CSE13202	Unstructured Database	L	T	P	C
Version 2.0		2	0	1	3
Pre-requisite/Exposure	RDBMS Concepts, Programming Concepts				
Co-requisite	Logical Ability				

Course Objectives:

1. Understand Unstructured Database Fundamentals: Explore document, key-value, wide-column, graph, and vector data models and relate them to OLTP, OLAP, data warehouses, and data lakes.
2. Develop Data Handling and Ingestion Skills: Gain practical experience in data ingestion, extraction, preprocessing, and querying using MQL and MongoDB Atlas.
3. Master Core MongoDB Operations: Apply MongoDB CRUD, cursor management, and the aggregation framework to build efficient queries and data pipelines.
4. Apply MongoDB Indexing and Advanced Features: Use indexing strategies, WiredTiger tuning, GridFS, Atlas Search, Atlas Vector Search, and Flask/PyMongo to build and optimise applications.
5. Work with Modern NoSQL Systems and Streaming: Operate Redis and Neo4j; build a Kafka pipeline integrated with MongoDB; perform semantic search with Atlas Vector Search.
6. Apply Backup, Recovery, and Production Strategies: Design and execute backup, disaster recovery, and MongoDB Atlas production deployment plans ensuring data durability and high availability.

Course Outcomes:

CO1: Select Appropriate Data Models: Justify the choice of NoSQL or analytical paradigm (document, key-value, wide-column, graph, vector, OLTP, OLAP, data warehouse, data lake) for a given use case.

CO2: Manage Data Ingestion and Extraction: Ingest, extract, and preprocess unstructured data; write MQL queries; differentiate ETL/ELT for operational and analytical workloads.

CO3: Execute Core MongoDB Operations: Perform CRUD operations, manage cursors, and construct multi-stage aggregation pipelines on real datasets.

CO4 [Modified]: MapReduce removed. CO4: Optimise MongoDB with Indexing and Advanced Features: Create and evaluate indexes, tune WiredTiger, use GridFS, and build a Flask API connected to MongoDB Atlas.

CO5: Operate Modern NoSQL and Streaming Systems: Use Redis and Neo4j for their respective use cases; build a Kafka-to-MongoDB streaming pipeline; query Atlas Vector Search.

CO6: Design Backup and Recovery Plans: Perform mongodump/mongorestore, configure Atlas backups, and harden a deployment with RBAC and encryption.

Course Description:

This hybrid course covers unstructured and NoSQL databases alongside key analytical paradigms. Students explore document, key-value, wide-column, graph, and vector data models and understand their relationship to OLTP, OLAP, data warehouses, and data lakes. Core and advanced MongoDB operations — CRUD, cursors, and aggregation — are studied before advancing to indexing, WiredTiger tuning, GridFS, and MongoDB Atlas. Redis and Neo4j are introduced as representative NoSQL systems, followed by Apache Kafka for real-time streaming and Atlas Vector Search for AI-driven retrieval. Advanced features such as MongoDB's MapReduce and integration capabilities with other database systems are also explored. Each unit embeds three lab sessions (one focused topic per lab hour) that directly practise the theory covered in the same unit.

Course Content:

Units	**Lecture Hours**	**Lab Hours**
Unit-I: INTRODUCTION TO UNSTRUCTURED DATABASES AND DATA MODELS Overview of unstructured data and its challenges. Comparison with structured and semi-structured DBMS. CAP theorem, BASE vs ACID. Data models: Document-based (JSON, BSON), Key-value, Wide-column, Graph, Vector. BSON types: ObjectId, nested documents, arrays. Choosing the appropriate data model for a given use case. **Practical:** Lab 1: Classify sample datasets by NoSQL model type; compare JSON vs BSON structures; sketch a document schema for a given real-world problem. Lab 2: Install MongoDB; connect via mongosh and Compass; create a free MongoDB Atlas cluster; import a sample JSON dataset using mongoimport.	**6**	**3**

