UDC 004.42

USE OF GRAPH DATABASES FOR USER BEHAVIOR ANALYSIS

Bastrykin Y.L.
master's degree, University of Edinburgh
(Edinburgh, United Kingdom)

Yumasheva N.B.
master's degree, University of Edinburgh
(Edinburgh, United Kingdom)

ИСПОЛЬЗОВАНИЕ ГРАФОВЫХ БАЗ ДАННЫХ ДЛЯ АНАЛИЗА ПОЛЬЗОВАТЕЛЬСКОГО ПОВЕДЕНИЯ

Бастрыкин Ю.Л.
магистр, Эдинбургский университет
(Эдинбург, Великобритания)

Юмашева Н.Б.
магистр, Эдинбургский университет
(Эдинбург, Великобритания)

Abstract

This paper investigates the application of graph databases for user behavior analysis, highlighting their advantages over relational models in representing and querying complex interaction patterns. Key aspects explored include data modeling techniques, query strategies using Cypher, integration into analytics pipelines, graph algorithm use cases, and system performance comparisons. Visual analyses and empirical benchmarks demonstrate the efficiency of graph-native operations in behavioral contexts, particularly for multi-hop queries and influence modeling. The study also addresses operational challenges and outlines emerging trends such as graph machine learning and temporal graph modeling. The results support the adoption of graph databases as a core component of intelligent, relationship-aware analytical systems.

Keywords: Graph databases, user behavior analysis, Cypher queries, graph algorithms, data modeling, performance comparison, behavior analytics, temporal graphs.

Аннотация

В статье рассматривается применение графовых баз данных для анализа поведения пользователей и подчёркиваются их преимущества перед реляционными моделями при работе с многосвязными структурами взаимодействий. Проанализированы подходы к моделированию данных, реализация запросов на языке Cypher, интеграция в аналитические пайплайны, применение графовых алгоритмов и сравнительная оценка производительности систем. На основе визуализаций и эмпирических тестов показана эффективность графового подхода в поведенческих сценариях, особенно при работе с глубокой связностью и моделированием влияния. Также обсуждаются эксплуатационные ограничения и ключевые перспективы, включая графовое машинное обучение и временные графы. Полученные результаты подтверждают целесообразность внедрения графовых СУБД как основы для построения интеллектуальных и ориентированных на отношения аналитических платформ.

Ключевые слова: Графовые базы данных, анализ поведения пользователей, Cypher-запросы, графовые алгоритмы, моделирование данных, сравнение производительности, поведенческая аналитика, временные графы.

Introduction

In recent years, the exponential growth of digital interaction data has driven the demand for more sophisticated tools to capture, represent, and analyze user behavior. Traditional relational databases, while effective for structured and tabular data, often struggle to model the complex, interconnected patterns inherent in user activity logs, social media footprints, and recommendation systems. This limitation has led to an increased interest in graph-based data representations, where entities and their relationships are treated as first-class citizens in the data model.

Graph databases (GDBs), such as Neo4j, Amazon Neptune, and ArangoDB, offer native support for relationship-centric data structures, enabling efficient traversal and pattern matching. Unlike relational models that rely heavily on joins, GDBs provide direct edge-based connections between nodes, significantly reducing query latency when exploring behavioral paths or networked interactions. These features make them especially suitable for applications involving user segmentation, fraud detection, influence mapping, and personalization engines, where the semantics of relationships are as critical as the attributes of individual users.

This paper aims to examine the practical application of graph databases in user behavior analysis, focusing on modeling techniques, query strategies, and performance trade-offs. The study compares graph-based and relational approaches using real-world datasets and demonstrates how graph algorithms-such as community detection, centrality metrics, and path analysis-can extract meaningful behavioral patterns. Through visual models, query examples, and benchmarking tables, the paper outlines best practices for designing graph-powered analytical workflows in dynamic digital environments.

