BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en

Vol. 31 No. 2. 2026

UDC 681.3:511.1
DOI: 10.62660/bcstu/2.2026.85

Application of linear algebra methods for optimising data processing and storage in modern information systems

Olexandr Kraychuk*
PhD in Physical and Mathematical Sciences, Professor
Rivne State University for the Humanities
33028, 12 Stepana Bandery Str., Rivne, Ukraine
https://orcid.org/0000-0002-5987-0460

Serhii Kraychuk
PhD inTechnical Sciences, Associate Professor
Rivne State University for the Humanities
33028, 12 Stepana Bandery Str., Rivne, Ukraine
https://orcid.org/0000-0001-9756-1979

Nataliia Ostapchuk
PhD in Pedagogical Sciences, Associate Professor
Rivne State University for the Humanities
33028, 12 Stepana Bandery Str., Rivne, Ukraine
https://orcid.org/0000-0001-8827-8741

Abstract. The aim of the study was to generalise approaches to coordinating decisions at the stages of planning, execution, and physical level of data representation in information systems under the dominance of resource constraints when operating with massive sets. The methodology was based on system-analytical formalisation, comparative-analytical generalisation and typology, comparative analysis, systematisation and integrative synthesis. It was established that linear algebra formalises the representation of data as matrices/vectors and calculations as a sequence of transformations. The generalisation of operations (semirings) extends these primitives to graph and Boolean problems, and Graph Basic Linear Algebra Subprograms standardises the implementation for sparse structures. Efficiency is determined by the consistent chain of "rewrite→cost model→physical execution": rewrite reduces intermediate data, and the cost model chooses a plan based on data input/output, memory, sparsity, and parallelism, without it, heuristics remain. Factorised representations have been found to reduce redundancy and pressure on memory and data input/output (for joins/groups) with system support. Randomised Numerical Linear Algebra and sketching reduce costs and manage the "accuracy-resources" trade-off in streaming processing. Matrix compression (low-rank representation, reduced precision) reduces memory and traffic and speeds up computations with a controlled loss of precision. In vector database management systems and distributed systems, the bottleneck is data input/output and redistribution between nodes, so DShuffle moves some processing closer to the data, emphasising the role of infrastructure optimisation. The linear algebraic representation of computations provides a basis for consistent optimisation of rewriting, plan selection, and execution, reducing materialisation and data movement under input/output, memory, and bandwidth constraints. The practical significance of the results lies in the possibility of the application by information-systems developers and engineers for the consistent selection of data formats, execution plans, and computational primitives to reduce materialisation and I/O, memory, and network overhead

Keywords: operations; matrices; resource constraints; sparsity; materialisation; databases

Article's History: Received: 29.12.2025; Revised: 25.03.2026; Accepted: 18.05.2026; Published: 26.06.2026

Suggested Citation:
Kraychuk, O., Kraychuk, S., & Ostapchuk, N. (2026). Application of linear algebra methods for optimising data processing and storage in modern information systems. *Bulletin of Cherkasy State Technological University*, 31(2), 85-96. doi: 10.62660/bcstu/2.2026.85.

*Corresponding author (o.kraychuk@outlook.com)

INTRODUCTION

Modern information systems of the 21st century work with large amounts of data in various storage formats (data lakes, lake-house) and perform complex analytical workloads, some of which are reduced to linear algebra operations (matrix and vector multiplication, system solution, factorisation, gradient and aggregate calculations). The performance of such tasks is determined not only by calculations, but also by the costs of data movement, the choice of physical representation (row/column, dense/sparse) and the ability of the optimiser to take into account the algebraic properties of operations. Analytical and Machine Learning (ML) calculations are performed in different engines and on different data formats, which leads to conversions, materialisation of intermediate results and additional input/output (I/O) costs. Because of this, local optimisations of individual components do not provide stable acceleration at the level of the entire pipeline, therefore, a coordinated approach is needed that combines the linear-algebraic representation of calculations with the choice of storage format and optimisation of execution plans.

The current scientific discourse outlines several interrelated directions, in particular trade-offs between data formats in analytical database management systems. The study by C. Liu *et al.* (2025) summarised that the physical format determines the cost profile for scanning, decoding, compression, vectorisation, and predicate access, and that there is no "universal" choice – optimality depends on the queries and the hardware environment. This approach underpins the argument that processing optimisation begins with the correct physical representation. The work of F. Geerts *et al.* (2021) substantiated that linear algebraic query languages can express a wide range of calculations typical of analytics and machine learning problems, and formally outlines which operators and extensions are needed to represent key algorithmic schemes. This allows for setting limits on the expressive power of such languages, which directly defines the class of transformations that the optimiser can apply correctly and systematically, rather than as heuristics. Continuing this trend, T.M. Serrano *et al.* (2024) showed that for conjunctive queries in linear algebraic formalism, the efficiency of execution depends on which language fragments allow for low-latency result enumeration and support for cost-effective updates. Working with dynamic data requires not a one-time "quick fix", but a structured approach to query representation and supporting data structures that provide stable performance as input sets change.

There is growing interest in factorised and polyglot approaches in machine learning pipelines, as significant costs arise from materialising intermediate results and re-passing through data. Z. Li *et al.* (2024) found that factorised representations reduce the amount of intermediate data and speed up typical operations, provided that the system preserves factorisation between stages and makes it consistent across languages/engines. This indicates that critical losses occur at the boundaries between formats, operators, and runtimes, not just within a single engine. At the storage level for a data lake, a study by G. Jin *et al.* (2022) highlighted that performance is determined not only by the columnar format, but also by adaptive caching and physical transformations of the data for repetitive read patterns. For linear algebraic workloads, this is critical because the same features/matrix representations are often reused multiple times in different analysis stages. At the hardware-oriented level, P.A. Lane & J.D. Booth (2023) showed that Sparse Matrix-Vector Multiplication (SpMV) is a memory-limited computational kernel and requires tuning of the storage format and execution parameters taking into account the sparsity structure of the matrix and the microarchitecture of the device. This means that universal optimisations do not guarantee portable performance across platforms.

