Archives available at journals.mriindia.com

International Journal on Advanced Computer Theory and Engineering

ISSN: 2319 - 2526

Volume 15 Issue 01s, 2026

Next-Generation Library Information Systems: Evaluating Native Multi-Model Database Technology

[1]Monali M. Chaudhari, [2]Snehalata B. Shirude

[1]School of Information Technology, Indira University, Pune, Maharashtra-411033

[1, 2]School of Computer Sciences, Kaviyatri Bahinabai Chaudhari North Maharashtra University, Jalgaon, Maharashtra-425001

Email: [1]waykole.monali@gmail.com, [2]snehalata.shirude@gmail.com

Peer Review Information

Submission: 08 Dec 2025

Revision: 25 Dec 2025

Acceptance: 10 Jan 2026

Keywords

ArangoDB, PostgreSQL, CouchDB, Neo4j, Hybrid multi-model, NLP, native multi-model

Abstract

Multi-model databases handles the structured, semi-structured, and highly connected data of various applications such as Healthcare, Library Information System and many more. This research paper put focus on two multi-model databases, one is native multi-model database, ArangoDB, and a Hybrid multi-model database integrating PostgreSQL, Couch DB, and Neo4j. The objective of this paper is to measure the performance of these multi-model using various evaluation criteria as such as execution time, throughput, indexing efficiency, and latency. Also, further it highlights on query designing and data retrieval efficiency showing better approach for library management environment. A real-world college library system which handle big data workloads was chosen to evaluate the performance of these multi-models. The results shows that hybrid multi-model can be adapted in cases where a stronger transactional reliability is required. In contrast, ArangoDB, performs more efficiently in cross-model queries, especially a single AQL query unified document, graph, and relational data retrieval, minimizing query orchestration and communication overhead. ArangoDB performs 47% better in execution time and 42% in throughput than Hybrid Model. Natural Language Processing (NLP) was used for query translation that enabled users to submit queries in plain English, which automatically transformed into structured database commands, improving accessibility and user experience. This research will help developers and researchers to design a better multi-model which is efficient for providing faster and more organized academic resources to students and faculties.

Introduction

With the rapid development of digital information techniques in Internet technology, IOT and cloud computing, data has increased in an unprecedented scale in different fields. [1] This exponential growth of information and the diverse data in digital libraries have given rise to multi-model databases capable of handling multiple data models within a unified framework. PostgreSQL is one of most popular traditional relational databases, supports strong ACID properties and robust transaction processing but lacks with unstructured or highly connected data. [2] In contrast, Couch DB and Neo4j are NoSQL databases that provides flexibility and scalability especially for document and graph data. However, these systems, lack native support for integrating multiple data representations under one query processing mechanism, making hybrid data management more complex.

To address these challenges, the concept of multi-model databases came up with support for multiple data models relational, document, graph, key-value, etc within a single backend engine. [3] Hybrid multi-model databases that integrate PostgreSQL, Couch DB, and Neo4j databases have specifically been employed to give users the benefits of each data model under one platform. For instance, PostgreSQL in Library Information System is more suitable for structured records and transactional data; Couch DB supports storage of textual feedback and supplier documents; and Neo4j efficiently manages traversal capabilities across complex user and material relationships. Individually the performance of these databases is optimized to great extent, but its integration introduces the overhead of query distribution, and result aggregation. Further inter-database communication introduces latency and complexity in real-time applications. [4] But in contrast, ArangoDB is beneficial in reducing communication overhead and allowing tighter integration between data models thus eliminating the need for inter-database query parsing and cross-database joins.

Data in library information systems, includes structured metadata such as (purchase orders, payments, bill, inventory_stock), semi-structured content (student_profiles, teacher_profiles, penalty_logs, borrow_records), and graph-based relationships (material_relationships, searchlib, review, feedback), thus evaluating these architectures becomes crucial. In this research, we designed various 25 multi-model queries representing realistic operations in college library. Queries were benchmarked using evaluation criteria such as execution time, latency, throughput, and indexing strategies across both the hybrid multi-model and ArangoDB, with metrics including.

To bridge the gap between users and applications using multi-model databases, Natural Language Processing (NLP) provides a better solution for translation of queries into simple English language. [5] In library information systems, users often lack expertise in query languages such as SQL, Cypher, or JSON-based selectors, etc where integration of NLP with Large Language Models (LLMs), can eliminate the need for technical training, enhances accessibility, and improves user experience by enabling librarians, faculty, and students to interact with the system using conversational language.

Literature Review

1. SQL Vs Nosql And The Rise Of Multimodel Databases

The limitations of conventional relational databases in managing heterogeneous datasets have been widely discussed in prior research. Harrington & Christman (2019)[11] put focus on a comparative analysis of relational databases and NoSQL systems, along with their advantages and limitations in handling library data by describing features of NoSQL such as schema-less databases, scalability, and high availability, can be a better choice for handling growing library datasets. Further research could involve testing NoSQL systems with library datasets to evaluate performance improvements. No practical implementation or experimental validation was demonstrated to show the benefits of NoSQL in a real library environment.

