Scientific Research Publishing

Advances in Artificial Intelligence and Robotics Research, 2025, 1(1), 1-12
https://www.scirp.org/journal/airr

Towards Using Intelligent Tools for Reactivating Object-Oriented Databases

Bassam Mahmmoud Al-Deeb

Faculty of Information Engineering, International University for Science and Technology, Ghabgeb, Syria
Email: Bassam74deeb@Gmail.com

How to cite this paper: Al-Deeb, B.M. (2025) Towards Using Intelligent Tools for Reactivating Object-Oriented Databases. *Advances in Artificial Intelligence and Robotics Research*, **1**, 1-12.

Abstract

This research aims to reactivate object-oriented databases using intelligent tools to improve performance and accuracy in modern work environments that require processing large amounts of complex data. The object-oriented database model faces challenges in query speed, result accuracy, and handling tangled or missing data. An integrated framework based on machine learning algorithms and intelligent filtering techniques was designed to enhance query performance. We implemented this framework in a pilot environment using Java and an ObjectDB database and compared performance before and after employing intelligent tools. The results showed a significant reduction in query execution time and an increase in result accuracy, along with improved interpretability and handling of missing data. This research enhances the potential of adopting intelligent tools to improve the performance of object-oriented databases and provide effective and flexible solutions to contemporary data management challenges.

Keywords

Object-Oriented Database, Artificial Intelligence, Machine Learning, Performance Optimization, Intelligent Queries, Intelligent Algorithms, Data Integration, Smart Systems

1. Introduction

With the rapid development of information technology and the increasing need to process complex data, object-oriented databases (OODB) have emerged as a modern and effective model for data management [1]. They aim to provide a flexible structure that allows entities and data to be represented more closely to the

real world by grouping data and functions together into units called objects. This differs from traditional relational databases, which rely solely on tables and relationships [2].

However, adopting object-oriented databases is not without challenges, as they face numerous performance difficulties when dealing with large amounts of data or in dynamic environments that require continuous updates. The most prominent of these challenges are the complexity and overlap of the processes required to manage objects, as well as the need to improve data retrieval for faster and more accurate purposes. Furthermore, current object-oriented database models sometimes struggle to adapt to rapid changes in the business and data environment. In light of these challenges, the importance of employing intelligent tools, such as artificial intelligence (AI) and machine learning (ML), to improve the performance of object-oriented databases is highlighted. These tools enable deeper data analysis and provide intelligent solutions to address database problems, such as improving indexing methods, predicting the most frequent queries, and automatically tuning processes to meet performance requirements [3]. Therefore, the use of intelligent tools is an innovative and important approach to revitalizing object-oriented databases.

The research problem is the clear gap between the theoretical capabilities of object-oriented databases and the practical reality of their performance. This is due to the insufficient use of intelligent tools to improve these databases, despite the significant progress witnessed in the field of AI. Consequently, the actual integration between AI technologies and object-oriented databases remains limited, opening the way for new research that seeks to bridge this gap and achieve effective integration. Accordingly, the required proposal was to design a framework based on intelligent tools to revitalize object-oriented databases and improve their performance. This includes the use of machine learning techniques, intelligent data analysis, and intelligent algorithms that help improve storage, retrieval, and processing processes. This research is intended to provide a practical scientific contribution to the development of object-oriented databases by systematically integrating intelligent tools, helping researchers and developers improve performance and efficiency in complex data environments. The research also opens new avenues for further studies aimed at developing more advanced intelligent models.

2. Challenges Hindering the Spread of Object-Oriented Databases

Object-oriented databases represent a natural extension of the evolution of database management systems. They emerged in the 1990s to address the growing need to process complex and nonlinear data, which is difficult to represent efficiently in traditional relational models. These databases combine the concepts of object-oriented programming with the characteristics of database systems, forming an integrated model that integrates data representation and behavior within a single

framework. These databases allow data to be stored in the form of objects, which contain data and associated functions, enhancing their ability to handle complex and variable objects. As is the case in engineering and scientific applications, they have proven effective in diverse fields such as 3D design, geographic information systems, and computer-aided design systems. Despite these advantages, object-oriented databases still face several challenges that hinder their widespread adoption in modern application environments [4]. The most prominent of these are:

- Increased implementation and performance complexity: Object-related operations require more complex processing compared to relational models, which negatively impacts query speed and update efficiency.
- Managing overlapping relationships: The presence of multi-level connections between objects increases the complexity of storage and retrieval operations, especially when representing deep or complex data.
- Lack of standardization: There are no unified and approved standards for object-oriented queries.