M. Boehm *et al.* (2023) demonstrated that the efficiency of tensor and linear algebraic computations in data systems is determined by automatic transformations at the compilation and execution levels, in particular, representation selection, computational graph rewriting, and memory management. This demonstrates that the alignment of linear algebra with optimisation mechanisms of data processing systems allows for reduced overhead and increased performance without changing the application logic. In the context of Ukrainian research, O.V. Hordiichuk-Bublivska & L.P. Fabri (2022) substantiated the feasibility of matrix factorisation for big data problems in industrial systems as a way to reduce dimensionality and obtain a compact representation. Factorisation can simultaneously reduce storage requirements and reduce the cost of subsequent analytical calculations. S.P. Striamets & Yu.V. Opotyak (2021) focused on an architectural approach to parallel matrix processing, proposing a homogeneous computing environment with a specialised computational "cell" for performing matrix operations. This means that such hardware-oriented solutions increase the throughput and predictability of linear algebraic routines, which is critical for data processing systems.

Despite significant progress in research on storage formats, linear algebraic query languages, factorised representations, and optimisation of individual routines, a holistic approach that reconciles these levels into a single optimisation loop is lacking. It remains insufficiently studied how the linear algebraic representation of calculations can coordinate the choice of data format, materialisation strategy, planning and implementation of basic operations taking into account dominant resource constraints (I/O, memory, bandwidth). Therefore, the purpose of the study was to systematise approaches to coordinating planning, execution and physical representation of data in information systems

taking into account dominant resource constraints. To achieve the goal, the following tasks were set: to generalise the representation of data and analytical procedures in terms of vector-matrix objects and basic linear algebraic primitives, to substantiate the framework of coordinated optimisation through the sequence "rewriting rules→cost model→physical operators/execution strategies", to systematise tools for reducing the resource cost of linear algebraic calculations according to the impact on memory and data input/output costs.

MATERIALS AND METHODS

The study combined the methods of system-analytical formalisation, system-structural analysis and synthesis, comparative-analytical generalisation and typology, comparative analysis, systematisation and integrative synthesis to generalise approaches to optimising Linear Algebra (LA)-oriented workloads in modern information systems. Using the method of system-analytical formalisation with elements of cost-based interpretation, heterogeneous data and calculations were compared and conceptually reduced to a matrix-vector representation and a set of standard LA primitives, the space of alternative implementations and the principles of the correct rewriting were determined and correlated with the execution level through the standardised Graph Basic Linear Algebra Subprograms (GraphBLAS) interfaces (GraphBLAS Forum, n.d.) and the practice of LA query processing in systems (Sun *et al.*, 2025). This was done in order to describe optimisation not as separate heuristics, but as a consistent mechanism "expression→rewriting/plan→cost selection→physical execution" taking into account sparsity, materialisation of intermediate results, memory locality, parallelism, and input/output costs. Using the method of system-structural analysis and synthesis of scientific sources, approaches to optimisation of LA expressions and execution plans were generalised in the form of an integrated plan control loop consisting of three interdependent levels – rewriting rules, cost model and physical operators/execution strategies by role in optimisation, LA specificity and typical risks in the absence of support. These levels were chosen because of corresponding to three necessary components of any planned optimisation: generation of equivalent alternatives, the cost ranking and selection, operationalisation of the selected plan at the execution level. This approach made it possible to formalise the necessary mechanisms that ensure stable optimisation of LA workloads in the system.

Using the method of comparative-analytical generalisation and typology of scientific sources, Randomised Numerical Linear Algebra (RNLA) (Udell & Frangella, 2023), sketching properties (Ahfock *et al.*, 2023) and sliding windows (Yin *et al.*, 2024) were compared, selected as components describing the goal (optimisation), guarantees (quality) and operating mode (online/dynamics). The comparison was performed according to four criteria: conceptual formulation (principle of using randomness/stability/incrementality), approximation object (matrices, operators, subspaces/spectral properties, sliding window sketch), system consequences (impact on Central Processing Unit (CPU)/memory/I/O costs, update latency, quality degradation control), typical application mode (batch analytics, critical paths, streaming processing). This allowed comparing the types of RNLA/sketching with system scenarios and justify the choice of approach according to the operating mode and the trade-off "resource economy↔quality control". Using the comparative analysis method, structural (low-rank) and structural-numerical (low-rank+low-precision) compressions were analysed: weighted low-rank factorisation (for compression of language models) and randomised low-rank factorisation with quantisation (low precision) according to the following criteria: compression principle, quality control and system effect (memory footprint, memory traffic, potential acceleration of multiplications). The choice of low-rank and low-precision is due to the fact that these approaches represent the two basic and most widely used classes of compression (structural and numerical), which directly affect the amount of parameters, memory traffic and performance, providing a manageable trade-off "quality-resources". This approach provided a choice between low-rank and low-precision compression, linking compression parameters to the quality-resources trade-off and the expected system effect. Additionally, DShuffle (Ding *et al.*, 2025) is presented as a representative example of infrastructure optimisation, selected based on the criteria of relevance to the shuffle bottleneck in distributed LA workloads and the presence of a measurable system effect (traffic, bandwidth, CPU load). In this study, DShuffle is used only to illustrate the system-level resource-result effect, without a detailed comparison of alternative infrastructure solutions.

Using the method of systematisation and integrative synthesis of scientific sources, the directions for optimising LA-oriented loads in information systems are generalised – optimisation of the execution of analytical/ML queries in database management systems, approximate matrix methods for accelerating calculations, streaming processing (sliding window), matrix/model compression, physical representation of big data, vector databases, and similarity search. These directions were chosen because of representing the main "points of influence" in LA pipelines: reduction of intermediate materialisations, controlled trade-off "accuracy-resources", reduction of data volume and traffic, and coordination of plans with the execution and infrastructure levels. The generalisation is performed according to unified criteria: direction of application, dominant resource constraint, method/class of methods, principle of action, target metrics (time, bandwidth, I/O, memory, accuracy/error), limitations, and risks. This was done in order to systematise LA-oriented workload optimisation tools for selection under specific resource

constraints and quality/performance requirements, linking methods to effect metrics and application risks.