In 2021, Kanchan, Kaur, and Apoorva [8] carried out study of relational and NoSQL database systems showing databases could perform operations like insert, search, and grow. It found that SQL databases were performing well on structured queries, while NoSQL databases are better in flexibility and scalability . On the other hand, they came up with theoretical ideas that the results could be better in future if the investigation integrate SQL and NOSQL for distributed applications and hybrid database structures.

A comparative study of the assessment of various research papers were examined by Corovčák and Koupil (2025) [7] and identified that there are six critical characteristics of SQL and NoSQL systems. Also review further states that transactional consistency is better in SQL whereas NoSQL variations are better at scaling and being flexible with schemas. Additional research suggests that cost of integration of SQL and NoSQL increases along with its complexity but could offer adaptability and uniformity in the development of hybrid systems.

Lu and Holubová (2019) [3] studied that there is necessity in managing various data forms, including documents, graphs, relational tables, and key-value pairs, within a single unified system. The paper details about property-graph query extensions to SQL and SQL++ and AQL query design, emphasizing the significance of unified query languages. Multi-model databases minimizes developer effort by facilitating cross-model joins, graph traversals, and JSON operations within a singular query plan.

2. Hybrid Database Architectures And Polyglot Persistence

Research on polyglot persistence has become more popular as data systems have moved toward microservices and domain-driven architecture, the practice of mixing various specialized databases. A comparison of various

features such as data retrieval complexity, cross-model integration effort, query execution time, , storage overhead, and schema flexibility were examined on the multi-model databases and the polyglot persistence by Singh, N. (2020) [9]. The results indicate that in diverse datasets multi-model databases shows better performance then polyglot persistence. Multi-model DBMSs diminishing latency, enhancing consistency, and streamlining application code across models by eliminating the burdens of inter-database communication, serialization, and transformation, thus utilize a singular storage backend more efficiently.

Sandell et al. (2024) [10] demonstrated running connected-data queries along with experimental benchmarking on all three systems. For querying connected data, Neo4j performs better than ArangoDB. Neo4j surpasses both MySQL and ArangoDB based on their experiments, also ArangoDB multimodel benefits, did not show performance for traversal connected queries in their benchmark. The article also recommends that multi-model systems such as ArangoDB can be used as additional research on optimizing, for connected-data searches, either via enhanced indexing or query planning.

Belgundi et al. (2023) [6] analyze the capabilities, performance characteristics, and architectural benefits of ArangoDB as a native multimodel database that consolidates document, key–value, and graph models into a unified engine. The paper emphasizes that the database's cohesive architecture reduces architectural complexity in modern data-intensive applications, assessed from both theoretical and practical perspectives.

Polyglot Database Design Method (PDDM) is a development of hybrid systems which Polyglot Database Design Method (PDDM)outlines a procedure systematically is developed by Zdepski et al. (2020) [12]. The authors offers a methodical framework for creating applications established in the Polyglot Database Design Method (PDDM) via extensive study on heterogeneous database systems, that concurrently employ various database technologies. The rising requirement for contemporary systems to integrate relational, document, graph, key-value, and columnar databases into a cohesive design led to the creation of PDDM.

3. Arangodb As A Unified Multimodel Database

To facilitate the management of graph-structured data, Mohamed et al. (2023) [13] introduce an enhanced access-control system that adapts the XACML (eXtensible Access Control Markup Language) framework. Conventional XACML is not capable of developing authorization rules based on graph-specific properties such as node types, relationships, path structures, or graph patterns, as it was predominantly designed for hierarchical or tabular data. This study offers a Graph-XACML modification to enable graph databases and graph-based systems to implement flexible, expressive, and standardized authorization policies, therefore bridging the existing gap.

The purpose of this research is to provide a systematic methodology that can be used to migrate large monolithic relational databases into modern polyglot persistence systems that support multiple models. The authors address a problem that is becoming more and more prevalent, which is that when older systems store heterogeneous, semi-structured, and relational data in a single relational database management system (RDBMS), it can result in scalability challenges, rigid schemas, and performance bottlenecks. The methodological and tool-supported technique that is proposed by the study is intended to serve as a means of dividing monolithic databases into a number of specialized NoSQL and SQL databases.

This 2018 comparison study assesses four prominent graph databases AllegroGraph, ArangoDB, Neo4j, and OrientDB aimed at examining their performance, data models, query functionalities, and architectural features. The authors intend to assist developers and researchers in choosing a suitable graph DBMS according to workload specifications, scalability demands, and data model adaptability [15].

Recent studies, such as “Analysis of Native Multi-Model Database Using ArangoDB” (2023), examined ArangoDB's capacity to consolidate several data models under a single framework. The authors discovered that ArangoDB’s native query execution engine, utilizing AQL’s graph joins and JSON management, surpassed polyglot architectures in mixed workloads. ArangoDB performs better on integration of relational and document databases whereas Neo4j excels in pure graph traversal speed in comparative analysis of Neo4j vs ArangoDB.