Main part. Modeling user behavior data in relational and graph databases

Modeling user behavior requires not only capturing discrete user actions, but also representing the relationships and dependencies between those actions. In relational database systems, this typically involves multiple normalized tables and foreign key constraints, where joins are used to reconstruct interaction histories. However, as the complexity and density of relationships increase-such as in social graphs, clickstreams, or recommendation engines-the performance and clarity of relational models degrade rapidly.

Graph databases offer an alternative paradigm in which users, actions, sessions, and resources are represented as nodes, and the relationships between them (e.g., «clicked», «viewed», «follows», «purchased») are stored as edges. This structure allows for more intuitive modeling and enables recursive traversal operations with significantly lower computational cost. Queries that would require nested joins in SQL can often be expressed as short, expressive traversal patterns in graph query languages such as Cypher or Gremlin [1].

The following table 1 summarizes the key conceptual and operational differences between relational and graph-based approaches to user behavior modeling.

Table 1

Comparison of relational and graph database models for user behavior analysis

Aspect	Relational model	Graph model
Data structure	Tables with rows and foreign keys	Nodes and edges representing entities and relationships
Relationship representation	Indirect (via joins)	Direct (via edges)
Query complexity	High for multi-hop relationships	Low for recursive traversal
Performance with deep links	Degrades with number of joins	Stable with graph traversal

Aspect	Relational model	Graph model
Schema flexibility	Rigid, predefined schemas	Schema-optional, supports heterogeneous data
Use cases	Transactional systems, structured tabular data	Behavioral analysis, recommendations, social networks

This comparison illustrates that graph databases provide significant advantages in scenarios where relationship depth and query flexibility are critical. As user behavior increasingly manifests in multi-layered and temporal patterns, adopting graph-based models becomes essential for building accurate, real-time analytics pipelines.

Graph query strategies for user behavior analysis

Graph databases offer powerful querying mechanisms that go beyond traditional filtering and aggregation. User behavior analysis often requires tracing interactions across multiple degrees of separation-such as identifying chains of content consumption, influence paths in social networks, or anomalous navigation patterns. Graph query languages like Cypher (used in Neo4j) and Gremlin (used in TinkerPop-based systems) provide native support for recursive traversals, subgraph pattern matching, and graph algorithms-all of which are essential for behavioral insights.

One of the most common techniques is path traversal, used to identify the sequence of actions taken by a user or to discover how different users are connected through shared interactions [2]. For example, detecting a community of users who consistently follow similar purchase paths or identifying influence chains in referral-based ecosystems. Graph databases can execute such queries in linear time relative to path depth, while in relational systems this often results in multiple nested joins and exponential growth in execution time.

Another frequent approach is the use of centrality metrics (e.g., PageRank, betweenness, closeness) to identify the most influential users or sessions within a network. In behavioral contexts, high-centrality nodes may represent key navigational hubs, referral generators, or fraudulent actors. Similarly, community detection algorithms (e.g., Louvain, label propagation) help cluster users into behaviorally similar groups for segmentation or targeting.

To highlight the performance advantage of graph databases for multi-hop queries, the figure below compares the average query response times in a synthetic dataset as the relationship depth increases from 1 to 6 hops. As shown in figure 1, while graph-based traversal scales linearly, relational joins exhibit exponential degradation.