RESULTS AND DISCUSSION

Linear algebraic optimisation of queries and computational plans

Linear algebra provides a formal apparatus for representing data as vectors and matrices and describing computations as compositions of transformations. Tabular, feature, graph, and network data can be treated as dense or sparse matrix objects, and analytical procedures can be reduced to a limited set of primitives: matrix-vector and matrix-matrix multiplications, element-wise operations, reductions, and permutations/projections. This type of unification is the basis for system optimisation, as it allows expressing heterogeneous ML/data science workloads in a common algebraic language and treating queries as sequences of LA operators (Sun *et al.*, 2025). The existence of equivalent implementations of many transformations (in particular, due to different multiplication orders and different materialisation decisions) creates a basis for a systematic search for efficient execution strategies, which is consistent with enumeration-oriented optimisation of LA workloads (Muñoz, 2025). The extension of the semantics of operations to generalised algebraic structures (in particular, semirings) makes the same primitives applicable to graph and Boolean problems, and GraphBLAS fixes a standardised interface for performing such operations on (mostly sparse) matrices and vectors (Brock *et al.*, 2021).

Representing computations in the form of LA expressions makes algebraic properties (associativity, distributivity, equivalence of factorisations) explicit and thereby enables correct rewrites to reduce execution costs: changing the order and grouping of operations, choosing alternative implementations, and controlling the materialisation of intermediate results. Since the LA representation is relatively declarative, it supports cost-based selection of physical operators and execution strategies, including the use of sparsity, blocking, and (where applicable) approximations. A critical system-level effect is also the reduction of intermediate data: in many analytical tasks, it is the materialisation of large intermediate matrices/vectors that determines the pressure on memory and I/O, and optimisations such as fusion, operation permutations, and sparsity exploitation reduce the number of intermediate structures and the total cost (Sun *et al.*, 2025; Muñoz, 2025).

The practical effectiveness of LA optimisations is determined not only by the possibility of rewriting expressions and planning, but also by how accurately and with minimal overhead the generated plans are reproduced at the execution level, taking into account sparsity, locality of memory access and parallelism. In this context, standardised APIs play a pivotal role, due to specifying a consistent set of primitives and ensuring portability of optimisations between implementations. A prime example is GraphBLAS (GraphBLAS Forum, n.d.), which formalises the interface of operations on matrices and vectors (multiplication, element-wise transformations, reductions, operations for constructing/modifying structures) and supports parameterisation of semantics through generalised algebraic constructs such as semirings (Brock *et al.*, 2021). For information systems, this means the presence of an actual "target platform" for executing LA plans: the optimiser generates plans in terms of standard operations, and the execution level implements those plans in a specific hardware-software stack. A standardised set of primitives also unifies the space of alternatives and increases the reproducibility of optimisation solutions, since algorithmic ideas formulated at the level of LA expressions do not require re-designing low-level operators in each environment (Brock *et al.*, 2021; Sun *et al.*, 2025).

In practical ML pipelines, calculations corresponding to sequences of LA operators are performed in a fragmented manner across database management systems and external libraries/services, which leads to excessive materialisation of intermediate results, data duplication, and movement overhead. The LA query processing approach consists in presenting the ML query as a single LA expression for optimisation at the expression level: through equivalent algebraic transformations (reordering and regrouping of operations), merging adjacent transformations, and selecting physical implementations taking into account data properties (in particular, sparsity), which reduces dominant costs – data movement, temporary structures, and repeated reads (Sun *et al.*, 2025). Data science workloads are characterised by a multiplicity of equivalent implementations of the same LA expression, so performance is determined by the global structure of the plan. The enumeration approach forms the space of alternatives and selects the best strategy according to cost criteria, taking into account the order of calculations, the degree of materialisation, implementation options for basic operations, and structural properties of the data (Muñoz, 2025).

The transfer of ML and analytics into information systems, in turn, requires support for new classes of operators and rules that are not fully covered by the traditional relational apparatus. LA computations are characterised by specific equivalence spaces, where algebraic rewrites, the choice of operator implementations, and control over the materialisation of intermediate results are decisive. Accordingly, the optimiser must be extensible: it must allow the introduction of new logical operators, rewriting rules, cost components, and physical implementations without radically rebuilding the optimisation kernel. In the literature, this is considered an engineering mechanism for controlled extension while maintaining the correctness and predictability of optimisation (Ding *et al.*, 2024). In this sense, extensibility is a basic prerequisite for system support for both LA query processing for ML queries and planned search for alternatives for LA workloads

in data science. Given this, the practical integration of LA logic into the optimiser requires not only conceptual support for new expressions, but also mechanisms for controlled rewrites, cost-based selection of plans, and executable implementation of alternatives. In general, these mechanisms can be reduced to three levels: rewriting rules, cost model, and physical operators/execution strategies, as presented in Table 1.

Table 1. Structure of the integrated computational plan control loop

Extensibility level	Key role	Specifics for LA workloads	Typical risks without support
Rewriting rules	Allow equivalent transformations of expressions	Regrouping/reordering, fusion, less intermediate data	Rewriting cycles, unmanaged alternative space, lack of domain-specific optimisations
Cost model	Compares alternative plans by "price"	Choosing a plan considering I/O, memory, sparsity, parallelism	Degradation to heuristics, incorrect comparisons of plans, "optimisation" without the predicted effect
Physical operators and execution strategies	Implement logic in efficient execution	Algorithmic options, data structure specialisations, efficient execution strategies	The inability to operationalise alternatives at the execution level leads to the lack of a measurable optimisation effect

Source: compiled by the authors based on B. Ding *et al.* (2024), W. Sun *et al.* (2025), T. Muñoz (2025)