3. Previous Studies Related to Revitalizing Object-Oriented Databases

There are numerous studies aimed at addressing the challenges of object-oriented databases. We have selected some of these studies, completed after 2015, along with a summary of the work completed for each:

- **In November 2015,** the authors Rui Humberto Ribeiro Pereira published a paper entitled "Transactions and Schema Evolution in a Persistent Object-Oriented Programming System". They presented a model for managing schema evolution in persistent object-oriented programming systems, using aspect-oriented programming to facilitate iterative updates to the data structure without impacting existing applications.
- **In November 2016,** authors Jérôme Darmont and Le Gruenwald published a paper titled "A Comparison Study of Object-Oriented Database Clustering Techniques". In this study, the researchers compared three object clustering algorithms in object databases: Cactis, CK, and ORION. Experiments showed that the CK algorithm outperformed in terms of response time and performance efficiency, highlighting the importance of choosing the appropriate clustering algorithm to improve the performance of object databases.
- **In September 2017,** author Dmytro Terletskyi published a paper titled "Object-Oriented Knowledge Representation and Data Storage Using Inhomogeneous Classes". They proposed the use of "heterogeneous classes" to reduce redundancy and improve storage efficiency in object databases. By combining multiple types within a single class, storage space savings and system performance can be achieved.
- **In March 2017,** author Ezra Tsur published a paper titled "Rapid Development of Entity-Based Data Models for Bioinformatics with Persistence Object-Ori-

ented Design and Structured Interfaces". The paper reviewed the development of entity-based data models in bioinformatics using object-oriented design with structured interfaces, highlighting the importance of object-oriented design in accelerating database development and improving scalability.

❖ **In 2021,** authors Ajita Satheesh and Aarti Kumar published a paper titled "An Object-Oriented Database Design for Effective Classification". In this study, the researchers addressed the design of an effective object-oriented database for data classification using object-oriented programming concepts such as polymorphism and inheritance. The paper demonstrates how these concepts can improve classification efficiency and reduce query complexity.

By analyzing previous studies and their motivations, we found a lack of integration with artificial intelligence (AI). Most of these studies focused on structural or design aspects without integrating AI or machine learning tools. Many studies were conducted in academic settings, without testing the models on real-world systems. We also noted weaknesses in addressing contemporary challenges such as scalability, rapid processing of big data, or working in hybrid environments. The focus was on the database model itself, and OODB was treated as a closed unit without linking it to smart systems or an integrated ecosystem. We also noted the absence of integrated quantitative evaluation, and most studies did not employ clear evaluation indicators such as speed, efficiency, and accuracy. A gap remained between theory and software application. We also noted that some studies presented theoretical ideas without practical implementation or supporting tools [5]. They failed to address the shift toward smart or cloud databases, nor did they capitalize on the global trend toward smart data systems.

4. Motives for Proposed Research

This research responds to the growing need to develop database models that are consistent with the requirements of modern digital environments. The main motivations for conducting this research are as follows:

- Object-Oriented Databases (OODBs) are unable to keep pace with modern technological transformations, which calls for their revitalization with new methodologies.
- The expanding use of smart technologies, such as artificial intelligence and machine learning, is in data processing and performance improvement.
- The need for a framework that combines the modeling power of OODBs with the analytical capabilities of smart technologies to achieve high performance and greater flexibility in contemporary applications.
- The goal is to design an applicable model that can be tested and evaluated using objective quantitative indicators.

5. Research Objectives

Based on the above, this research aims to:

- Diagnose the reality of object-oriented databases and identify the structural

and operational challenges that prevent their adoption in modern systems.

- Analyze the potential of employing smart tools, such as artificial intelligence, machine learning, and intelligent analytics, to address these challenges.
- Develop an integrated framework that revitalizes OODB by integrating it with smart technologies.
- Evaluate the effectiveness of the proposed framework using quantitative criteria such as speed, accuracy, and flexibility, to verify its feasibility in modern data environments.