Lab 3: Explore BSON types in Compass; insert documents with nested objects and arrays; query using dot-notation on embedded fields.		
Unit-II: DATA INGESTION, EXTRACTION AND ANALYTICAL PARADIGMS Data ingestion techniques: batch vs streaming, ETL vs ELT. mongoimport, mongoexport, and bulk Python inserts. Text extraction, tokenization, and Named Entity Recognition (NER). MongoDB Query Language (MQL): projection, sort, limit, skip, and query operators. OLTP vs OLAP: characteristics and workload patterns. Data Warehouse: star schema, snowflake schema, dimensional modelling. Data Lake: storage zones (raw, curated, consumption), schema-on-read vs schema-on-write. When to use OLTP, OLAP, Data Warehouse, or Data Lake. **Practical:** Lab 4: Load a CSV dataset via mongoimport; write a Python script to bulk-insert documents; validate the imported schema in Compass. Lab 5: Write MQL queries using $eq, $gt, $in, $and, $or, $regex; practice projection, sort, limit, and skip on the ingested dataset. Lab 6: Model the same dataset as a normalised OLTP schema and as a star-schema OLAP structure; document trade-offs between the two designs.	**6**	**3**
Unit-III: CORE MONGODB OPERATIONS: CRUD, CURSORS AND AGGREGATION MongoDB architecture: mongod, replica sets, connection pooling. CRUD in depth: insertOne/insertMany, updateOne/updateMany/upsert, deleteOne/deleteMany, bulk writes. Projection, limit, sort. Cursors: lifecycle, methods (next, hasNext, toArray, forEach), batch size, cursor.explain(), cursor.hint(). Aggregation Framework: $match, $group, $project, $sort, $unwind, $lookup, $facet, $addFields; accumulators. Relationships: embedded vs referenced documents, $lookup joins. Atomic operations and multi-document ACID transactions. **Practical:**	**7**	**3**

Lab 7: Perform updateOne with $set/$inc/$push, upsert operations, and a bulk write; verify results in Compass. Lab 8: Iterate a cursor with forEach and toArray; compare cursor.explain() output for a collection scan vs an indexed query; adjust batch size. Lab 9: Build a pipeline using $match, $group, and $lookup across two collections; use $unwind and $addFields; view output in Compass Charts.		
Unit-IV: INDEXING, WIREDTIGER, ADVANCED MONGODB AND ATLAS Indexing: single-field, compound, text, TTL, sparse, wildcard; index management, covered queries, and index limitations. WiredTiger Storage Engine: MVCC, compression codecs (snappy, zlib, zstd), cache management, checkpointing, journaling. Text search and regular expressions. GridFS: fs.files and fs.chunks; uploading and retrieving binary files. MongoDB Atlas: cluster setup, Atlas Search (full-text), Atlas Vector Search ($vectorSearch), Atlas Triggers, Atlas Charts. Replication: replica set roles, read preferences, write concerns. Sharding: shard keys, balancer. MapReduce. MongoDB-Python (Flask) CRUD via PyMongo. **Practical:** Lab 10: Create single-field, compound, text, and TTL indexes; run explain() before and after each; identify the most efficient index for a given query. Lab 11: Adjust WiredTiger cache size; benchmark snappy vs zlib compression; inspect db.serverStatus() stats. Upload a file to GridFS and retrieve it via PyMongo. Lab 12: Enable Atlas Search and run a full-text query; build a CRUD REST API with Flask and PyMongo connected to Atlas; test all endpoints in Postman.	**6**	**3**
Unit-V: NoSQL ECOSYSTEM, KAFKA STREAMING, VECTOR SEARCH AND BACKUP Redis: core data structures (strings, hashes, sorted sets), persistence (RDB/AOF), pub/sub, primary use cases. Neo4j: graph model (nodes, relationships, labels), Cypher queries (MATCH, CREATE, WHERE, RETURN), use cases. Apache Kafka: brokers, topics, partitions, producers, consumers. MongoDB Kafka Sink Connector; Change Data Capture via Change Streams. Vector databases: embeddings, cosine similarity, Atlas Vector Search ($vectorSearch). Retrieval-Augmented Generation (RAG): chunking, retrieval, LLM integration. Backup and recovery: mongodump,	**5**	**3**

mongorestore, Atlas point-in-time snapshots. MongoDB in Production: RBAC, encryption at rest, audit logging, Atlas performance advisor. **Practical:** Lab 13: Redis — insert and query strings, hashes, and sorted sets; build a leaderboard. Neo4j — load a small graph dataset; run MATCH and shortest-path Cypher queries. Lab 14: Set up Kafka with Docker Compose; write a Python producer sending JSON events; configure MongoDB Kafka Sink Connector; verify documents inserted into Atlas. Lab 15: Generate sentence embeddings; run $vectorSearch on Atlas; compare semantic vs keyword results. Perform mongodump, restore to a new namespace, and configure a custom RBAC role.		