Table 1 summarises the mechanisms through which the optimiser is extensible for LA workloads and shows these mechanisms as an interdependent chain from logical transformations to efficient execution. Rewriting rules specify semantically equivalent transformations of expressions and create the basis for optimisation: such rules allow changing the order and grouping of operations, applying fusion, and reducing the amount of intermediate data. Without the controlled application of rules, rewriting cycles and uncontrolled expansion of the space of alternatives are possible. The cost model provides a choice among the alternatives generated by rewrites, taking into account the dominant costs (I/O, memory, sparsity, locality of access, parallelism). In the absence of an adequate cost model, optimisation degrades to heuristics or gives unstable results due to incorrect plan comparisons. Physical operators and execution strategies convert logical alternatives into measurable performance gains through algorithmic options, data structure specialisations, and appropriate execution strategies. If this level is not supported, the alternatives remain formal and do not give a practical effect. Therefore, the extensibility of the optimiser for LA workloads is effective only if there is a consistent triad: the rules shape the space of alternatives, the cost model provides a reasonable choice, and the physical layer guarantees an executable gain. Weakening any component reduces the efficiency of the others and limits the practical value of the optimisation. Since the extensible optimiser provides controlled rewriting, cost-effective selection, and executable implementation of alternatives, it is a natural "loop" for implementing approximate LA methods as part of system optimisation. In this context, RNLA and matrix sketching are of interest, which allow reducing computational complexity and resource consumption with a controlled error. The main characteristics of these approaches and typical application modes are summarised in Table 2.

Table 2. Characteristics of RNLA and matrix sketching in information systems

Approach/ component	Conceptual presentation	Approximation object	Systemic consequences	Typical application mode
RNLA for optimisation	Controlled use of randomness for approximate calculations with controlled error	Large Matrices and Linear Operators (Sketch Representation)	Reducing computational and resource costs (CPU/memory/I/O) with acceptable accuracy	Batch analytics and optimisation problems on big data
Sketching properties	Probabilistic guarantees of approximation quality and stability regimes	Norms, subspaces, spectral characteristics	Justification of parameterisation and control of the risk of quality degradation	Integrating approximations into critical processing/ decision-making loops
Sliding windows	Incremental maintenance of approximation when updating the data "window"	Sliding window matrix object sketch	Limiting in-memory state and reducing update latency	Streaming (online) analytics and streaming systems

Source: compiled by the authors based on D. Ahfock *et al.* (2023), M. Udell & Z. Frangella (2023), H. Yin *et al.* (2024)

Table 2 summarises three complementary directions of application of approximate LA methods in information systems: RNLA as a way to reduce the resource cost of calculations, theoretical properties of sketching as the basis of reliability, and sliding-window sketching as a mechanism for working in streaming modes. The common basis is the replacement of operations on full-dimensional matrix objects with calculations on compact representations with a controlled error. RNLA is focused on acceleration and savings of CPU/memory/I/O through sketch-representation of matrices and operators in batch analytics problems. Analysis of sketching properties

provides probabilistic quality guarantees (for norms, subspaces and spectral characteristics), which supports parameter selection and control of degradation risk. The sliding-window approach provides incremental sketch updates when the data "window" changes, limiting the state in memory and reducing the latency of updates in streaming systems. The system value of RNLA/sketching is determined by a combination of three components: practical cost reduction, availability of quality guarantees and adaptation to the processing mode (batch or streaming). In short, approximate LA methods are not only mathematical techniques, but also an engineering scaling tool that allows balancing the accuracy of results with memory limitations, I/O and execution time requirements. Along with approximate calculations through RNLA and sketching, a critical direction for optimising resources in systems is the compression of the matrix representations – both data and model parameters. Compression of matrices and models in information systems comes down to two complementary mechanisms: low-rank, when a matrix object is approximated by a lower internal dimension, and low-precision, when the bit depth of the element representation is reduced. Both approaches reduce memory and bandwidth requirements, and can also speed up matrix operations with a controlled loss of accuracy (Table 3).

Table 3. Comparative characteristics of low-rank and low-precision matrix and model compression approaches

Approach	Compression principle	Quality management	Main system-level effect
Weighted low-rank factorisation (for compression of language models)	Low-rank approximation of parametric matrices	Weights in the loss function specify uneven approximation accuracy (component prioritisation)	Reducing the amount of parameters and potentially speeding up multiplications with controlled loss of quality
Randomised low-rank factorisation with quantisation (low precision)	Combination of structural (rank) and numerical (precision) compression	The rank and bitness parameters determine the "accuracy-resources" tradeoff, criticality is the total consideration of errors	Reduced memory footprint and memory traffic + possible acceleration of calculations on low-precision hardware

Source: compiled by the authors based on Y.-C. Hsu *et al.* (2022), R. Saha *et al.* (2023)

Weighted low-rank factorisation is focused on problem-specific quality preservation: the weights in the loss function allow for differentially distributing the approximation error, providing more accurate reproduction of critical components and more "aggressive" compression of less significant ones. Randomised low-rank and low-precision factorisation combines two independent compression mechanisms – structural (rank reduction) and numerical (bit size reduction), which directly reduces the storage volume of factors and traffic between memory levels, and can also increase computational efficiency on hardware optimised for low precision. At the same time, the combination of low-rank and low-precision increases the requirements for total error control, since the contributions of rank approximation and quantisation accumulate. In general, factorisation compression can be constructed as an adaptive precision control (through weights) to preserve quality in the target problem, or as a system-oriented combined compression (low-rank+low-precision) aimed at minimising memory footprint and memory access costs. Both approaches implement the "precision-resources" trade-off, but do so with different "leverages": weighted low-rank factorisation – through component prioritisation, randomised low-rank factorisation with quantisation – through simultaneous limitation of the internal dimensionality and bitness of the representation. This emphasises the role of linear algebra as a tool for engineering optimisation, where factorisation parameters serve as a means of tuning performance to specific system constraints.

