Student NO: 230072BSIT

Technical Report : AIU SmartCampus Polyglot Database Platform

Executive summary

This project delivers a Docker Compose–based polyglot persistence environment for an AIU SmartCampus / URMS (University Records Management System) extension. The stack combines MySQL 8.0 (relational core), MongoDB 7.0 (document profiles/catalogs), ClickHouse 24.8 (time-series analytics), and Neo4j 5.22 (graph relationships), each paired with a lightweight admin UI: Adminer (MySQL), Mongo Express (MongoDB), Tabix (ClickHouse), and Neo4j Browser (Neo4j). The goal is to demonstrate real-world data modeling, indexing, analytics, ETL, and cross-database trade-offs in a clean, reproducible environment.

Architecture and deployment model

The platform is orchestrated through a single docker-compose.yml file with persistent named volumes for each datastore (db_data, mongo_data, clickhouse_data, neo4j_data). Port mappings are parameterized via .env, enabling consistent local development while keeping the Compose file portable. Example mappings include MySQL on 3307, Adminer on 8082, MongoDB on 27018, Mongo Express on 8084, ClickHouse HTTP on 8125, Tabix on 8087, and Neo4j HTTP on 7475. This environment supports rapid spin-up/down with stateful persistence, which is ideal for coursework demos, portfolio evidence, and repeatable testing.

Relational layer: MySQL (BCNF-focused URMS extension)

The MySQL schema establishes a normalized academic core: departments, rooms, lecturers, students, courses, enrollments, and student_course_performance. Design choices emphasize BCNF-friendly structure, referential integrity, and data quality through FOREIGN KEY constraints and CHECK constraints (e.g., student status enumerations, credit bounds, sensor

types). A notable technique is the final_score stored generated column in student_course_performance, reducing update anomalies by deriving totals from component scores.

The URMS extension adds:

1. materials (course content metadata with file constraints)
2. activity_logs (LMS usage events with strict activity-type validation)
3. sensors and sensor_readings (smart classroom instrumentation; uniqueness prevents duplicate readings per sensor timestamp)

Performance engineering: indexing and views (MySQL A2)

The project includes a dedicated indexing script that is idempotent (drop-if-exists via information_schema + dynamic SQL). Index strategy targets common access patterns:

1. activity_logs: single-column indexes plus a required composite (student_id, timestamp) and a covering index to accelerate reporting queries.
2. sensor_readings: time and status-time indexes for windowed monitoring and reliability analysis.
3. materials: type and upload-date indexes for course content search/filter.
4. students: status index for dashboards.

A view vw_student_activity_agg materializes a “student + activity summary” shape by joining student records to aggregated activity metrics (count, total seconds, last activity), simplifying consumption by apps and reports.

Server-side programming objects (MySQL A3)

To demonstrate database programmability and lifecycle management, the project implements:

1. An archive table activity_logs_archive for long-term retention
2. A UDF udf_CalculateStudentEngagement(student_id, days) producing a 0–100 engagement score based on frequency and minutes/day
3. A stored procedure sp_StudentAnalytics(student_id) returning student profile + enrollments + performance + engagement score
4. A trigger trg_archive_activity that archives qualifying records before delete when older than one year

This set shows how to push repeatable analytics and governance logic closer to the data.

Document layer: MongoDB (B1)

MongoDB models "shape-flexible" entities:

- student_profiles (embedded preferences, extracurriculars, enrollments, activity summary)
- course_catalogs (nested materials arrays + metadata)

The script creates JSON Schema validators (tuned to accept numeric types commonly produced by scripts) and indexes for query patterns (department/status, last activity timestamp, plus full-text search over course descriptions and tags). It also demonstrates CRUD with $inc, $set, $push, and an aggregation pipeline using $match, $group, and $project.

Time-series layer: ClickHouse (B2)

ClickHouse provides high-throughput storage for sensor telemetry with:

1. sensor_readings_raw partitioned by date and ordered by (sensor_id, ts)
2. A TTL policy deleting raw data after 365 days

3. A materialized view that continuously down-samples into an hourly AggregatingMergeTree table, enabling efficient rollups (avg/min/max/count states)

A Python ingestor generates realistic minute-level readings and relies on the MV to populate hourly aggregates automatically.

Graph layer: Neo4j (B3)

Neo4j models academic structure, clubs, enrollments, and material access. The Cypher script builds constraints, indexes (including a full-text index), seeds example entities (including Clive Aono), and runs advanced queries: prerequisite path finding, community grouping, peer-based course recommendations, influence proxy via shared relationships, and most-viewed materials.

ETL and distributed transaction reasoning (C1/C2)

A robust Python ETL (pipeline/C1_etl.py) simulates streaming ingestion from a file inbox into all four systems, using checkpointing, BOM-safe parsing, “complete-line only” incremental reads, and bad-row quarantine. A companion analysis discusses cross-database consistency, highlighting lost updates, 2PC-style coordination, and saga/compensation trade-offs showing you understand the operational reality of polyglot systems.