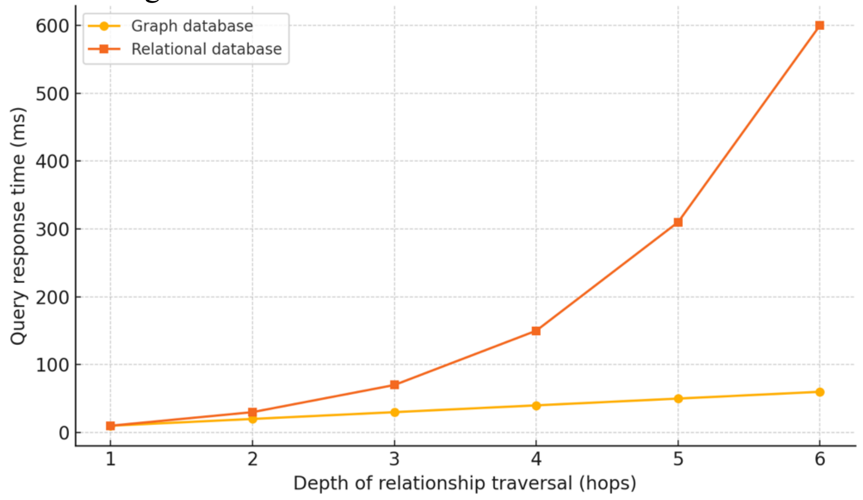

Figure 1. Query response time by depth of relationship traversal in graph vs. relational database models

The results illustrated in Figure clearly demonstrate the scalability advantage of graph databases for multi-hop relationship queries. As the traversal depth increases, the response time in relational systems grows exponentially due to the compounding cost of join operations. In contrast, graph databases maintain near-linear performance, enabling efficient exploration of deeply connected user behavior patterns. This characteristic makes graph-based models particularly suitable for real-time behavioral analytics, where low-latency insights across complex interaction chains are required.

Graph algorithms for extracting behavioral patterns

Graph databases support a wide range of algorithms that can reveal latent behavioral structures not easily accessible through traditional data analysis techniques. These algorithms enable analysts to uncover user clusters, detect anomalies, evaluate influence, and optimize recommendation strategies based on structural properties of the user interaction graph [3].

One widely used category is community detection algorithms, such as the Louvain method or label propagation. These help to identify groups of users who exhibit similar behavioral trajectories-visiting similar sequences of pages, reacting to the same content, or purchasing related products. Such clustering is valuable for audience segmentation, personalization, and targeted marketing.

Another critical class is centrality algorithms, including PageRank, degree centrality, betweenness, and closeness. These metrics quantify the importance of nodes within the network. In behavioral contexts, central nodes may correspond to super-users, content hubs, or actors involved in suspicious activity propagation.

Similarity and proximity algorithms, such as Jaccard similarity or personalized PageRank, can identify users with shared interests or patterns, supporting collaborative filtering and social recommendations. For anomaly detection, graph outlier detection methods identify structurally rare patterns, such as unexpected edge density or users disconnected from typical interaction flows.

The table 2 below outlines common categories of graph algorithms, their primary goals, and examples of behavioral analysis use cases.

Table 2

Graph algorithms for user behavior analysis

Algorithm type	Primary goal	Example use cases
Community detection	Cluster users with similar behavior	Segmenting audiences, recommending group-based content
Centrality metrics	Identify influential nodes	Detecting super-users, fraud hubs, or key referrers
Similarity/proximity	Find structurally similar users	Collaborative filtering, social recommendations
Pathfinding/traversal	Discover behavioral chains	Navigation flow analysis, content funnel optimization
Outlier detection	Identify anomalous user patterns	Fraud detection, bot activity identification

As summarized in table, graph algorithms provide a versatile analytical toolkit for modeling and interpreting complex user behavior patterns. Each category of algorithms serves a distinct purpose-ranging from identifying communities and influential users to detecting anomalies and reconstructing behavioral paths. The ability to apply these algorithms directly within graph databases enables real-time, relationship-aware analytics that traditional systems often struggle to achieve. By selecting appropriate algorithms aligned with specific behavioral objectives, analysts can uncover hidden structures, personalize user experiences, and improve decision-making in dynamic digital environments.

Implementing behavior analysis using Cypher queries

Cypher, the declarative query language for Neo4j and other property graph databases, enables analysts to describe complex patterns of user behavior in a concise and expressive way. Rather than relying on cumbersome SQL joins, Cypher operates natively on node–relationship structures, making it particularly effective for analyzing sequences, cycles, and multi-hop interactions.

In behavioral analytics, Cypher is often used to implement three major classes of queries: path-based queries, structural ranking, and pattern discovery. Path-based queries are frequently employed to trace navigation flows, user conversion funnels, or repeated access loops [4]. Structural ranking is useful for identifying high-impact users, such as referrers or hubs in content networks, while pattern discovery is used for detecting suspicious behavior, churn indicators, or anomalous interaction graphs.