The results of the study showed that the effectiveness of LA optimisations in sparse computing is determined not only by the correctness of algebraic rewrites, but also by the execution mode, in particular, the way of materialising intermediate results and organising memory access. This is consistent with the conclusions of A. Mastoras *et al.* (2023), who found that the transition to nonblocking semantics in GraphBLAS makes deferred execution and fusion of sequences of operations possible, which reduces the volume of intermediate structures and changes the memory access profile. The authors also showed that the performance gain is parameter-dependent: it is determined by the configuration of parallelism and block sizes, and can degrade in the event of an unsuccessful choice due to the overhead of execution management. The approach taken in both studies concretises the "bottleneck" between the optimiser and execution: algebraically equivalent plans require execution mechanisms that are able to transform rewrites and planning into a real reduction in memory/I/O pressure and obtain measurable speedup.

The results of the study showed that structural low-rank compression of tensor representations reduces memory and storage requirements at the cost of a controlled loss of accuracy, and the effect is determined by the approximation parameters. This is consistent with the study by T. Kwon *et al.* (2025), who proposed TensorCodec based on neural tensor-train decomposition and showed that the compression quality is further improved by folding into a higher-order tensor and reordering modes to better detect structural dependencies. The authors experimentally demonstrated an improvement in the "compression degree – reconstruction error" trade-off compared to the basic alternatives, which

confirms the parameterised nature of tensor compression: resource savings are achieved not "for free", but through a controlled choice of the compression level, which directly specifies the permissible quality degradation. A sustainable acceleration effect is achieved only with a coordinated optimisation of plans, execution primitives and infrastructure that removes network and I/O constraints.

In the study by Z. Shen *et al.* (2025), the researchers found that the use of matrix operations in storage systems (via erasure coding) is key to overcoming resource constraints while ensuring fault tolerance. The authors emphasised that the evolution of coding methods is aimed at minimising the I/O overhead during data recovery, which requires deep integration of LA primitives into the physical layer of the system. This correlates with the results of this study on the role of linear algebra as an engineering optimisation tool, where the choice of storage format and materialisation strategy determines the I/O profile of the entire system. Both approaches emphasise that a sustainable speedup effect is achieved only with a coordinated optimisation of plans, execution primitives, and infrastructure that removes network and I/O constraints. Thus, linear algebra unifies data representation and analytical/ML procedures in the form of LA expressions, which allows for correct rewrites and cost-effective plan selection with minimal intermediate materialisation. The practical effect is determined by the consistency of rewriting rules, cost model, and physical implementation, which sets real trade-offs between performance, memory/I/O, and accuracy.

Optimisation of storage, access, and execution of LA computations

Matrix representations are natural in big data problems, but matrices are mostly sparse, so dense storage incurs unnecessary memory and I/O overhead and slows down computations by handling zero elements. Sparse formats store only non-zero values and indices, reducing data size and the cost of passes. Critically, the choice of sparse format affects not only compactness but also the cost of LA operations: formats optimise row/column access, updates, and parallelisation differently, determining which operations will be relatively "cheap" or "expensive" in LA pipelines (Marichal *et al.*, 2021). Columnar storage is fundamental to analytical systems, as queries selectively read attributes and perform scans with filters and aggregations, with performance determined by the speed of reading the required columns, the overhead of encoding/decompression, and cache management, i.e., the storage format as a key factor in the I/O profile. For LA workloads, this means that feature preparation and data transformation often boil down to intensive scans, where I/O and decoding can dominate arithmetic. Accordingly, the throughput of reading/processing columns determines the cost of building matrix representations from tabular data (Zeng *et al.*, 2023).

Unlike sparse and columnar, which optimise physical storage at the element or column level, factorised databases reduce redundancy at the result-structure level. In queries with joins and groupings, factorised representations compactly encode dependencies and common subexpressions instead of fully materialising all combinations, which potentially reduces the size of intermediate results, memory pressure, and I/O, provided that the optimiser and execution layer support it (Olteanu, 2023). In the context of LA workloads, factorised representations are a complementary level of compactness alongside sparse (zero minimisation) and columnar (efficient attribute scans): together, these approaches emphasise the multi-level nature of storage optimisation and its impact on the performance of LA-oriented plans (Marichal *et al.*, 2021; Olteanu, 2023). The spread of embeddings has formed a separate class of vector database management systems focused on storing high-dimensional vectors and similarity queries (Nearest Neighbours (NN)/Approximate Nearest Neighbour (ANN)) or hybrid queries with metadata filters. The performance is determined not only by the search algorithms, but also by the storage architecture, indexes, query execution framework and resource management. A typical architecture includes storage subsystems (vector/metadata placement), indexing (ANN and the "accuracy-speed" trade-off), execution (scheduling, filters, similarity calculations) and background support (reindexing, compaction), with the computational foundation directly relying on linear algebra operations (Pan *et al.*, 2024).

In practical vector search, the bottleneck is not similarity calculations, but I/O and the cost of data access. In storage-based ANNs, vectors and indices do not fit entirely in RAM, so performance is determined by storage characteristics, access patterns, and caching. Under such conditions, ANN should be treated as a system optimisation problem that requires coordinated co-design of indices, physical representation, compression/quantisation, and execution strategies; local speedup of one component may not yield any benefit without coordination with others. Thus, vector search is a class of workloads that combine LA computations, compression, and access optimisation, and the result is determined by the system consistency of these components (Pan *et al.*, 2023; Pan *et al.*, 2024). The effectiveness of LA methods depends not only on the optimiser, but also on the execution primitives. Standardised programming interfaces specify a unified set of operations and create a basis for low-level optimisations (parallelisation, vectorisation, and sparsity). GraphBLAS formalises matrix/vector operations and parameterises semantics via semirings, acting as a "bridge" between LA representation and productive execution; standardisation also increases portability of optimisations across implementations (Brock *et al.*, 2021).

