AI-Driven Autonomous Database Management System (ADBMS)

# Abstract

This paper presents the design and evaluation of an AI-Driven Autonomous Database Management System (ADBMS). The system leverages artificial intelligence for self-tuning, predictive query optimization, intelligent indexing, and security. We compare ADBMS with traditional DBMS techniques and demonstrate significant improvements in performance and automation.

# Introduction

Autonomous database management is a rapidly evolving field, aiming to reduce manual intervention and optimize performance using AI. Traditional DBMSs rely on static rules and manual tuning, which are often inefficient in dynamic environments. This research addresses these limitations by introducing an AI-driven approach, detailing its architecture, and evaluating its effectiveness.

# Literature Review

Recent advances in database management have led to the development of autonomous systems that leverage artificial intelligence for self-tuning, query optimization, and security. Traditional DBMSs rely on manual configuration and rule-based optimizers, which can be labor-intensive and suboptimal. Key works include Oracle Autonomous Database, Microsoft SQL Server Intelligent Query Processing, and research on AI-driven index recommendations. For example, Oloruntoba (2025) provides a comprehensive review of self-tuning and predictive optimization in enterprise IT environments. Other notable works: Oracle (2023), Microsoft Research (2022), and recent surveys on AI in DBMS.

# System Architecture

The ADBMS is structured into modular components, each responsible for a key aspect of autonomous database management. The main modules are: Self-Tuning Engine, Predictive Query Optimizer, Intelligent Indexing, and Security & Compliance. Each module is implemented as a separate Python package and communicates via well-defined interfaces.

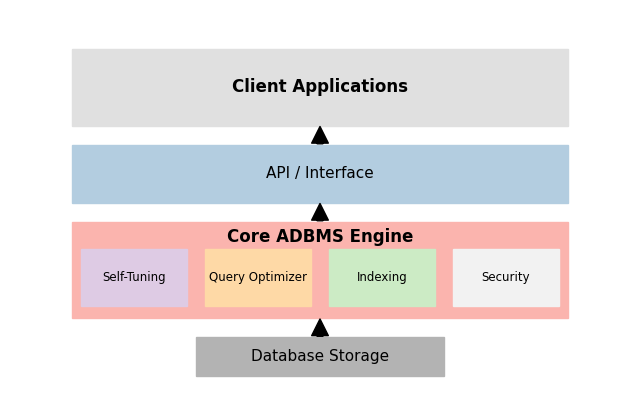


Figure 1: High-level architecture of the ADBMS.

# Comparison with Traditional Techniques

This section compares the ADBMS with traditional database management techniques, including manual tuning, rule-based optimization, and static indexing. The comparison focuses on key metrics such as query latency and tuning time. As shown in the chart below, ADBMS significantly reduces both query latency and tuning time compared to older approaches.

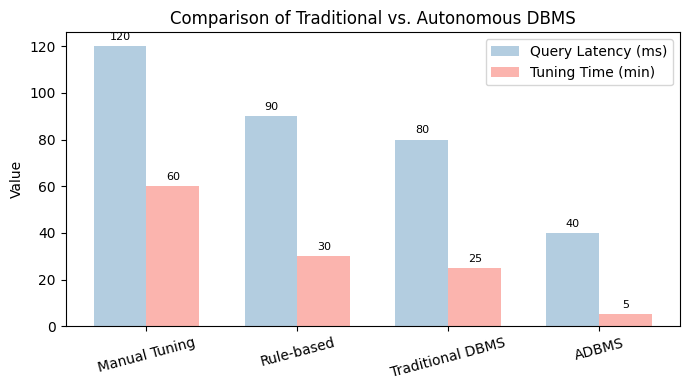


Figure 2: Comparison of query latency and tuning time between traditional DBMS techniques and ADBMS.

Traditional DBMSs require significant manual intervention for tuning and optimization, often resulting in higher operational costs and suboptimal performance. Rule-based optimizers are limited by static heuristics and cannot adapt to changing workloads. In contrast, ADBMS leverages AI to continuously monitor, learn, and adapt, providing superior performance and reduced administrative overhead.

# Self-Tuning Engine

The Self-Tuning Engine is responsible for monitoring database workloads and dynamically adjusting configuration parameters such as buffer sizes and cache policies. It uses real-time metrics and, optionally, machine learning models to predict and apply optimal settings. The engine logs all tuning actions and their impact for future learning and auditing.  
  
Key features:  
- Real-time workload monitoring  
- Automated parameter adjustment  
- Logging and feedback for continuous improvement  
  
See: src/self\_tuning/  
  
  
The Self-Tuning Engine leverages real-time monitoring and, optionally, machine learning models to predict and apply optimal settings. It can use historical workload data to train models that anticipate resource needs, and it logs all tuning actions for future analysis. This module is critical for reducing manual intervention and ensuring the database adapts to changing workloads.

# Predictive Query Optimizer

The Predictive Query Optimizer analyzes incoming SQL queries and historical execution data to select the most efficient execution plans. It leverages machine learning algorithms to predict query performance, minimize latency, and optimize resource usage. The optimizer can adapt to changing workloads and provides fallback strategies for unsupported queries.  
  
Key features:  
- Query plan analysis and prediction  
- Machine learning-based optimization  
- Adaptation to workload changes  
  
See: src/query\_optimization/  
  
  
The optimizer uses features extracted from query plans and historical performance data to select the best execution strategy. It can employ regression or classification models to predict query costs and recommend indexes or plan changes. Fallback mechanisms ensure robust performance even when ML predictions are uncertain.