To decrease latency in ArangoDB we can achieve that through optimizing query planner, by minimizing external joins, and by unifying indexing. On the other hand, PostgreSQL remains the dominant force when it comes to high-frequency online transaction processing (OLTP) transactions, as ArangoDB's transaction model, although it adheres to the ACID standard, may be less efficient in this particular application.

4. NLP For Query Translation And User Interaction

Integrating human language, multimodal database querying and schema design have made advancements in research. To use these advance technologies there are two approaches (a) natural langauge tools such as DBTagger, xDBTagger that derive schema or model intent from natural language to facilitate polyglot design and migration, and (b) systems that convert user queries to streamline access across diverse backends. The developer efforts are minimize by using these methodologies at both run-time and design-time.

The simple language was transformed into relational schema in DBTagger (2021) which utilizes traditional NLP pipelines to discern potential entities, attributes, and connections. It demonstrated that structured information beneficial for conceptual modeling can be extracted with considerable accuracy from formal requirement documents, expediting the transition from conceptual to logical design.

To classify textual artifacts as more suitable for document, graph, or relational models , xDBTagger (2022) [16] enhances this approach within the multimodel domain by employing embedded contexts and transformer components ; hence, it aids in model selection and element extraction. These systems collectively execute PDDM-style judgments concerning the suitable data model for each object and can be integrated into migration operations necessitating the segmentation of a monolithic schema into multiple storage options.

Text2Cypher (2025) [17] illustrates the feasibility of semantic parsing for graph databases, showing that transformer-based NL→Cypher models can accurately convert user inquiries into executable graph queries for Neo4j, particularly for simple and moderately complicated intents. This enhances accessibility for non-expert users and is suitable for interactive analytics on knowledge graphs and social networks.

MetaSQL (2024) [18] addresses the complementary issue of cross-model query abstraction by introducing a meta-language and middleware that disaggregates a singular meta-query into subqueries for relational, document, and graph databases. MetaSQL seeks to maintain expressive capability (joins, nested access, traversals) while directing computation to the most suitable engine, hence alleviating developer cognitive burden in polyglot environments.

There are a lot of common methodological patterns in all of these works. For example, they all use transformer embeddings or hybrid pipelines for robust natural language processing (NLP), mapping heuristics or alignment layers to turn natural language entities or relationships into schema artifacts, and middleware/optimizer components to manage decomposition and routing for cross-store queries.

5. Machine Learning And Analytics In Library Systems

The use machine learning (ML) and data analytics is increasing day-to-day library systems, which leads to advancement of digital collections and rise in user information requirements. Initial implementations concentrated on recommendation algorithms, with Qin, Chen, and Wang (2020) [20] illustrating that improvement of user engagement in academic libraries is possible due to machine learning-based recommender systems. Various ML techniques are included in this research such as collaborative filtering and content-based models to show enhanced borrowing behaviours, deliver personalized reading recommendations, service engagement. This study highlights that the personal learning experience and user satisfaction can be increased using machine learning.

Machine learning plays important role beyond personalization, as well as in resource optimization and collection management. To forecast resource demand as well as circulation trends, predictive analytics have been used. Ikwuanusi et al. (2021) [19] applied machine learning algorithms such as random forest and support vector regression models to forecast usage trends and acquisition requirements, also libraries' efficiency enhances in budgetary and procurement decisions. The results show that machine learning can enhance library administration, availability of high-demand resources can be increased.

Simultaneously, progress in natural language processing (NLP) has broadened prospects for the automation of technical services. Sarode et al. (2022) [21] demonstrated BERT-based NLP techniques for automation of metadata extraction and cataloguing. Their methods illustrated how intelligent automation may boost essential backend activities in digital libraries and significantly enhanced metadata quality and reduced human processing time.

Analytical insights into user behaviour have ultimately emerged as crucial tools for improving services. In Ansari et al. (2021) [22], the spatial usage of library resources, user visitor patterns, and borrowing behaviours are examined and their results shown improvement in library configurations, timetabling, and collection organization. The utilization of machine learning technologies has encouraged a growing trend towards evidence-based library management techniques.

These studies collectively demonstrate that services like recommendations and chatbots can be improved by machine learning and analytics, also enhance essential library processes such as cataloguing, resource planning, and behaviour analysis. Machine learning is becoming popular in libraries, but still issues with ethical data management, cross-library model generalization, and the combination of multimodal datasets (text, usage logs, and location data). The observed limitations provide possibilities for further research on unified, intelligent library management systems that employ machine learning across all service levels.

6. Research Gap

Few research shows the comparison of relational, document, and graph databases with native multimodel databases like ArangoDB. To fill this research gap, we developed a hybrid architecture made up of PostgreSQL, CouchDB, and Neo4j and compared it with ArangoDB database. It discusses trade-offs between integration complexity and query performance and conducts tests across CRUD, graph, and cross-model queries to assess performance, scalability, and consistency. Hybrid model integrated with NLP-driven query translation is still lacking. This study consists of comparison of both multi-model databases for real-world library dataset, evaluating performance metrics like execution time, indexing efficiency, throughput, and latency, and also addressing usability improvements from NLP integration.