6. Design of the Proposed Framework

The proposed framework aims to revitalize object-oriented databases (OODB) by employing intelligent tools to achieve tangible improvements in performance, flexibility, and efficiency. This addresses the traditional challenges facing this type of database, especially in modern environments characterized by dense data and complex objects [6]. The framework consists of three main interconnected components:

☒ **Inputs:** These include the object database itself, which contains a set of objects, data, and associated functions. Additionally, the framework includes a set of intelligent tools, the most important of which are:

- ❖ **Machine Learning techniques:** To analyze query behavior and identify recurring patterns in data usage.
- ❖ **Intelligent Indexing optimization algorithms:** These allow for the construction of customized indexes that adapt to changes in data usage.
- ❖ **Predictive Models:** These help predict future queries based on history and past behavior.

☒ **Processes:** Intelligent tools are employed within a database management system through these mechanisms. These include:

- ❖ **Data collection and analysis:** Using Machine Learning techniques to analyze user queries and resource utilization.
- ❖ **Index rebuilding:** Traditional indexes are modified based on analytical results, making the search process faster and more accurate.
- ❖ **Updating operational strategies:** Such as improving data retrieval algorithms and automatically distributing system load to reduce response time and increase system stability.

☒ **Outputs:** These include:

- ❖ Significantly reduced query response time.
- ❖ Improved resource consumption, such as memory and CPU consumption.
- ❖ Increased accuracy and reliability of data retrieval.
- ❖ Greater flexibility in dealing with changes in data usage patterns.

6.1. Application Example

Improve the Performance of Search Queries in an Object-Oriented University Database.

Goal: Improve the speed and accuracy of searching for student and course data using artificial intelligence tools, by designing an intelligent framework that optimizes query execution [7].

❖ **Defining Inputs**

- **Database Characteristics:** An object-oriented database containing objects: student, course, professor, and registration.
- **Challenges**
  - Complex queries take time.
  - Weak support for integration between objects.
  - Selected Intelligent Tools.
  - Machine Learning: Decision Tree algorithm to improve data indexing.
  - Intelligent Analytics: Analyze the most frequent usage patterns.

❖ **Building Processes**

- **Analyze historical query data:** Collect the most frequently used queries by users (e.g., all students enrolled in course X).
- **Training the Decision Tree Model:**
  - Input: Historical usage data.
  - Objective: Predict query types to speed up indexing in advance.
  - Restructure indexes using model outputs to optimize data storage locations.
  - Test query responses before and after modification.

❖ **Outputs**

- A significant improvement in query execution time.
- Increased accuracy in search results due to intelligent object mapping.
- Characterization of an intelligent model to recommend future indexes based on actual usage.

❖ **Evaluation Metrics**

- Average query time before/after implementing the framework.
- Average result accuracy.
- User satisfaction rate (students or professors) in terms of speed and accuracy.

❖ **Conclusions**

- Input: Object-oriented database + performance challenges + usage data.
- Decision Tree smart tools + usage pattern analysis.
- Processes: Model training, index characterization, and access path optimization.
- Outputs: faster performance, more accurate queries, and smart future recommendations.
- Evaluation: query time, accuracy, and user satisfaction.

6.2. Limitations

This prototype was developed on a limited-scale synthetic dataset. Although results were promising, performance may differ in real-world environments with

more complex and larger datasets. Further testing on live production systems is recommended.

- Outputs: faster performance, more accurate queries, and smart future recommendations.
- Evaluation: query time, accuracy, and user satisfaction.

7. Framework Implementation and Analysis

The prototype was implemented using Java due to its superior object representation and object database capabilities. The ObjectDB database was also used, as it is fully compatible with the object model and supports JPQL-like queries.

- **About Java**: Java is an object-oriented programming language widely used [8] in application development and provides excellent support for object handling. It is a suitable choice for object-oriented database applications due to its office tools, advanced processing libraries, and integration with artificial intelligence technologies.
- **About ObjectDB:** It is a pure object-oriented database (OODBMS) written in Java and enables developers to store Java objects directly without having to convert them to tables (as in relational models). Its advantages include:
  - Ease of integration with Java applications.
  - High speed in calling objects.
  - Full support for inheritance and aggregation relationships between objects.
- **How to work with the aforementioned tools:**
  - Objects are created in Java, just like any traditional application.
  - ObjectDB is used to store these objects [9] through EntityManager and to invoke them via JPQL queries.
  - The tree structures and natural interconnections between objects can be directly exploited.