In distributed systems, performance is limited not by arithmetic but by shuffling data between nodes, so

LA optimisations can lose the effectiveness if infrastructure costs dominate. DShuffle is an example of an infrastructure optimisation of data redistribution in large-scale processing, where the bottleneck is the internode exchange of intermediate data and the overhead of the central processor. The approach is based on the use of a data processing unit (DPU) to offload some operations, which reduces the load on the CPU and improves system performance (traffic, throughput) in distributed scenarios (Ding *et al.*, 2025). Together, GraphBLAS and DPU-oriented shuffle illustrate the principle of co-design: sustained speedup is achieved by concerted optimisation of LA plans, execution primitives, and infrastructure that removes network and I/O constraints. Based on the approaches considered, a generalisation of linear algebra approaches is presented in Table 4 by application areas, types of resource constraints, metrics, and risks.

Table 4. Generalisation of linear algebra approaches for optimising data processing and storage in modern 21st century information systems

Direction of application	Resource constraint type	Method (linear algebra/related)	Principle of operation	Target metrics	Limitations and risks
Optimisation of analytical/ML query execution in database management systems	Materialisation of intermediate results, ineffective plans	Linear algebraic query processing, expression rewriting, dynamic scheduling	Equivalent expression transformations, operator fusion; intermediate data reduction	Response time, bandwidth, read/write (I/O), RAM	Cost model, integration complexity
Approximate matrix methods for accelerating calculations	CPU, sometimes RAM	RNLA, randomised sketching	Random projections/sampling, dimensionality reduction	Computation time, RAM, error	Accuracy-speed tradeoff, parameter selection, dependence on data properties
Streaming processing (sliding window)	CPU for calculation, RAM	Sketching matrices on a sliding window	Incremental sketch updates	Response time, bandwidth; RAM, error	Error control, dependence on window parameters, complexity of implementation
Matrix/Model Compression	RAM/video memory, storage, memory bandwidth	Low-rank (weighted) factorisation, low-rank+low-precision factorisation	Approximation with fewer parameters, quantisation	Size, memory usage, speed, error	Quality deterioration, accumulation of errors, validation, and calibration required
Physical representation of big data	I/O, RAM	Sparse storage (compressed rows/columns), columnar formats	Storing non-zeros, reading the required columns, compression	I/O, volume, query time	Expensive upgrades, possible CPU bottleneck shift (decompression)
Vector databases and similarity search	Mostly I/O, indexes, storage	Vector database management systems, I/O-aware ANNs	Indexing; reading reduction, candidate selection	Response time, I/O, volume, accuracy/completeness	Accuracy-speed tradeoff, vector distribution dependency, I/O dominance

Notes: Materialisation (materialisation of intermediate results) – execution of a query operator with physical formation and temporary storage of its output data set (e.g., in a temporary table or file) for further processing by subsequent operators
Source: compiled by the authors based on R. Marichal *et al.* (2021), Y.-C. Hsu *et al.* (2022), R. Saha *et al.* (2023), D. Ahfock *et al.* (2023), M. Udell & Z. Frangella (2023), X. Zeng *et al.* (2023), B. Ding *et al.* (2024), H. Yin *et al.* (2024), J.J. Pan *et al.* (2024), W. Sun *et al.* (2025), T. Muñoz (2025), Z. Ren *et al.* (2025)

LA-oriented optimisation in information systems is reduced to managing the interrelated trade-offs between performance and correctness of the result. Approximate approaches (RNLA, sketching, low-rank, low-precision) provide a reduction in time and memory costs, but require controlled parameterisation and quality validation procedures, especially in incremental update modes. At the same time, in practical workloads, the determining factors are not arithmetic operations, but the costs of data access and movement (I/O, caches, exchange between nodes), which is manifested in storage-based vector search configurations. Therefore, a stable optimisation effect is achieved only by coordinating optimisations at the expression/plan level with the cost model and physical implementations of operators, otherwise the optimisation remains partial and vulnerable to system bottlenecks.

Given the set of results considered, the choice of LA methods and system mechanisms for the implementation should be made taking into account the dominant resource constraints and the type of workload. For ML queries and data science pipelines, the algebraic representation of calculations in the form of LA expressions is a priority, since it supports rewriting and cost-based selection of plans, reducing the materialisation of intermediate results and integration overhead. Under the conditions of operator evolution and the emergence of domain-specific rules, the extensibility of the optimiser (rewriting rules, cost model, physical implementations) is decisive, without which the optimisation effect becomes unstable. When the execution profile is compute-bound and approximation is permissible, RNLA and matrix sketching are applicable, and in streaming modes, sliding-window sketch maintenance mechanisms are used to maintain quality during incremental updates. When memory and bandwidth constraints dominate, low-rank and low-precision factorisations are appropriate, which reduce the footprint and potentially speed up multiplication while controlling the total error. In terms of storage, the format should be consistent with the operations: sparse and columnar representations determine the I/O profile and decoding overhead, and factorised approaches reduce the structural redundancy of materialisations. For vector search, the co-design of indexing, execution, and storage is critical, particularly in storage-based modes, where the scaling limit is set by I/O characteristics. In general, the effectiveness of LA optimisations depends on the "vertical" consistency of planning, execution primitives (in particular GraphBLAS), and, in distributed scenarios, the infrastructure costs of the network and shuffle.

The results of the study confirmed that in practical vector search workloads, the main bottleneck is I/O costs and data access costs, which directly determine the scaling limits for storage-based systems. The revealed pattern justifies the need for coordinated co-design of indexes, physical representation, and execution strategies as a key means of overcoming infrastructure limitations. This approach is consistent with the findings of Y. Pan *et al*. (2023), who showed that to ensure scalability on large data volumes, the transition to disk-oriented structures with low RAM usage (LM-DiskANN) is critical. The authors argued that optimising graph indexes directly for disk-specificity allows maintaining high performance of ANN queries even under strict RAM limitations. Within the framework of system logic, both works confirm that the real efficiency of vector systems is determined not only by the chosen algorithm, but also by the architectural integrity with the storage, where it is I/O characteristics that become the determining factor of global performance.