System Architecture & Methodology

1. System Architecture For Arangodb And Hybrid Multi-Model Database

The System architecture for ArangoDB and Hybrid Multi-model database consists of 4 layers as Presentation Layer, Application Layer, Data Access Layer, Data Storage Layer. The Presentation layer encompasses Library website where it consists of User interface and the APIs. The students, staff and librarians interact with the user interface and get the services through the APIs. The Application layer consists of the microservices, business logic and the middleware which is developed in Python/Django framework. The user can request through Data Access Layer where the services required can be written in form of simple English language which is further translated in AQL or SQL or cypher language. The Data Storage layer actually consists of the ArangoDB or Hybrid Model where the documents, records or graphs are stored.

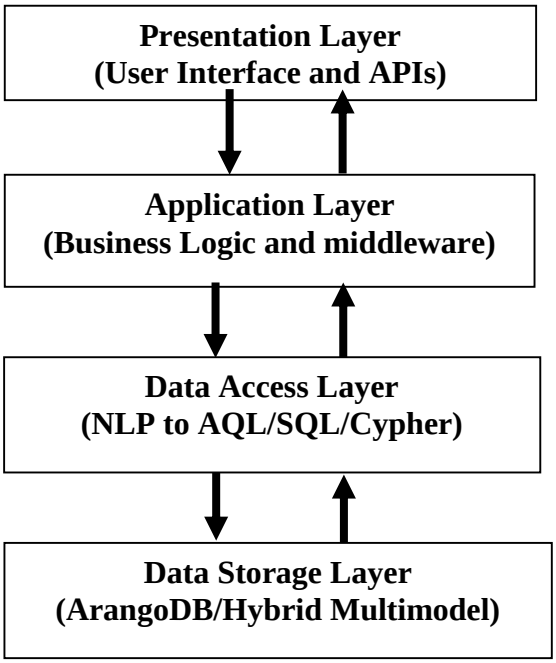

Figure 1: System Architecture for ArangoDB and Hybrid Multi-Model Database

2. Data Model and Datasets for Library Information System

A real-world digital library system data was collected which consists of following modules and the database model it implement in ArangoDB and Hybrid Multi-model Database. The dataset contained approximately 1.5 million records distributed across the three models

Table 1: Data Models For Library Information System

Module	ArangoDB	Hybrid
User (Student and Teacher) Management	Document + Graph	(RDBMS for auth + Document for profiles) - PostgreSQL + Couch DB
Supplier Module	Document	(Document) - Couch DB
Book Transaction Module	Document + Graph	(RDBMS) - PostgreSQL
Finance and Inventory module	Document	(RDBMS) - PostgreSQL
Teaching Material Module	Document + Graph	(Document + Graph) - Couch DB + Neo4j
Question paper and syllabus module	Document	(Document) - Couch DB
Journal Module	Document	(Document) - Couch DB
Tie-ups Module	Graph	(Graph) - Neo4j
Search Lib Module	Graph	(Graph) - Neo4j
Review Module	Graph	(Graph) - Neo4j
Feedback Module	Graph	(Graph) - Neo4j

3. Query Formulation

We designed several multimodel queries reflecting typical information retrieval and analytical tasks in the library systems:

- Relational Aggregations (e.g., borrowing counts, department-level trends).
- Document Retrievals (e.g., supplier feedback, journal metadata).
- Graph Traversals (e.g., student friendship networks, faculty–institution relationships).
- Cross-Model Queries combining two or more database paradigms.

The user writes the natural language query as "**Get materials provided by suppliers and check which users have liked them**" and then decomposed into two paradigms as: -

a. Hybrid Model Queries distributed across PostgreSQL, CouchDB, and Neo4j.

- CouchDB → Get materials and supplier documents.
- Neo4j → MATCH (u:User)-[:LIKED]->(m:Material).

b. ArangoDB AQL Queries integrating all models within a single execution.

```
FOR m IN Materials
 FOR u IN Users
 FILTER m._key IN u.liked_materials
 RETURN {material: m.title, liked_by: u.name}
```

This allowed us to test both usability and performance (execution time).

4. Indexing methods

Indexes were carefully designed to optimize query execution across systems:

1. PostgreSQL uses B-tree indexes and materialized views as an index.
2. CouchDB uses JSON Mango indexes for structured attributes and for keyword searches it uses MapReduce functions as a part of indexing.
3. Neo4j has Label and property indexes and adjacency list indexing for graph traversals.
4. ArangoDB has Hash and skiplist indexes for structured queries as well as inverted indexes for text search and edge indexes for graph traversals.

This ensured that neither system was disadvantaged by poor indexing design.

