NoSQL Databases with MongoDB

Practical Work Instructions for Group Tasks

Academic Year 2025/2026

Course Instructor: Diogo Ribeiro

Repository: `https://github.com/diogoribeiro7/nosql-databases-labs`

January 9, 2026

Abstract

This document provides comprehensive instructions for the practical group work component of the NoSQL Databases course. Students will work collaboratively through five core MongoDB labs, progressing from basic CRUD operations to advanced topics including replication and aggregation. Each group will maintain a dedicated repository folder, follow structured submission guidelines, and complete a final project demonstrating real-world MongoDB application design. The course emphasizes hands-on learning with over 30 sample datasets and automated testing infrastructure.

Contents

1 Course Overview

1.1 Learning Objectives

Upon successful completion of this practical work, students will be able to:

- Design and implement document-based database schemas
- Perform complex queries and aggregations in MongoDB
- Optimize database performance through indexing strategies
- Configure replication for high availability
- Apply NoSQL best practices to real-world scenarios
- Collaborate effectively using Git-based workflows

1.2 Technical Requirements

Important Note

Ensure your development environment meets these requirements before starting:

- MongoDB 6.x or 7.0 (Community Edition or Atlas)
- Node.js v20+ with npm
- MongoDB Shell (mongosh) latest version
- Git for version control
- Docker (optional but recommended)
- 10GB free disk space for datasets

2 Group Work Organization

2.1 Team Formation

Tip

Form groups of 2-3 students by Week 2. Register your group by creating a folder in the repository: `group-work/group_XX/` where XX is your assigned group number (01-23).

2.2 Folder Structure

Your group folder must follow this organization:

```
group-work/
  group_05/                    # Your group folder
    README.md                  # Group members and overview
    project/                   # Final project
      README.md
      architecture.md
      queries/
      data/
      tests/
```

Listing 1: Required Group Folder Structure

2.3 Collaboration Workflow

2.3.1 Git Workflow

1. **Fork** the main repository on GitHub
2. **Clone** your fork locally:

```
git clone https://github.com/<your-username>/nosql-databases-
    labs.git

```

3. **Create a branch** for each lab:

```
git checkout -b group_05-lab01

```

4. **Commit** with descriptive messages:

```
git add group-work/group_05/lab01/
git commit -m "group_05: complete lab01 CRUD operations"

```

5. **Push** and create a pull request:

```
git push origin group_05-lab01

```

2.3.2 Code Review Process

Each submission undergoes peer and instructor review:

- Automated tests run via GitHub Actions
- Code quality checks (ESLint, formatting)

- Manual review of design decisions
- Feedback provided via PR comments

3 Final Project Requirements

3.1 Project Scope

Design and implement a complete MongoDB-backed application addressing a real-world scenario.

Minimum Requirements:

- 3+ interconnected collections
- 20+ diverse queries (CRUD + aggregation)
- Performance optimization with indexes
- Data validation and error handling
- Comprehensive documentation

3.2 Suggested Project Topics

- E-commerce platform
- Social media analytics
- IoT sensor management
- Healthcare records system
- Real estate marketplace
- Event ticketing system
- Learning management system
- Supply chain tracking
- Restaurant ordering system
- Travel booking platform

4 Assessment Criteria

4.1 Project Evaluation

Final project assessment criteria:

- **Requirements Coverage (20%)**: Meets all specifications
- **Data Modeling (20%)**: Appropriate schema design
- **Query Complexity (20%)**: Diverse, real-world queries
- **Performance (15%)**: Optimized with proper indexing
- **Code Quality (15%)**: Clean, maintainable code

- **Documentation (10%)**: Clear, comprehensive docs

Important Note

Late submissions receive a 10% penalty per day. Extensions must be requested 48 hours in advance with valid justification.

5 Tools and Resources

5.1 Development Tools

- **MongoDB Compass**: GUI for database exploration
- **mongosh**: Command-line interface

5.2 Available Datasets

The repository includes 30+ ready-to-use datasets:

- sample_airbnb (listings, reviews)
- sample_mflix (movies, theaters)
- sample_analytics (accounts, customers)
- sample_supplies (sales data)
- sakila-db (film rental)
- ColoradoScooters (geospatial)
- crunchbase (startups)
- books, restaurants, students

5.3 Testing Infrastructure

Run automated tests to validate your work:

```
# Test specific lab
npm run test:lab01

# Test all labs
npm run test:labs

# Check code quality
npm run lint

# Generate coverage report
npm run test:coverage

# Validate group submission
node group-work/scripts/group_submission_validator.js
```