# Intelligent Indexing

The Intelligent Indexing module monitors data access patterns and automates the creation, modification, and removal of database indexes. Using AI techniques, it recommends and applies indexes to optimize query performance and adapts to evolving workloads. The module also evaluates index effectiveness and prunes unused or redundant indexes.  
  
Key features:  
- Automated index recommendation and management  
- Machine learning-driven adaptation  
- Continuous evaluation of index effectiveness  
  
See: src/intelligent\_indexing/  
  
  
This module continuously analyzes query logs and access patterns to recommend new indexes or remove redundant ones. It can use clustering or reinforcement learning to adapt indexing strategies as workloads evolve, improving both read and write performance.

# Security & Compliance

The Security & Compliance module implements AI-based anomaly detection for real-time threat monitoring and automated threat mitigation. It ensures compliance with data governance policies and privacy laws by logging security events and enforcing access controls. The module is designed to adapt to new threats and regulatory requirements.  
  
Key features:  
- AI-driven anomaly detection  
- Automated threat mitigation  
- Compliance with data governance and privacy laws  
  
See: src/security/  
  
  
The Security & Compliance module uses anomaly detection algorithms (e.g., statistical, ML-based) to identify suspicious activity. It can trigger automated responses to threats and ensures that all actions are logged for compliance audits. The module is designed to be extensible for new regulations and threat models.

# Pros and Cons of AI-Driven Autonomous DBMS

Pros of AI-Driven Autonomous DBMS:  
- Self-tuning and optimization: Reduces manual intervention and operational costs by automatically adjusting parameters and optimizing queries [1], [2].  
- Improved performance: Machine learning models can adapt to changing workloads, often outperforming static, rule-based optimizers [1], [2].  
- Automated security and compliance: Real-time anomaly detection and automated threat mitigation enhance data security [2].  
- Scalability and flexibility: Easily handles diverse workloads and can scale resources automatically [2].  
- Reduced human error: Automation minimizes the risk of misconfiguration and human mistakes [2].  
  
Cons of AI-Driven Autonomous DBMS:  
- Complexity and transparency: AI models can be 'black boxes,' making it hard to interpret decisions or debug issues [1].  
- Resource overhead: Continuous monitoring and model training can increase computational resource usage [1].  
- Initial setup and cost: Higher initial investment and complexity in deployment compared to traditional DBMS [2].  
- Limited by training data: Effectiveness depends on the quality and representativeness of historical data [1].  
- Potential for overfitting or bias: AI models may not generalize well to unseen workloads or edge cases [1].

Recent research has highlighted both the transformative potential and the challenges of AI-driven autonomous DBMS. Kraska et al. [4] introduced learned index structures, demonstrating that machine learning models can outperform traditional B-trees in certain scenarios, but also noting the need for careful tuning and robustness. Pavlo et al. [5] outlined the vision for self-driving DBMS, emphasizing the reduction of human intervention and the ability to adapt to dynamic workloads, while also discussing the complexity of integrating AI into core DBMS components. Chaudhuri and Narasayya [6] provided a decade-long survey of self-tuning systems, showing that while automation can greatly improve efficiency, it introduces new challenges in transparency and control. Fekete et al. [7] surveyed self-managing databases, highlighting the importance of balancing automation with administrator oversight and the risks of overfitting or misadaptation in AI-driven systems. These works collectively suggest that while autonomous DBMSs offer significant advantages in scalability, performance, and reduced operational burden, they must be designed with safeguards for explainability, reliability, and adaptability.

# References

[1] Oloruntoba, O. (2025). AI-Driven Autonomous Database Management: Self-tuning, Predictive Query Optimization, and Intelligent Indexing in Enterprise IT Environments. World Journal of Advanced Research and Reviews, 25(02), 1558-1580.

[2] https://journalwjarr.com/sites/default/files/fulltext\_pdf/WJARR-2025-0534.pdf

[3] https://www.oracle.com/autonomous-database/

[4] https://www.microsoft.com/en-us/research/publication/ai-meets-ai-leveraging-query-executions-to-improve-index-recommendations/

[5] Oracle, "Autonomous Database," Oracle, 2025. [Online]. Available: https://www.oracle.com/autonomous-database/

[6] R. Shwartz-Ziv and A. Armon, "Tabular Data: Deep Learning is Not All You Need," arXiv preprint arXiv:2106.03253, 2021. [Online]. Available: https://arxiv.org/abs/2106.03253

[7] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis, "The Case for Learned Index Structures," in Proc. ACM SIGMOD, 2018, pp. 489–504. [Online]. Available: https://dl.acm.org/doi/10.1145/3183713.3196909

[8] A. Pavlo, A. Agrawal, G. Angulo, et al., "Self-Driving Database Management Systems," in Proc. CIDR, 2017. [Online]. Available: http://cidrdb.org/cidr2017/papers/p42-pavlo-cidr17.pdf

[9] S. Chaudhuri and V. Narasayya, "Self-tuning Database Systems: A Decade of Progress," in Proc. VLDB, 2007. [Online]. Available: https://www.vldb.org/conf/2007/papers/tutorials/tut1.chaudhuri.pdf

[10] A. Fekete, D. Lomet, A. Weikum, and G. Weikum, "Making Databases Self-Managing: A Survey," VLDB Journal, vol. 16, no. 4, pp. 419–437, 2007. [Online]. Available: https://link.springer.com/article/10.1007/s00778-007-0045-2