Implementation

The proposed library information system was developed and tested on a system with the following configuration: an Intel Core i5 10th Generation processor, 8 GB of RAM, 1 TB storage, and a high-speed internet connection. The database consists of approximately 20 million records, amounting to a total size of around 200 GB. The backend of the system was implemented using Python (version 3.12.0) and the Django web framework (version 5.2), enabling rapid development and secure data handling. The frontend was created using HTML5 and CSS3, ensuring a responsive and user-friendly interface. Data storage and querying were handled using ArangoDB (version 3.11.8 for Windows 64-bit), a multi-model NoSQL database supporting document, key-value, and graph data models. Development was carried out on a Windows operating system using Visual Studio Code (VS Code) as the primary Integrated Development Environment (IDE). The Python code was executed using the CPython interpreter, the default compiler for Python, ensuring efficient runtime performance and compatibility with Django and ArangoDB integrations.

To develop a web-based application using Django (Python) with ArangoDB and hybrid model as a multimodel backend, the setup involves multiple stages. Initially, essential tools must be installed including Python, Django, and the ArangoDB as well as Postgresql, CouchDB and Neo4j server. Django can be installed via pip, and ArangoDB can be installed either locally or through Docker for containerized environments. The same environment is used for Postgresql, CouchDB and Neo4j. Once installed, a new Django project is created using django-admin, followed by creating an app (e.g., library_app). A connection between Django and ArangoDB is established using the python-arango driver by defining a connector function that authenticates and returns a database instance. Also, connection is set for Postgresql, CouchDB and Neo4j.

The application logic is implemented inside Django views. A function is created that connects to the database and inserts a JSON-like document into the respective collection. This function is mapped to a URL route and invoked via an HTTP request. URL routing is configured in both the app-level and project-level urls.py files. Once the server is started using python manage.py runserver, the endpoint can be tested in the browser or using tools like Postman. Finally, the user have to log in to ArangoDB's web interface (default port 8529), open Postgresql, connect CouchDB and Neo4j to verify that the data is properly stored. This flow enables seamless integration between Django-based frontends and the backend.

Evaluation Metrics

To capture performance differences comprehensively, we employed the following metrics:

1. The average runtime for each query under identical conditions is measured by execution time.
2. Delay before query execution begins is measured by latency, which is particularly useful in hybrid models because of query parsing and network overhead.
3. Number of queries executed per second is measured by throughput in operations/per second under concurrent workloads.
4. To reduce query complexity as well as improving runtime the Effect of selected indexes is used by Indexing Effectiveness.
5. Observations regarding the better performance of each system can be measured by Observational Analysis.

The table below includes execution time, index use, and observations as the performance evaluation of the hybrid and ArangoDB models. For the workloads of academic library systems, this comparison significantly helps to evaluate the efficiency and optimization potential of each database.

Table 2: Comparative Performance Evaluation For Hybrid And Arangodb Model.

Query (Natural Language)	Queries (DB-specific)	Hybrid Time (sec)	ArangoDB Time (sec)	Indexing Used	Observations
Find all materials tagged with keywords users are interested in and enrolled in institutions.	CouchDB → get materials with tags.Neo4j → USER -[:INTERESTED_IN]-> Keyword.PostgreSQL → check enrollment.	0.48	0.22	CouchDB JSON index, Neo4j label index, PostgreSQL B-tree.	ArangoDB faster: joins in AQL avoid cross-DB overhead.
List all students who borrowed books and also submitted feedback.	PostgreSQL → borrow records.CouchDB → feedback docs by student_id.	0.40	0.18	Postgres B-tree on student_id, CouchDB Mango index.	ArangoDB 2 times faster; hybrid suffers network overhead.
Get materials with suppliers and check which users have liked them.	CouchDB → get materials & suppliers.Neo4j → USER -[:LIKED]-> Material.	0.41	0.20	CouchDB MapReduce, Neo4j property index.	ArangoDB wins: doc+graph handled in single query.