The flexibility of Cypher allows analysts to combine conditions on both node attributes (e.g., user age, session duration) and relationship properties (e.g., frequency, time intervals), enabling fine-grained filtering of behavior. This makes it a powerful tool for segmenting user groups based on interaction types, recency, or cross-platform activity.

To better understand the practical use of Cypher in behavior analysis, figure 2 presents the relative frequency of different query types across a sample of real-world graph-based analytics workloads. The data illustrates that path queries dominate most use cases, followed by influence/ranking queries, with anomaly detection and temporal pattern matching growing steadily in adoption.

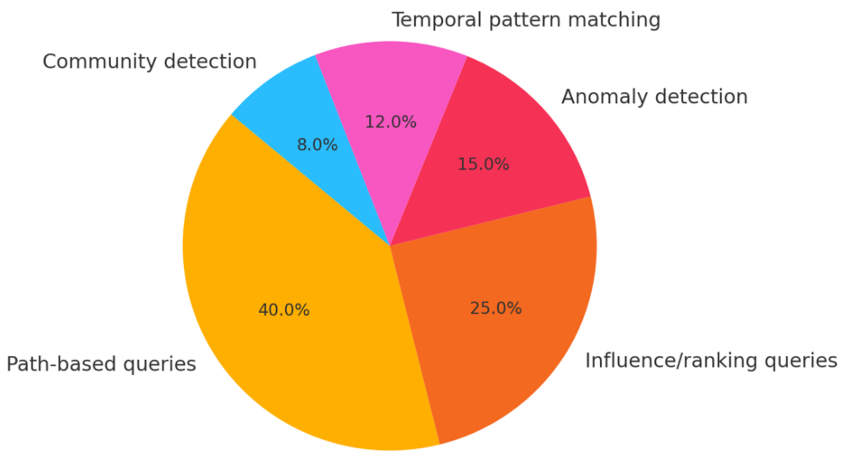

Figure 2. Distribution of Cypher query types in user behavior analysis use cases

As shown in figure, path-based queries constitute the largest share of Cypher-based behavior analysis workloads, reflecting their central role in tracing user navigation, conversion paths, and repeated interaction sequences. Influence and ranking queries are also widely adopted, particularly in use cases involving social graphs, referral systems, and recommendation optimization. The growing use of anomaly detection and temporal pattern matching indicates a shift toward more predictive and adaptive analytical approaches. This distribution highlights the flexibility of Cypher in supporting both descriptive and advanced behavioral modeling tasks within graph databases.

Integrating graph databases into behavior analytics architectures

While GDBs provide a powerful foundation for modeling and analyzing complex user interactions, their effective use in production requires seamless integration into broader data processing and analytics pipelines. In contemporary data-driven environments, behavioral data is typically generated across multiple systems-web logs, mobile applications, customer relationship management (CRM) platforms-and must be aggregated, normalized, and enriched before meaningful analysis can be performed. Integrating GDBs into this landscape demands careful orchestration of data flows, performance tuning, and alignment with organizational data governance policies.

One of the most common integration patterns involves extract–transform–load (ETL) pipelines that move behavioral event data from transactional sources into the graph store. Tools such as Apache NiFi, Kafka Connect, and Neo4j’s own Data Importer enable automated ingestion of session logs, interaction events, and user profiles. Preprocessing steps may include timestamp normalization, deduplication, and enrichment with metadata (e.g., device type, location, user segment). A consistent data model must be designed to accommodate both entity diversity and evolving relationship schemas [5].

Once ingested, graph databases often operate alongside other analytical systems. Hybrid architectures may combine GDBs for relationship modeling with columnar databases (e.g., ClickHouse, BigQuery) for high-speed aggregations or with document stores (e.g., MongoDB) for flexible session metadata storage. Business intelligence (BI) platforms can connect to GDBs via Cypher or GraphQL interfaces, while machine learning (ML) workflows increasingly incorporate graph embeddings and topological features derived from GDBs to improve model accuracy [6].

