# NaturalDB Midterm Progress Report

**DSCI 551: Data Management**

**Project:** A Natural-Language-Driven NoSQL Database System and its Application in E-commerce

**GitHub Repo:** https://github.com/felix-wang-0307/NaturalDB

**Date:** October 17, 2025

---

## Project Overview

NaturalDB is a custom NoSQL database system designed to store and query JSON data while supporting natural language interaction. The project follows a 4-layer architecture: Storage System, JSON Parser & Query Engine, Natural Language Query Interface, and E-commerce Front-end Application.

---

## Milestone Status Summary

All planned milestones for the midterm evaluation have been successfully completed, with both Layer 1 (Storage System) and Layer 2 (JSON Parser & Query Engine) fully implemented and functional.

### Week 5 Milestone: Layer 1 - Storage System

**Status:** Successfully implemented and functional

**Completed Components:**

- **File-based key-value store architecture**
  - Folders map to tables (e.g., `Products/` → Products collection)
  - Files map to records (e.g., `Products/1.json` → product with ID 1)
  - Local data directory structure for efficient organization
- **Core Storage Classes:**
  - `Storage`: Base storage class with static path management
  - `DatabaseStorage`: Handles database-level operations and metadata
  - `TableStorage`: Manages table-level operations and record handling
- **CRUD Operations Implementation:**
  - Create: `create_user()`, `create_database()`, `create_table()`
  - Read: `load_record()`, `load_all_records()`, `list_records()`
  - Update: `save_record()` (overwrites existing)
  - Delete: `delete_user()`, `delete_database()`, `delete_record()`
- **Metadata Management:**
  - Database metadata tracking (`metadata.json` files)

    - Table metadata with indexes and keys support
    - Automatic directory structure creation

- **File Locking System:**

    - Thread-safe read-write locks (`RWLock` class)
    - Lock manager for concurrent access control
    - Prevents data corruption during simultaneous operations

---

## Week 7 Milestone: Layer 2 - JSON Parser and Query Engine

**Completed Components:**

### 2.1 Custom JSON Parser (`json_parser.py`):

- **Full JSON parsing without external libraries** (as required)
- Supports all JSON data types: strings, numbers, objects, arrays, booleans, null
- **Advanced features:**
    - Unicode escape sequence handling (`\uXXXX`)
    - String escaping for special characters
    - Proper error handling with descriptive messages
    - File and string parsing capabilities
- **JSON serialization:** Python objects → JSON strings with optional indentation
- **Robust error handling:** Custom `JSONParserError` exceptions

### 2.2 Query Engine (`query_engine.py`):

- **High-level database interface** with intuitive API
- **Complete CRUD operations:**
    - `insert()`, `find_by_id()`, `find_all()`, `update()`, `delete()`
- **Advanced query capabilities:**
    - Field-based filtering with multiple operators (eq, ne, gt, lt, gte, lte, contains)
    - Multi-field projection with nested field support (e.g., `specs.storage`)
    - Sorting with ascending/descending options
    - Grouping and aggregation (count, sum, avg, min, max)
    - Join operations (inner join, left join)

### 2.3 Query Operations (`operations.py`):

- **Filtering system:**
    - Condition-based filtering with lambda functions
    - Field-specific filtering with comparison operators
    - Nested field access using dot notation
- **Aggregation functions:**
    - Mathematical operations: sum, average, min, max
    - Counting and grouping capabilities
- **Join operations:**
    - Inner joins and left joins between tables

    - Configurable field mapping and table aliases
- **Sorting and limiting:**
    - Multi-field sorting with null value handling
    - Pagination support with offset and limit

**2.4 Data Import/Export:**

- **JSON file import:** Direct file-to-table import functionality
- **JSON file export:** Table-to-file export with formatting options
- **Batch operations:** Support for importing arrays of records

---

## Detailed Project Member Contributions

This is an individual project; all contributions were made by the primary developer.

### Felix Wang (felix-wang-0307)

- Setup project architecture and GitHub repository
- Complete design and implementation of **Layer 1 (Storage System)** and **Layer 2 (JSON Parser & Query Engine)**
- Complete test scripts for demonstrating functionality
- Documentation and reporting for project milestones

---

## Next Steps

- Begin implementation of **Layer 3: LLM-powered Natural Language Query Interface**
- Research and integrate suitable LLM services for natural language processing
- Develop Flask REST API for database operations
- Plan and start development of **Layer 4: E-commerce Front-end Application**
- Deploy back-end and front-end applications using Vercel