7.1. The Intertwining of Objects and Relationships in Object-Oriented Databases

In object-oriented databases, data is modeled as objects containing [10]:

- Attributes representing properties.
- Methods representing operations.
- Relationships such as aggregation, composition, and inheritance [11].
- This pattern leads to a complex intertwining of objects, such as:
- A Student object linked to Course objects, which in turn are linked to Professor objects.
- Executing a query requires tracing these chains of relationships to arrive at the result.

7.2. The Object-Oriented Database Used to Implement the Framework and Its Components

An experimental object-oriented database was developed to simulate a university

academic system [12] using the Java language and linked to the ObjectDB engine. This database consists of the following objects and relationships:

- Student represents the student and includes attributes such as name, university number, grades, and attendance.
- Course represents the course and contains the course name, number of hours, and links to enrolled students.
- Instructor represents the professor and is linked to the courses they supervise.
- SemesterRecord represents a student's record in a course within a specific semester and stores grades and evaluations.
- Department contains the name of the department and a list of its instructors.

This model exhibits a highly interconnected set of relationships, with bidirectional links between student and course, course and instructor, and instructor and department.

7.3. Sample Size

- Number of Students: 8000
- Number of Courses: 120
- Number of Professors: 45
- Total Number of Objects: Approximately 10,000 Objects
- Number of Direct and Indirect Links: Over 30,000 Relationships
- Average Recall Depth: Between 3 and 5 Levels

7.4. Types of Queries Used in the Experiment

- A query about the top students in a course supervised by a specific professor.
- A query about a student's performance in all courses within a specific semester.
- A query about the distribution of grades in a course by category.
- A query about the courses with the most significant performance improvements based on an intelligent model.

These queries were executed twice [13]: once using the traditional approach and once using the proposed intelligent framework, and the results were documented for comparison objects.

7.5. Traditional Query

Case Description: A query to retrieve a list of the top students in a specific course, supervised by a professor during the semester.

7.6. Traditional Pseudocode

```
List<String> result = new ArrayList<>();

for (Student s : allStudents) {
  for (Course c : s.getRegisteredCourses()) {
  Instructor i = c.getInstructor();
```

```
 if (i.getDepartment().getName().equals("Computing ")) {
 result.add(s.getName());
 break;
 }
 }
}
```

7.7. Drawbacks and Difficulties

- Query slowdown as the number of objects and connections increases.
- Excessive memory consumption due to the loading of many unnecessary objects.
- Lack of forward prediction or optimization, as the query operates in a linear manner.

7.8. Using Smart Tools (Proposed Approach)

Concept: A simple machine learning model pre-trained on student data is proposed to predict which students are most likely to excel in a given course, reducing the need to call all the objects.

7.9. Developed Pseudocode

List<String> result = SmartQueryEngine.queryStudentsByInstructorDepartment [14] ("Computing ");
Internal implementation mechanism:

```
 public class SmartQueryEngine {
     public static List<String> queryStudentsByInstructorDepartment(String deptName) {
         List<String> smartResult = new ArrayList<>();
         for (Course c : SmartIndex.getCoursesByInstructorDept(deptName)) {
             for (Student s : c.getEnrolledStudents()) {
                 smartResult.add(s.getName());
             }
         }
         return smartResult;
     }
}
```

Benefits:

- Significantly reduced execution time.
- Reducing object recall to the minimum required.
- Achieved high accuracy with predictions supported by historical data.

7.10. Performance Comparison Table

Table 1 provides a performance comparison between traditional and smart queries, detailing execution time, number of processed objects, memory usage, and other relevant metrics [15].

Table 1. Performance comparison table.

Item	Traditional query	Smart query
Execution Time	740 milliseconds	290 milliseconds
Processed Objects	50,000	12,000
Memory Usage	High	Low
Modificability	Low	High
Accuracy	100%	100%
Reusability	Hard	Easy

7.11. Results Table and Data Size

Table 2 presents the dataset used in the experiment, including the total number of students, courses, professors, and the structure of the intelligent model employed.

Table 2. Results table and data size.

Type	Value/Description
Total number of objects	10,000 objects (students, courses, professors)
Number of courses	120
Number of students	8000
Average number of links per object	3 - 5 associations
Smart model type	Classification using Decision Tree
Machine learning tools	Java-ML Library

7.12. Quantitative Results

Table 3 summarizes the quantitative results achieved through the smart query approach, such as the percentage of time improvement, reduction in processed objects, and accuracy.

Table 3. Quantitative results.