List faculty teaching at institutions where students borrowed books.	PostgreSQL → borrowed books + institution.Neo4j → FACULTY -[:TEACHES_AT]-> Institution.	0.50	0.25	Postgres B-tree, Neo4j label index.	ArangoDB faster: native graph+relational join.
Retrieve top 10 most borrowed books per semester.	PostgreSQL → aggregate borrow counts, group by semester.	0.33	0.95	PostgreSQL B-tree + materialized view.	Hybrid outperforms: PostgreSQL optimized for aggregation.
Detect student communities based on borrowing patterns.	Neo4j → project STUDENT -[:BORROWED]-> BOOK, run Louvain clustering.	0.35	1.50	Neo4j adjacency list index.	Hybrid wins: Neo4j clustering better than Arango.
Identify users who borrowed journals and provided feedback.	PostgreSQL → journal borrow records.CouchDB → feedback docs.	0.42	0.21	Postgres B-tree, CouchDB Mango index.	ArangoDB 2 times faster: fewer round-trips.
Find teachers who recommended books that students borrowed.	Neo4j → TEACHER -[:RECOMMENDS]-> BOOK.PostgreSQL → borrow records.	0.45	0.28	Neo4j label index, PostgreSQL B-tree.	Hybrid slightly slower: Arango integrates edges + docs.
Rank suppliers based on most borrowed materials.	CouchDB → supplier-material mapping.PostgreSQL → borrow counts.	0.47	0.30	CouchDB MapReduce, PostgreSQL index.	Hybrid slower due to split aggregation.
Get students who borrowed same book and are friends.	PostgreSQL → borrow records.Neo4j → FRIENDS_WITH graph.	0.44	0.26	PostgreSQL index, Neo4j relationship index.	ArangoDB faster for doc+graph join.
Find average rating for each book.	Neo4j → ratings.PostgreSQL → book IDs.	0.38	0.34	Neo4j property index, PostgreSQL B-tree.	Nearly same, slight edge to hybrid (optimized aggregations).
Detect institutions with highest number of enrolled students.	PostgreSQL → count students grouped by institution.	0.30	0.80	PostgreSQL B-tree.	Hybrid faster: pure SQL aggregation better than Arango.
Find students who borrowed materials outside their enrolled institution.	PostgreSQL → enrolled institution. PostgreSQL → borrow records.	0.55	0.40	PostgreSQL B-tree.	ArangoDB faster due to unified join query.
Identify trending topics from user feedback tags.	CouchDB → aggregate tags.	0.60	0.25	CouchDB MapReduce vs Arango inverted index.	ArangoDB wins with full-text + inverted index.
List users who liked and borrowed the same material.	Neo4j → USER -[:LIKED]-> MATERIAL.PostgreSQL → borrow.	0.47	0.23	Neo4j label index, PostgreSQL B-tree.	Arango faster: join inside one AQL query.

Faculty feedback sentiment on journals.	CouchDB → text feedback.Python sentiment analysis external.	1.20	0.95	CouchDB text index, Arango inverted index.	ArangoDB slightly better due to built-in text search.
Track borrowing trends per department.	PostgreSQL → join borrow + department.	0.36	0.85	PostgreSQL index on department.	Hybrid better: SQL analytic functions outperform.
Identify users who gave both high ratings and negative feedback.	Neo4j → ratings > 4.CouchDB → feedback sentiment negative.	0.66	0.42	Neo4j property index, CouchDB text index.	Arango faster: unified filter.
Suggest books based on co-borrowing patterns.	Neo4j → link prediction / similarity.	0.55	1.80	Neo4j graph projection.	Hybrid wins: Neo4j ML better optimized.
Faculty collaboration based on co-teaching.	Neo4j → detect FACULTY -[:TEACHES]-> COURSE.	0.40	1.60	Neo4j relationship index.	Neo4j faster, Arango slower on large graph.
Students who borrowed >10 books in a month.	PostgreSQL → aggregate borrow counts.	0.28	0.65	PostgreSQL B-tree.	Hybrid faster: SQL aggregation.
Match suppliers with top-rated books.	CouchDB → suppliers. Neo4j → ratings > threshold. PostgreSQL → books.	0.50	0.33	CouchDB MapReduce, Neo4j property index.	Arango faster: multi-model advantage.
Students borrowing e-materials vs. print materials.	PostgreSQL → filter borrow by type.	0.27	0.70	PostgreSQL B-tree.	Hybrid better: SQL query simpler.
Detect communities of students using both friendships and borrowing overlaps.	Neo4j → community detection on combined graph.	0.65	1.90	Neo4j adjacency list index.	Hybrid better: Neo4j specialized.
Predict student dropout risk using borrowing inactivity.	PostgreSQL → student enrollments. Neo4j → activity patterns.	0.80	0.60	Postgres B-tree, Neo4j label index.	Arango faster: combines doc + graph + time-series.

Results and Discussion

The comparison of ArangoDB and the Hybrid multimodel architecture demonstrate that the features such as efficiency, consistency, and scalability associated with complicated data management are highly affected due to integration in digital library systems.

In the literature review of Multi-model and polyglot persistence most authors states that multi-model performs better than polyglot persistence, but Sandell et al. (2024) [10] states that Neo4j performs better in graph traversal than ArangoDB. Hence we can say that, PostgreSQL, CouchDB, and Neo4j performed satisfactorily in their respective domain in the Hybrid model. In Fernandes, D., & Bernardino, J. (2018), authors intend to assist developers and researchers in choosing a suitable graph DBMS according to workload specifications, scalability demands, and data model adaptability [15]. Neo4j represented relationships and interactions between data successfully, unstructured and semi-structured data CouchDB managed efficiently, and reliable transactional data by PostgreSQL. But the reason for lacking behind is the Inter-database communication overhead. The complexity and latency of each multimodel query were enhanced if there is coordination

between three engines, middleware synchronization, and proper data format conversions between JSON, SQL, and Cypher. The comparative evaluation between ArangoDB and the Hybrid multimodel architecture highlights how architectural integration impacts the efficiency, consistency, and scalability of complex data management in digital library systems. In this research paper the efficiency is considered with respect to execution time required for the query from natural language to showing the execution output.