Despite these advantages, integration also poses challenges. Maintaining data consistency across systems, ensuring low-latency synchronization, and managing schema evolution in dynamic environments require ongoing architectural and operational effort. Additionally, access control, auditing, and compliance constraints must be enforced across all interconnected components, especially in industries like finance and healthcare.

Nonetheless, when properly integrated, graph databases significantly enhance the depth and contextual quality of behavioral analytics, enabling systems that go beyond static segmentation toward truly relational, adaptive, and intelligent user modeling.

Performance comparison of graph and relational databases in behavior analytics

Selecting the appropriate database architecture for behavior analytics is not merely a matter of data modeling preferences-it directly impacts system scalability, query latency, and analytical flexibility [7]. To evaluate this, a series of benchmark tests were conducted using representative behavior analysis tasks, including multi-hop traversal, user influence detection, and session pattern extraction. Both relational and graph database platforms were tested on equivalent datasets with controlled dimensions.

The results consistently indicate that graph databases outperform relational systems in queries involving deep relationship chains and structural computations. In contrast, relational databases maintain an advantage in flat aggregations and predefined tabular reporting. The performance gap widens with query complexity, especially as the number of joins in SQL increases beyond three or four hops.

Figure 3 presents the execution time (in milliseconds) for five common behavior analysis tasks executed in both relational and graph-based implementations. These include tasks such as detecting returning user loops, calculating node centrality, and reconstructing behavioral paths. The data clearly demonstrate the scalability advantage of graph systems as relationship complexity grows.

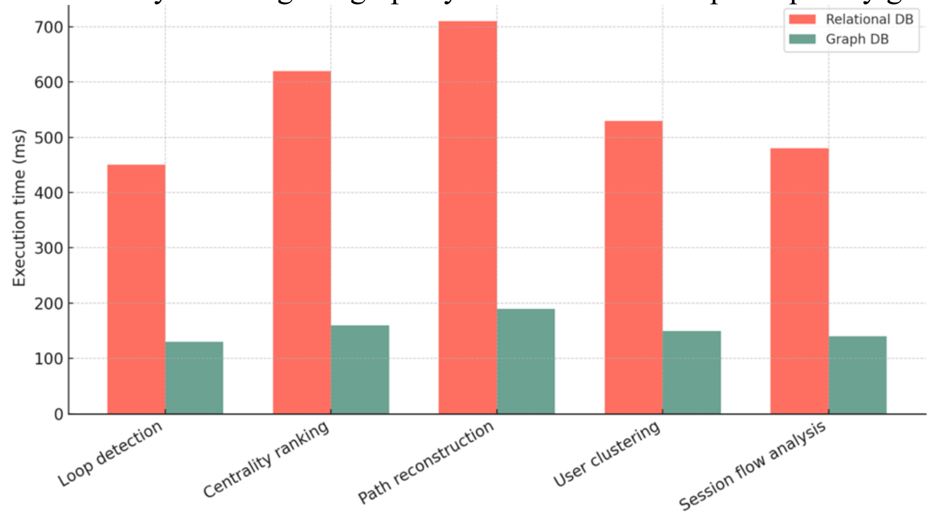

Figure 3. Execution time comparison of graph and relational databases on behavior analysis tasks

As illustrated in figure, graph databases demonstrate significantly lower execution times across all tested behavior analysis tasks compared to relational databases. The performance advantage becomes more pronounced for structurally complex operations such as path reconstruction and centrality ranking, where the relational model incurs substantial overhead due to join-based traversal. This trend underscores the scalability and efficiency of graph-native querying, particularly in scenarios involving multi-hop relationships and dynamic user interactions [8]. These results support the adoption of graph databases as a more performant solution for behavior-centric analytical workloads.

Applications of graph-based behavior analysis in real-world systems

The practical applications of graph-based behavior analysis extend far beyond academic experimentation. In commercial and operational environments, graph databases are increasingly employed to power mission-critical systems that rely on understanding user intent, context, and interaction history.