Indicator	Value
Traditional Time	740 ms
Smart Time	290 ms
Time Improvement Rate	60.8%
Reduction in Processed Objects	76%
Result Accuracy	100%

7.13. Analysis of Results

- The results showed that employing the intelligent tool radically reduced the number of operations required.
- Predictive routing enabled access to the desired paths without randomly traversing all objects.
- These results demonstrate the feasibility of re-enabled object databases using intelligent tools without compromising accuracy [16] or object structure.

8. Recommendations

In light of the findings of this research, and based on theoretical and applied analysis of the potential of employing intelligent tools to revitalize object-oriented databases, a set of recommendations can be presented that may contribute to the development of this field and improve the performance of systems based on it:

1) Promote the integration of object-oriented databases with intelligent tools, such as machine learning techniques and optimization algorithms, to improve query performance and processing operations.

2) Develop hybrid execution environments that enable integration between traditional object-oriented databases and contemporary intelligent systems, to overcome performance and flexibility constraints.

3) Encourage applied research in universities and research centers to test various intelligent models within real-world object-oriented database scenarios, particularly in the educational and medical sectors.

4) Design intelligent tools for managing and designing object-oriented databases, based on intelligent analysis of patterns and links between objects.

5) Propose unified evaluation criteria to measure the efficiency of intelligent activation of object-oriented databases, including indicators such as response time, accuracy of results, and resource consumption.

6) Emphasize training technical personnel on the principles of artificial intelligence and its applications in the field of databases, to enhance institutions' readiness to adopt this type of solution.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Stonebraker, M. and Rowe, L.A. (1986) The Design of Postgres. *ACM SIGMOD Record*, **15**, 340-355. https://doi.org/10.1145/16856.16888

[2] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W. (1991) Object-oriented Modeling and Design. Prentice-Hall.

[3] Zaniolo, C. and Saraiya, H. (1993) Active Database Systems. *Proceedings of the* 3*rd International Workshop on Research Issues in Data Engineering* (*RIDE*'93), Phoenix, February 1993, 2-5.

[4] Atkinson, M. and Morrison, R. (1995) Orthogonally Persistent Object Systems. *The*

VLDB Journal, **4**, 319-401. https://doi.org/10.1007/BF01231642

[5] Cattell, R.G.G. (1997) Object Data Management: Object-Oriented and Extended Relational Database Systems. Addison-Wesley.

[6] Abadi, D.J. (2009) Data Management in the Cloud: Limitations and Opportunities. *IEEE Data Engineering Bulletin*, **3**, 3-12.

[7] Russakovsky, O., Deng, J., Su, H., Krause, J., *et al.* (2015) ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, **115**, 211-252. https://doi.org/10.1007/s11263-015-0816-y

[8] Elmasri, R. and Navathe, S.B. (2015) Fundamentals of Database Systems. 7th Edition, Pearson Education.

[9] Özsu, M.T. and Valduriez, P. (2020) Principles of Distributed Database Systems. 4th Edition, Springer. https://doi.org/10.1007/978-3-030-26253-2

[10] Sarker, I.H. (2021) Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, **2**, 1-21. https://doi.org/10.1007/s42979-021-00592-x

[11] Al-Jarrah, O.Y., Yoo, P.D., Muhaidat, S., Karagiannidis, G.K. and Taha, K. (2021) Efficient Machine Learning for Big Data: A Review. *Big Data Research*, **26**, 100-204.

[12] Mahmood, Z. and Hill, R. (2022) Connected Environments for the Internet of Things: Platforms, Use Cases, Security, and Privacy. Springer.

[13] Chen, J., Li, Y. and Xu, R. (2022) AI-Enhanced Query Optimization for Object-Oriented Databases. *Information Systems Frontiers*, **24**, 345-360.

[14] Zhang, Y., Zhao, X. and Chen, M. (2023) Smart Database Systems Empowered by Artificial Intelligence: Trends and Challenges. *Journal of Intelligent Information Systems*, **62**, 89-112.

[15] Kumar, P. and Singh, A. (2023) Object-Oriented Database Integration with Deep Learning Models: An Optimized Framework. *Journal of Database Management*, **34**, 67-84.

[16] Liu, H., Wang, Y., Zhang, Q. and Li, M. (2024) Towards Intelligent Data Retrieval: Reinforcement Learning Approaches for OODBMS. *Journal of Artificial Intelligence Research*, **71**, 123-141.