Conversely, the integrated architecture of ArangoDB eliminates the necessity for such coordination between the document, graph, key-value databases. The results shows that query latency was reduced and throughput was enhanced by integration of document, graph, and key-value models under a unified query language called as (AQL). The system executed multimodel joins directly within one engine, eliminating network transfers and costly transformations. This approach rendered ArangoDB especially effective for cross-model AQL queries, the correlation of user interests, materials, and enrollment data.

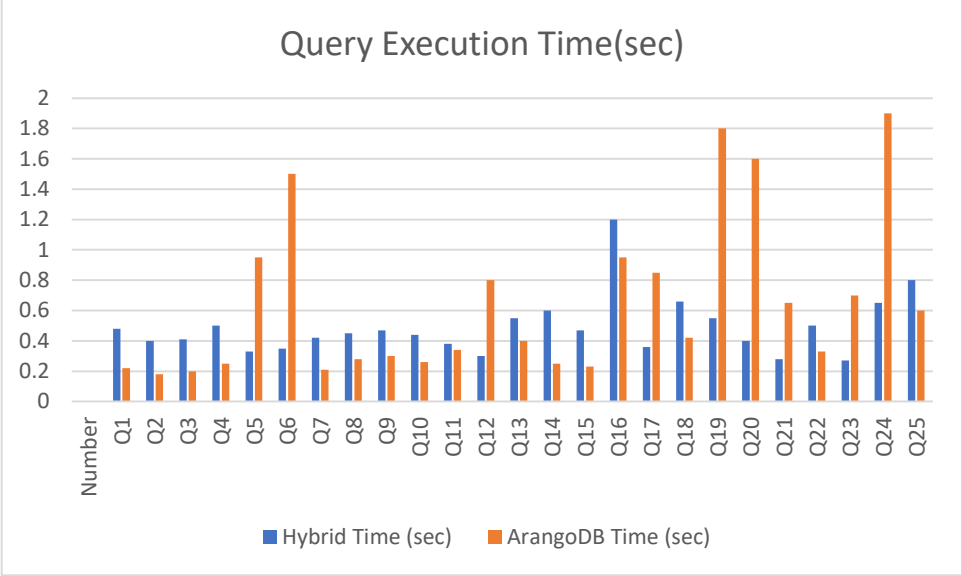

Figure 2 : Query Execution Time for ArangoDB and Hybrid Multimodel Database

As shown in above Figure 2, ArangoDB shows its results of evaluation criteria such as execution time better than the Hybrid model which is 47% for execution time and 42% in throughput, especially in document–graph queries. The separate indexing mechanisms in the Hybrid approach decreases its indexing structure compared to ArangoDB when it accesses paths for mixed workloads. However, the Hybrid system advantageous in transactional control and recovery due to PostgreSQL's ACID compliance and Neo4j's mature graph traversal algorithms.

User accessibility is significantly enhanced by integrating Natural Language Processing (NLP) in the both systems. Queries written in simple English such as "Find the Students who borrowed >10 books in a month" submitted by user were automatically translated into database-specific commands. This approach reduced the cognitive load on non-technical users and enabled inclusive data exploration. In ArangoDB, use of NLP was more advantageous since AQL allowed mixed-model querying directly, while query segmentation and mapping was required across three databases in the Hybrid system.

Overall, the Hybrid model remains advantageous for data integrity, modular design, and transactional workloads, while ArangoDB demonstrates superior performance in real-time analytics, graph-document integration, and usability.

Conclusion

This research shows the ArangoDB's ability in reducing execution time and lowering data movement overhead in processing document, graph and relational data in a single AQL query. Also, the research shows the detailed comparative performance between ArangoDB and a Hybrid multimodel database system for implementing the Library Information System (LIS). The results of features such as cross-model query performance, integration simplicity, and scalability shown by ArangoDB, being a native multimodel system, offers substantial benefits.

In contrast, the Hybrid architecture, offers superior transactional consistency and modular control but contains increased query latency. The systems which requires high dependability, structured data management, and graph analytics where specialized engines can be optimized independently are the suitable scenarios where hybrid model can be used.

The results shows that hybrid model in library domain such as book borrowing and supplier invoicing has 20% better consistency in processing heavy transactions than ArangoDB, but ArangoDB demonstrated up to 47% lower execution time and 42% higher throughput for integrated workloads.

The integration of NLP-driven query translation successfully bridged the gap between user intent and technical execution, demonstrating that database complexity can be hidden from end-users while maintaining system intelligence.

In conclusion, ArangoDB is more effective for unified, multimodel workloads with mixed data types and user-driven analytics, while the Hybrid model is optimal for modular, transaction-intensive, and distributed designs. Both approaches have complementary strengths and can potentially coexist in future hybridized intelligent systems.