Listing 2: Testing Commands

5.4 Getting Help

- **GitHub Discussions**: Post questions with appropriate labels
- **Office Hours**: Weekly sessions
- **Documentation**: `docs/` folder in repository
- **Issue Tracker**: Report bugs or suggest improvements
- **Email**: Contact instructor for private matters

6 Best Practices

6.1 Code Standards

- Use consistent naming conventions (camelCase for JavaScript)
- Comment complex logic and business rules
- Handle errors gracefully with try-catch blocks
- Never commit credentials or secrets
- Follow the DRY principle (Don't Repeat Yourself)

6.2 Performance Optimization

Tip

Key optimization strategies:

- Create indexes before running queries
- Use projection to limit returned fields
- Implement pagination for large result sets
- Monitor slow queries with profiler
- Batch operations when possible

6.3 Documentation Guidelines

Every submission must include:

1. **README.md**: Overview, setup instructions, team members
2. **NOTES.md**: Design decisions, challenges, learnings
3. **Code comments**: Inline explanations for complex logic
4. **Performance metrics**: Query execution times, index impact

7 Common Pitfalls and Solutions

7.1 Frequent Mistakes

Common Mistake	Solution
Forgetting to switch databases	Always use `use <database>` first
Incorrect JSON syntax	Validate with online JSON validator
Missing indexes on queries	Run explain() before production
Unbounded document growth	Use references for growing arrays
Not handling connection errors	Implement retry logic
Hardcoding credentials	Use environment variables

Table 1: Common Issues and Resolutions

7.2 Debugging Tips

1. Enable MongoDB logging for detailed diagnostics
2. Use `.explain("allPlansExecution")` for query analysis
3. Check replica set status with `rs.status()`
4. Monitor performance with `db.currentOp()`
5. Validate data integrity with `db.collection.validate()`

8 Submission Checklist

Before submitting any lab or project:

Pre-Submission Checklist

- ☐ Pull latest changes from main branch
- ☐ All tests pass locally (`npm test`)
- ☐ Code is properly formatted (`npm run format`)
- ☐ No linting errors (`npm run lint`)
- ☐ Documentation is complete and accurate
- ☐ Screenshots/outputs included where required
- ☐ Sensitive data removed or anonymized
- ☐ Branch follows naming convention
- ☐ Commit messages are descriptive
- ☐ Pull request template completed

9 Conclusion

This practical work provides hands-on experience with MongoDB and NoSQL concepts through progressive skill-building exercises. Success requires consistent effort, effective collaboration, and attention to best practices.

Remember:

- Start early and work incrementally
- Ask questions when stuck
- Test thoroughly before submission
- Document your learning journey
- Collaborate effectively with your team

Good luck with your NoSQL journey!

A Quick Reference

A.1 Essential MongoDB Commands

```
// Database operations
show dbs                    // List all databases
use mydb                    // Switch to database
db.dropDatabase()           // Delete current database
```

```
5
6  // Collection operations
7  show collections             // List collections
8  db.createCollection("col")   // Create collection
9  db.col.drop()                // Delete collection
10
11 // CRUD operations
12 db.col.insertOne({...})      // Insert single document
13 db.col.find({...})           // Query documents
14 db.col.updateOne({...})      // Update single document
15 db.col.deleteMany({...})     // Delete multiple documents
16
17 // Indexing
18 db.col.createIndex({...})    // Create index
19 db.col.getIndexes()          // List indexes
20 db.col.dropIndex("name")     // Remove index
21
22 // Aggregation
23 db.col.aggregate([...])      // Run aggregation pipeline
24 db.col.count({...})          // Count documents
25 db.col.distinct("field")     // Get distinct values
26
27 // Replication
28 rs.initiate()                // Initialize replica set
29 rs.status()                  // Check replica status
30 rs.add("host:port")          // Add replica member
```

Listing 3: MongoDB Quick Reference

A.2 Git Workflow Summary

```
1  # Initial setup
2  git clone <repository-url>
3  git remote add upstream <original-repo-url>
4
5  # Starting new work
6  git checkout main
7  git pull upstream main
8  git checkout -b group_XX-labYY
9
10 # Saving work
11 git add .
12 git commit -m "Descriptive message"
13 git push origin group_XX-labYY
14
15 # Creating pull request
16 # Go to GitHub and click "New Pull Request"
17 # Select your branch and create PR with template
18
```

```
# Updating your fork
git fetch upstream
git checkout main
git merge upstream/main
git push origin main
```

Listing 4: Git Commands for Group Work