X. Bao *et al*. (2024) found that integrating machine learning models directly into the logical rules of a data cleansing system (Rock) reduces the overhead associated with the gap between declarative semantics and procedural ML components. The authors showed that "embedded" execution of ML models within the logical plan avoids external calls, intermediate materialisation, and data duplication, which directly affects performance and predictability of execution. These results are consistent with current research on the need for an algebraic representation of ML calculations as part of the optimisation core of the system, rather than as external black boxes. In particular, the authors' approach illustrates that the extensibility of the optimiser through the introduction of new rules and operators is a critical prerequisite for effective support of LA workloads in data science, since it is at the level of algebraic representation that consistent rewriting, planning, and cost-effective execution selection become possible. At the same time, the approach of X. Bao *et al*. has an applied focus on the data cleaning problem and does not consider the general space of linear algebraic rewrites and cost-based optimisation for a broader class of ML and analytical workloads. In contrast to the system framework formulated in this study, the authors' work is limited to analysing the impact of physical operators and execution parameters (memory, parallelism, I/O) on the global cost of the plan. Integrating ML into the declarative core is feasible, but requires LA-oriented optimisation at the logical, cost, and physical levels.

B. Tian *et al*. (2024) found that offloading part of the ANN search computations to SmartSSDs (solid-state storage) allows for scaling queries to billions of volumes while reducing latency and increasing energy efficiency. The authors showed that in storage-based ANNs, the bottleneck is not the similarity calculation itself, but the data transfer between the storage and the host, so near-storage computing reduces the pressure on the transmission bus and reduces host overhead. The effectiveness of this approach depends on the redistribution of functions between the device and the host (organisation of accesses, candidate selection, execution of some operators on the SSD side), i.e., on the architectural consistency of the execution loop. This correlates with the results of this study on infrastructure optimisation (DPU/DShuffle) as a way to remove network and I/O constraints in distributed scenarios: the acceleration is achieved not by local optimisation of a single operator, but by moving critical processing stages closer to the data and reducing the amount of inter-component traffic. Thus, the obtained results confirm the principle of co-design: a sustainable performance gain is formed under the condition of coordination of the execution plan (including LA operators that implement selection/aggregation/similarity calculation) with the available hardware primitives and the limitations on bandwidth and data exchange.

The results of the study showed that the key factor in the effectiveness of LA optimisations is the "vertical" consistency between the planning process and specific physical execution strategies, especially in conditions of a shortage of fast memory. Minimising the materialisation

of intermediate results by choosing optimal execution plans allows increasing system performance and reducing resource load. This conclusion is supported by the results of the work of Z. Zhang *et al.* (2024), where it was demonstrated that the implementation of ordered pipeline processing allows efficiently executing vector queries, even if the amount of data exceeds the available video memory of the GPU. The authors confirmed that careful control of the order of operations in combination with overlapping calculations by data transfer allows successfully levelling out materialisation delays. The acceleration effect discovered by Z. Zhang *et al.* serves as a practical validation of the approach proposed in this study to the systematic minimisation of intermediate data through the consistency of the planning and execution levels. Thus, the effectiveness of LA-oriented optimisation is determined by the "vertical" consistency of planning with physical execution and the ability to minimise the materialisation of intermediate results; a sustainable effect is achieved only with the operation of a consistent chain "rewriting→cost model→physical execution".

CONCLUSIONS

The results showed that matrix representations are natural for large datasets, but the prevailing sparsity makes dense storage a source of excessive memory and I/O overhead and slowdown due to processing of zero elements. Sparse storage formats reduce the size of the data by storing only non-zero values and indices, reducing the cost of passes. The choice of sparse format affects not only compactness, but also the "price" of LA operations due to different efficiencies of row/column accesses, updates, and parallelisation. For analytical systems, columnar storage determines the I/O profile of queries, and performance depends on the speed of reading the required columns, the overhead of encoding/decompression, and the behaviour of caches. In LA workloads, feature preparation and transformations often boil down to intensive scans, where I/O and decoding can dominate arithmetic. Factorised databases reduce redundancy at the result-structure level by reducing the amount of intermediate materialisations in joins and groupings, provided that such representations are supported by the optimiser and execution layer.

The spread of embeddings has formed a class of vector database management systems in which efficiency is determined not only by the ANN algorithm, but also by the storage architecture, indexes, execution path, and resource management. In storage-based ANNs, the bottleneck is usually not the similarity computation, but I/O and the cost of accessing data when vectors and indexes do not fit entirely in RAM. I/O dominance requires a coordinated co-design of indexes, physical representation, compression/quantisation, and execution strategies, and local speedups of one component may not yield any benefit without coordination with the others. In distributed scenarios, the effect of LA optimisations is limited by the costs of data movement (shuffle), so sustainable acceleration requires simultaneous coordination of LA plans with execution primitives and infrastructure mechanisms for removing network and I/O constraints. The limitation of the study is its theoretical nature, without experimental validation on a single prototype of the information system. A promising direction for future research is the development and experimental validation of an LA-oriented optimisation loop (rewriting + cost model + physical operators), coordinated with storage-based ANN and hardware primitives (DPU/near-storage/SmartSSD) and supplemented by compression and RNLA/sketching methods with formal quality guarantees and automatic parameter selection.

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Ahfock, D., Astle, W.J., & Richardson, S. (2023). On randomized sketching algorithms and the Tracy-Widom law. *Statistics and Computing*, 33, article number 34. doi: 10.1007/s11222-022-10148-5.

[2] Bao, X., *et al.* (2024). Rock: Cleaning data by embedding ML in logic rules. In *Companion of the 2024 international conference on management of data* (pp. 106-119). New York: Association for Computing Machinery. doi: 10.1145/3626246.3653372.

[3] Boehm, M., Interlandi, M., & Jermaine, C. (2023). Optimizing tensor computations: From applications to compilation and runtime techniques. In *Companion of the 2023 international conference on management of data* (pp. 53-59). New York: Association for Computing Machinery. doi: 10.1145/3555041.358940.

[4] Brock, B., Buluç, A., Mattson, T.G., McMillan, S., & Moreira, J. (2021). *The GraphBLAS C API specification: Version 2.0.0*. Retrieved from https://graphblas.org/docs/GraphBLAS_API_C_v2.0.0.pdf.