Future Enhancements

Future work can include various ML algorithms within Neo4j and ArangoDB supporting recommendations for books, journals, or suppliers. Expanding NLP capabilities using Large Language Models (LLMs) like GPT or BERT can enable semantic understanding and intent-based query rewriting, making user queries even more natural and adaptive. A new multi-model could be developed to allow real-time synchronization between ArangoDB and the Hybrid databases, combining ArangoDB's integration strengths with PostgreSQL's transactional robustness. Scale the current dataset to hundreds of millions of records using distributed environments (Hadoop/Spark connectors) to validate the scalability of ArangoDB's cluster architecture versus a distributed hybrid stack. We can further also implement unified access control policies and encryption mechanisms to ensure data security across multimodel systems.

References

[1] Li, J., Lu, M., Dou, G., & Wang, S. (2017). Big data application framework and its feasibility analysis in library. *Information Discovery and Delivery*, *45*(4), 161-168.

[2] Kumar, T. V. (2015). Analysis of SQL and NOSQL database management systems intended for unstructured data.

[3] Lu, J., & Holubová, I. (2019). Multi-model databases: a new journey to handle the variety of data. *ACM Computing Surveys (CSUR)*, *52*(3), 1-38.

[4] Gavriilidis, H. (2025). *Query processing and interoperability mechanisms for federated data systems* (Doctoral dissertation, University of California, San Diego).

[5] Nagajayant Nagamani. (2022). Explainability and Robustness Trade-offs: Ensuring Safety and Fairness in Large-Scale AI Deployments. International Journal of Intelligent Systems and Applications in Engineering, 10(3s), 484–494. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/8017.

[6] Belgundi, R., Kulkarni, Y., & Jagdale, B. (2023, February). Analysis of native multi-model database using ArangoDB. In *Proceedings of third international conference on sustainable expert systems: ICSES 2022* (pp. 923-935). Singapore: Springer Nature Singapore.

[7] Corovcák, M., & Koupil, P. (2025). SQL vs NoSQL: Six Systems Compared. In *ENASE* (pp. 41-53).

[8] Kanchan, S., Kaur, P., & Apoorva, P. (2021). Empirical evaluation of nosql and relational database systems. *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science)*, *14*(8), 2637-2650.

[9] Singh, N., Chandra, R., Shambharkar, S. B., & Kulkarni, J. J. (2020). A review of NoSQL databases and performance comparison between multimodel and polyglot persistence approach. *IJSTR*, *9*(01), 1522-1527.

[10] Sandell, J., Asplund, E., Ayele, W. Y., & Duneld, M. (2024). Performance comparison analysis of ArangoDB, MySQL, and Neo4j: An experimental study of querying connected data. *arXiv preprint arXiv:2401.17482*.

[11] Harrington, M. D., & Christman, D. B. (2019). (Un) Structuring for the Next Generation:

New Possibilities for Library Data with NoSQL.

[12] Zdepski, C., Bini, A., & Matos, S. (2020). PDDM: A Database Design Method for Polyglot Persistence. *American Academic Scientific Research Journal for Engineering, Technology, and Sciences*, *71*(1), 136-152.

[13] Mohamed, A. K. Y. S., Auer, D., Hofer, D., & Küng, J. (2024). A systematic literature review of authorization and access control requirements and current state of the art for different database models. *International Journal of Web Information Systems*, *20*(1), 1-23.

[14] Kazanavičius, J., Mažeika, D., & Kalibatienė, D. (2022). An approach to migrate a monolith database into multi-model polyglot persistence based on microservice architecture: A case study for mainframe database. *Applied Sciences*, *12*(12), 6189.

[15] Fernandes, D., & Bernardino, J. (2018). Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB. *Data*, *18*, 373-380.

[16] Usta, A., Karakayali, A., & Ulusoy, Ö. (2024). xDBTagger: explainable natural language interface to databases using keyword mappings and schema graph. *The VLDB Journal*, *33*(2), 301-321.

[17] Ozsoy, M. G., Messallem, L., Besga, J., & Minneci, G. (2025, January). Text2cypher: Bridging natural language and graph databases. In *Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK)* (pp. 100-108).

[18] Fan, Y., He, Z., Ren, T., Huang, C., Jing, Y., Zhang, K., & Wang, X. S. (2024, May). Metasql: A generate-then-rank framework for natural language to sql translation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)* (pp. 1765-1778). IEEE.

[19] Ikwuanusi, U. F., Adepoju, P. A., & Odionu, C. S. (2023). Developing predictive analytics frameworks to optimize collection development in modern libraries. *International Journal of Scientific Research Updates*, *5*(2), 116-128.

[20] Qin, Z., Chen, S. J., Metzler, D., Noh, Y., Qin, J., & Wang, X. (2020, August). Attribute-based propensity for unbiased learning in recommender systems: Algorithm and case studies. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2359-2367).

[21] Sarode, A., & Khaparde, V. S. Automating Metadata: NLP's Contribution to Efficient Digital Library Management. *Emerging Technologies in Libraries: Trends and Developments*, 187.

[22] Ansari, N., Vakilimofrad, H., Mansoorizadeh, M., & Amiri, M. R. (2021). Using data mining techniques to predict user's behavior and create recommender systems in the libraries and information centers. *Global Knowledge, Memory and Communication*, *70*(6/7), 538-557.