One key application domain is fraud detection, where relationship structures often reveal collusive behavior or anomalous transaction flows. Graph models enable the identification of subtle patterns, such as shared devices across multiple accounts or coordinated login sequences, which would remain undetected in flat data representations.

Another major area is personalized recommendation systems, where user-to-user or user-to-content graphs are leveraged to model affinities and co-engagement. Unlike traditional collaborative filtering based on matrix factorization, graph-based methods capture contextual dependencies and multi-step relationships (e.g., user → item → tag → user), enhancing recommendation accuracy and explainability.

Customer journey mapping is also increasingly powered by graph analytics. By tracing paths through digital touchpoints-websites, support interactions, app sessions-organizations can optimize experience design and reduce churn. Graph traversals help reconstruct nonlinear, multi-session behavior that conventional systems struggle to capture.

Finally, in the cybersecurity domain, graph analytics are applied to user activity graphs to detect lateral movement, privilege escalation, and access anomalies, particularly in enterprise environments with zero-trust policies. These use cases validate the robustness and adaptability of graph-based approaches in behavior-centric decision-making systems.

Limitations and challenges of graph databases in behavior analysis

Despite their advantages, graph databases are not a universal solution. Their adoption in behavior analytics comes with a range of limitations that must be considered when designing real-world systems.

One key challenge is scalability under high-volume write operations. While graph traversal is efficient for reads, ingesting large-scale behavioral logs in real time may require careful tuning, partitioning, or even polyglot persistence approaches where ingestion is offloaded to stream processors. Another issue is tooling maturity and standardization. While SQL is universally supported and optimized across decades, graph query languages like Cypher, Gremlin, and GSQL still lack full interoperability and may involve vendor lock-in. This can impact long-term maintainability and ecosystem integration.

Query optimization in graph databases also requires graph-specific expertise. Traversals that seem intuitive can become inefficient without appropriate indexing, cardinality management, or query rewriting. Performance tuning demands an understanding of graph topology, data distribution, and storage backend characteristics. Moreover, cost modeling and capacity planning are less predictable in graph systems, particularly under highly dynamic workloads. Horizontal scaling strategies like sharding remain more complex in graph databases due to their inherent reliance on relationship locality.

Lastly, compliance and auditability pose additional concerns. Unlike relational systems with mature logging and rollback mechanisms, ensuring consistent governance in GDB environments requires custom implementation, especially when dealing with sensitive behavioral data under GDPR or CCPA frameworks. Figure 4 provides a visual overview of the relative weight of these limitations, based on expert evaluation and literature trends. The prominence of write scalability and tooling immaturity underscores the importance of architectural readiness and operational planning when integrating graph solutions into analytics workflows.

