: 25, $lte: 50 } } )`	See `find()`, `$gt`, and `$lte` for more information.
`SELECT * FROM users WHERE user_id like "%bc%"`	`db.users.find( { user_id: /bc/ } )`	See `find()` and `$regex` for more information.
`SELECT * FROM users WHERE user_id like "bc%"`	`db.users.find( { user_id: /^bc/ } )`	See `find()` and `$regex` for more information.
`SELECT * FROM users WHERE status = "A" ORDER BY user_id ASC`	`db.users.find( { status: "A" } ).sort( { user_id: 1 } )`	See `find()` and `sort()` for more information.

SQL SELECT Statements	MongoDB find() Statements	Reference
`SELECT * FROM users WHERE status = "A" ORDER BY user_id DESC`	`db.users.find( { status: "A" } ).sort( { user_id: -1 } )`	See `find()` and `sort()` for more information.
`SELECT COUNT(*) FROM users`	`db.users.count()` *or* `db.users.find().count()`	See `find()` and `count()` for more information.
`SELECT COUNT(user_id) FROM users`	`db.users.count( { user_id: { $exists: true } } )` *or* `db.users.find( { user_id: { $exists: true } } ).count()`	See `find()`, `count()`, and `$exists` for more information.
`SELECT COUNT(*) FROM users WHERE age > 30`	`db.users.count( { age: { $gt: 30 } } )` *or* `db.users.find( { age: { $gt: 30 } } ).count()`	See `find()`, `count()`, and `$gt` for more information.
`SELECT DISTINCT(status) FROM users`	`db.users.distinct( "status" )`	See `find()` and `distinct()` for more information.
`SELECT * FROM users LIMIT 1`	`db.users.findOne()` *or* `db.users.find().limit(1)`	See `find()`, `findOne()`, and `limit()` for more information.
`SELECT * FROM users LIMIT 5 SKIP 10`	`db.users.find().limit(5).skip(10)`	See `find()`, `limit()`, and `skip()` for more information.
`EXPLAIN SELECT * FROM users WHERE status = "A"`	`db.users.find( { status: "A" } ).explain()`	See `find()` and `explain()` for more information.

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	Reference
`UPDATE users SET status = "C" WHERE age > 25`	`db.users.update( { age: { $gt: 25 } }, { $set: { status: "C" } }, { multi: true } )`	See `update()`, `$gt`, and `$set` for more information.
`UPDATE users SET age = age + 3 WHERE status = "A"`	`db.users.update( { status: "A" } , { $inc: { age: 3 } }, { multi: true } )`	See `update()`, `$inc`, and `$set` for more information.

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	Reference
`DELETE FROM users WHERE status = "D"`	`db.users.remove( { status: "D" } )`	See `remove()` for more information.
`DELETE FROM users`	`db.users.remove( )`	See `remove()` for more information.

ankitsari commented on Jun 24, 2017

Hi
UPDATE tb1 SET col1 = co1.toLowerCase(), col2 = col2.toLowerCase()

I need convert query into mongodb. is it possible to without foreach function.? or can we do using aggregate function with update.

imbenwolf commented on Aug 18, 2017 • edited

> Hi
> UPDATE tb1 SET col1 = co1.toLowerCase(), col2 = col2.toLowerCase()
>
> I need convert query into mongodb. is it possible to without foreach function.? or can we do using aggregate function with update.

this link should do the trick: https://docs.mongodb.com/manual/reference/operator/update/rename/

mehhrad commented on Jun 2, 2018

thanks but what about querying and embeded objects ?