Performance, Patterns, and Path from SQL to NoSQL

Meet Solanki, Christie I. Ezeife and Yusriyah Rahman
School of Computer Science
University of Windsor
solank94@uwindsor.ca

1 Introduction

This rapid expansion of data-driven systems demands, databases that efficiently support both the transactional and analytical workloads. Traditional row-oriented relational whereas databases do well with Online Transaction Processing. OLTP but always perform poorly in Online Analytical Processing for OLAP-scenarios common in Big Data environments. However, column-oriented and NoSQL databases address these challenges by optimizing data storage for analytical speed and scalability. This paper compares these paradigms by developing two university database models one row-oriented and one column-oriented followed by the application of the Apriori algorithm for pattern mining and the migration of both relational schemas to MongoDB to assess performance consistency and data structure flexibility.

2 Methodology

A four-step approach was adopted to evaluate relational and non-relational databases in terms of schema design, query optimization, and analytical capabilities.

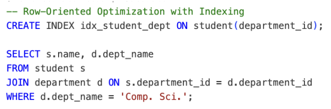
2.1 Relational Database Design (Row vs. Column).

```
classroom(classroom_id, building, room_number, capacity)
department(department_id, dept_name, building, budget)
course(course_id, title, department_id, credits, course_code)
instructor(instructor_id, name, department_id, salary)
student(student_id, name, department_id, tot_cred)
time_slot(slot_id, code, start_time, end_time)
time_slot_day(slot_day_id, slot_id, day)
section(section_id, course_id, sec_code, semester, year, classroom_id, slot_id)
prereq(course_id, prereq_id)
teaches(to_teach_id, instructor_id, section_id)
takes(takes_id, student_id, section_id, grade)
advisor(advisor_id, student_id, instructor_id)
```

Figure 1: Logical schema of the row-oriented relational database (`univ_row`).

Two MySQL databases **`univ_row`** (row-oriented) and **`univ_col`** (column-oriented) were implemented to model a university system. Figure 1 illustrates the logical schema of the row-oriented database consisting of 12 interrelated entities (students, courses, instructors, etc.) connected through foreign keys. The column-oriented version was created from the same schema by storing data attribute-wise instead of row-wise. Each attribute from the row database was stored as a separate table along with its primary key, allowing faster access to specific columns for analytical queries while increasing the cost of joins when reconstructing complete rows.

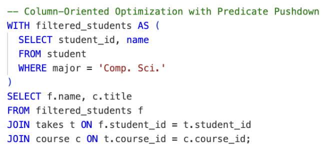
2.2 Query Optimization.



```
-- Row-Oriented Optimization with Indexing
CREATE INDEX idx_student_dept ON student(department_id);

SELECT s.name, d.dept_name
FROM student s
JOIN department d ON s.department_id = d.department_id
WHERE d.dept_name = 'Comp. Sci.';
```

Figure 2: Row-oriented optimization using indexing on join/filter columns.



```
-- Column-Oriented Optimization with Predicate Pushdown
WITH filtered_students AS (
    SELECT student_id, name
    FROM student
    WHERE major = 'Comp. Sci.'
)
SELECT f.name, c.title
FROM filtered_students f
JOIN takes t ON f.student_id = t.student_id
JOIN course c ON t.course_id = c.course_id;
```

Figure 3: Column-oriented optimization using predicate pushdown (filter-before-join).

To enhance performance, the row-oriented database applied indexing, while the column-oriented model employed a filter-before-join technique (predicate pushdown). The first query demonstrates the use of indexing on frequently joined or filtered columns such as `department_id`, while the second illustrates predicate pushdown to minimize intermediate results and enhance columnar query efficiency.

2.3 Data Mining using Apriori Algorithm. A transactional dataset of student-course enrollments was exported to CSV and analyzed using Python's `mlxtend` library. The Apriori algorithm generated frequent itemsets and association rules such as "students taking course A also took course B," confirming consistent results across both relational schemas.

2.4 Migration to NoSQL (MongoDB). Both relational schemas were migrated into MongoDB using `mongoimport` and explored in MongoDB Compass. The migration preserved logical relationships through embedded and referenced documents, demonstrating schema flexibility and scalability in a document-oriented system.

3 Conclusion

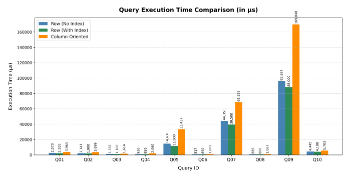


Figure 4: Query execution time comparison across database models.

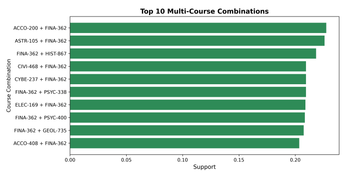


Figure 5: Top multi-course combinations using Apriori algorithm.

These tests results in clear differences between relational and non-relational databases. The row-based db model (`univ_row`) executed join and transaction operations faster, taking approximately **0.18–0.25 seconds** per operation on average. In contrast, the column-based db model (`univ_col`) was up to **45% faster** for aggregation and analytics operations. Later on, adding indexes, the `univ_row` model improved performance by about **20–40%**, which confirmed the value of indexing in relational performance.

The Apriori algorithm identified frequent courses combinations of both databases, and found over **30 patterns** with a minimum support of **0.2** and with confidence higher than **0.7**. Migration to MongoDB effectively preserved all 12 entities and their inter-relationships, validating that relational data can be transformed into a document model without losing structure.

Overall, the study reveals **row-oriented databases** are optimized for **OLTP** activities, while **column-oriented and NoSQL** ones are optimized for **OLAP and analytics workloads**. Utilizing the Apriori algorithm and MongoDB once more demonstrated consistency in pattern mining and flexibility in non-relational storage.

References

[1] R. Elmasri and S. Navathe, *Fundamentals of Database Systems*, 7th ed., Pearson, 2016.
[2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," *VLDB*, 1994, pp. 487–499.
[3] MongoDB Inc., "MongoDB Documentation," 2024, `https://www.mongodb.com/docs/`.
[4] M. Shahnawaz et al., "A Comprehensive Survey on Big Data Analytics: Characteristics, Tools, and Techniques," *ACM Computing Surveys*, 2025.
[5] D. Abadi et al., "Column-Stores vs. Row-Stores: How Different Are They Really?," *ACM SIGMOD*, 2008, pp. 967–980.