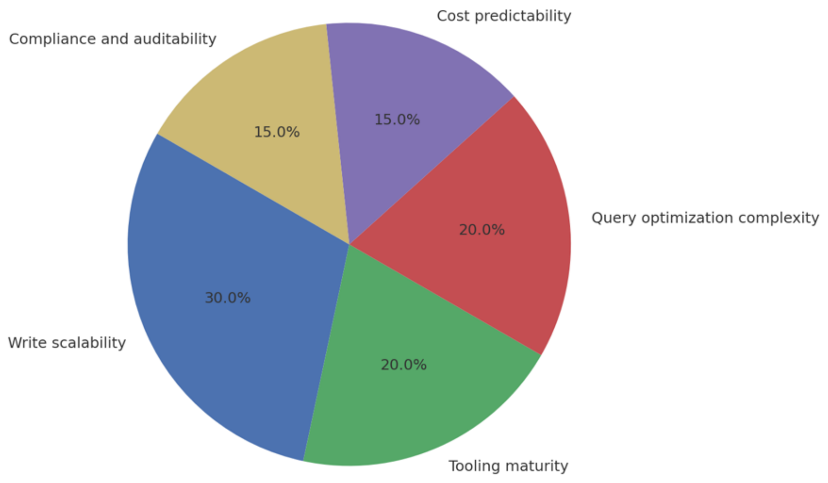

Figure 4. Major limitations of graph databases in behavior analysis

The analysis presented in this section highlights that, while graph databases offer substantial analytical advantages, their adoption introduces a distinct set of architectural and operational challenges. As shown in figure, write scalability and tooling maturity represent the most prominent concerns, especially in high-throughput or enterprise-grade deployments. Issues related to query optimization, cost predictability, and compliance further underscore the need for specialized expertise and robust infrastructure planning. These findings emphasize that graph databases should not be viewed as drop-in replacements for traditional systems, but rather as complementary technologies that require deliberate integration and governance strategies to deliver sustained value in user behavior analytics [9].

Future directions in graph-based behavior analytics

As digital ecosystems continue to grow in complexity and scale, graph-based approaches to behavior analysis are poised to play an increasingly central role in intelligent decision-making. Emerging technological and methodological trends suggest several key directions for the evolution of this field.

One prominent development is the convergence of graph databases and machine learning. Techniques such as graph embeddings, graph neural networks (GNNs), and link prediction models are enabling systems to move beyond explicit querying into predictive and prescriptive analytics. This shift allows for more adaptive personalization, real-time fraud anticipation, and autonomous user segmentation-particularly when graph representations are integrated into ML pipelines.

Another promising area is the adoption of temporal and dynamic graph models, which extend static graphs with time-aware semantics. In behavior analysis, user actions are inherently temporal and context-dependent. Dynamic graphs allow analysts to capture evolving relationships, detect behavioral shifts over time, and correlate events across sessions, devices, or platforms.

Standardization of graph query languages is also likely to influence adoption. Initiatives such as GQL (Graph Query Language) aim to unify diverse graph querying approaches (e.g., Cypher, Gremlin, SPARQL) under a common standard, improving interoperability and reducing vendor lock-in. This evolution will lower the barrier to entry and foster the integration of GDBs into mainstream data platforms [10].

Lastly, graph-based systems are expected to become more tightly integrated with cloud-native infrastructures. Serverless graph engines, streaming-compatible ingestion pipelines, and managed GDB services will make it easier to deploy scalable, real-time behavior analytics at lower operational cost.

These trajectories suggest that graph-based behavior analytics is not only a viable tool for current challenges, but a foundational technology for the next generation of user-centric, intelligent systems. Figure 5 presents a strategic outlook on key technological trends, showing how impact and estimated adoption timelines vary across graph-centric innovations in analytics.

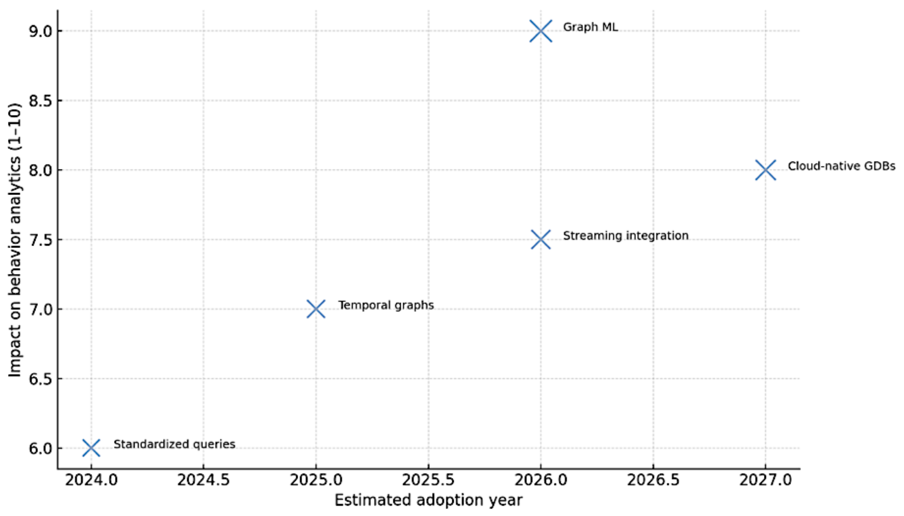

Figure 5. Strategic projection of key trends in graph-based behavior analysis

The visualization in figure highlights the rising importance of Graph ML and cloud-native architectures, with both expected to exert high impact on behavioral analytics within the next few years. Temporal modeling and streaming integration are also gaining traction as organizations seek real-time, context-aware insights [11]. The relatively earlier maturation of standardized query languages suggests that ecosystem interoperability will play a foundational role in enabling these advanced capabilities. Collectively, these trends point toward an increasingly intelligent, adaptive, and event-driven analytics paradigm grounded in graph technologies.

Conclusion

The use of graph databases for user behavior analysis represents a significant advancement in the modeling and interpretation of complex, relationship-driven data. Unlike traditional relational systems, graph databases enable native traversal of interconnected behavioral entities, allowing for more intuitive and efficient querying of user paths, influence networks, and temporal patterns. This advantage is particularly evident in multi-hop queries and dynamic interaction scenarios, where relational models face performance and modeling limitations.

Throughout the study, key aspects of graph-based behavior analytics were examined, including modeling strategies, query techniques, graph algorithm applications, integration into analytical pipelines, and performance comparisons. The empirical results and visualizations confirm that graph databases consistently outperform relational systems in structurally complex analytical tasks, providing both scalability and analytical depth. At the same time, several practical limitations-such as write scalability, query complexity, and compliance concerns-highlight the need for careful architectural planning and operational maturity.

Looking ahead, the convergence of graph technologies with machine learning, temporal modeling, and cloud-native infrastructure suggests a promising future for behavior analytics. Graph-based systems are well positioned to become a core component of next-generation intelligent platforms, supporting adaptive, real-time, and context-aware decision-making across industries. The findings presented in this paper contribute to a deeper understanding of how graph databases can be leveraged to extract value from behavioral data and support the design of resilient, user-centric analytics architectures.

References

1. Guia J., Soares V. G., Bernardino J. Graph databases: Neo4j analysis //ICEIS (1). 2017. P. 351-356.
2. Almabdy S. Comparative analysis of relational and graph databases for social networks // 2018 1st International conference on computer applications & information security (ICCAIS). IEEE. 2018. P. 1-4.
3. Beutel A. User behavior modeling with large-scale graph analysis // Computer Science Department, Carnegie Mellon University. 2016.

4. Muramudalige S.R., Hung B.W., Jayasumana A.P., Ray I. Investigative graph search using graph databases // 2019 first international conference on graph computing (gc). IEEE. 2019. P. 60-67.

5. Shang F., Ding Q., Du R., Cao M., Chen H. Construction and application of the user behavior knowledge graph in software platforms // Journal of Web Engineering. 2021. Vol. 20. No. 2. P. 387-411.

6. Larriba-Pey J.L., Martínez-Bazán N., Domínguez-Sal D. Introduction to graph databases // Reasoning Web International Summer School. Cham: Springer International Publishing. 2014. P. 171-194.

7. Mozannar H., Bansal G., Fourney A., Horvitz E. Reading between the lines: Modeling user behavior and costs in AI-assisted programming // Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems. 2024. P. 1-16.

8. Mendu M., Krishna B., Mahesh G., Pallavi J. Development of real time data analytics based web applications using NoSQL databases // AIP Conference Proceedings. AIP Publishing LLC. 2022. Vol. 2418. No. 1. P. 020038.

9. Teng S., Khong K.W. Examining actual consumer usage of E-wallet: A case study of big data analytics // Computers in Human Behavior. 2021. Vol. 121. P. 106778.

10. Kejriwal M. Knowledge graphs: A practical review of the research landscape // Information. 2022. Vol. 13. No. 4. P. 161.

11. Zhong L., Wu J., Li Q., Peng H., Wu X. A comprehensive survey on automatic knowledge graph construction // ACM Computing Surveys. 2023. Vol. 56. No. 4. P. 1-62.