[5] Ding, B., Narasayya, V., & Chaudhuri, S. (2024). Extensible query optimizers in practice. *Foundations and Trends in Databases*, 14(3-4), 186-402. doi: 10.1561/1900000077.

[6] Ding, C., Li, S., Lu, K., Yao, T., Wang, D., Wu, H., Wan, J., Tan, Z., & Xie, C. (2025). DShuffle: DPU-optimized shuffle framework for large-scale data processing. In *2025 USENIX annual technical conference* (pp. 1371-1386). Boston: USENIX Association.

[7] Geerts, F., Muñoz, T., Riveros, C., & Vrgoč, D. (2021). Expressive power of linear algebra query languages. In *Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems* (pp. 342-354). New York: Association for Computing Machinery. doi: 10.1145/3452021.3458314.
[8] GraphBLAS Forum. (n.d.). Retrieved from https://graphblas.org/.
[9] Hordiichuk-Bublivska, O.V., & Fabri, L.P. (2022). Matrix factorization of big data in the industrial systems. *Ukrainian Journal of Information Technology*, 4(2), 68-73. doi: 10.23939/ujit2022.02.068.
[10] Hsu, Y.-C., Hua, T., Chang, S.-E., Lou, Q., Shen, Y., & Jin, H. (2022). Language model compression with weighted low-rank factorization. *ArXiv*. doi: 10.48550/arXiv.2207.00112.
[11] Jin, G., Bian, H., Chen, Y., & Du, X. (2022). Columnar storage optimization and caching for data lakes. In *Proceedings of the 25th international conference on extending database technology* (pp. 419-423). Edinburgh: Open Proceedings. doi: 10.48786/edbt.2022.33.
[12] Kwon, T., Ko, J., Jung, J., Jang, J.-G., & Shin, K. (2025). Compact lossy compression of tensors via neural tensor-train decomposition. *Knowledge and Information Systems*, 67, 1169-1211. doi: 10.1007/s10115-024-02252-x.
[13] Lane, P.A., & Booth, J.D. (2023). Heterogeneous sparse matrix-vector multiplication via compressed sparse row format. *Parallel Computing*, 115, article number 102997. doi: 10.1016/j.parco.2023.102997.
[14] Li, Z., Sun, W., Zhan, D., Kang, Y., Chen, L., Bozzon, A., & Hai, R. (2024). Amalur: The convergence of data integration and machine learning. *IEEE Transactions on Knowledge and Data Engineering*, 36(12), 7353-7367. doi: 10.1109/TKDE.2024.3357389.
[15] Liu, C., Pavlenko, A., Interlandi, M., & Haynes, B. (2025). Data formats in analytical DBMSs: Performance trade-offs and future directions. *The VLDB Journal*, 34, article number 30. doi: 10.1007/s00778-025-00911-1.
[16] Marichal, R., Dufrechou, E., & Ezzatti, P. (2021). Optimizing sparse matrix storage for the big data era. In M. Naiouf, E. Rucci, F. Chichizola & L. De Giusti (Eds.), *9th conference: Cloud computing, big data & emerging topics* (pp. 121-135). Cham: Springer. doi: 10.1007/978-3-030-84825-5_9.
[17] Mastoras, A., Anagnostidis, S., & Yzelman, A.-J.N. (2023). Design and implementation for nonblocking execution in GraphBLAS: Tradeoffs and performance. *ACM Transactions on Architecture and Code Optimization*, 20(1), article number 6. doi: 10.1145/3561652.
[18] Muñoz, T. (2025). Enumeration-based dynamic query processing for linear algebra workloads in data science. In *VLDB 2025: PhD workshop*. London: QEII Centre.
[19] Olteanu, D. (2023). Report on the workshop on factorized databases. *SIGMOD Record*, 52(2), 53-56.
[20] Pan, J.J., Wang, J., & Li, G. (2024). Survey of vector database management systems. *The VLDB Journal*, 33, 1591-1615. doi: 10.1007/s00778-024-00864-x.
[21] Pan, Y., Sun, J., & Yu, H. (2023). LM-DiskANN: Low memory footprint in disk-native dynamic graph-based ANN indexing. In *2023 IEEE international conference on big data* (pp. 5987-5996). Sorrento: IEEE. doi: 10.1109/BigData59044.2023.10386517.
[22] Ren, Z., Doekemeijer, K., Apparao, P., & Trivedi, A. (2025). Storage-based approximate nearest neighbor search: What are the performance, cost, and I/O characteristics? In *2025 IEEE international symposium on workload characterization* (pp. 407-422). Irvine: IEEE. doi: 10.1109/IISWC66894.2025.00041.
[23] Saha, R., Srivastava, V., & Pilanci, M. (2023). Matrix compression via randomized low rank and low precision factorization. In *Proceedings of the 37th international conference on neural information processing systems* (pp. 18828-18872). Red Hook: Curran Associates Inc.
[24] Serrano, T.M., Riveros, C., & Vansummeren, S. (2024). Enumeration and updates for conjunctive linear algebra queries through expressibility. In *Proceedings of the 27th international conference on database theory (ICDT 2024)*. Wadern: Dagstuhl Publishing. doi: 10.4230/LIPIcs.ICDT.2024.12.
[25] Shen, Z., Cai, Y., Cheng, K., Lee, P.P.C., Li, X., Hu, Y., & Shu, J. (2025). A survey of the past, present, and future of erasure coding for storage systems. *ACM Transactions on Storage*, 21(1), article number 4. doi: 10.1145/3708994.
[26] Striamets, S.P., & Opotyak, Yu.V. (2021). Matrix parallel processor based on a homogeneous computational medium using an advanced computing cell. *Ukrainian Journal of Information Technology*, 3(1), 78-84. doi: 10.23939/ujit2021.03.078.
[27] Sun, W., Katsifodimos, A., & Hai, R. (2025). Accelerating machine learning queries with linear algebra query processing. *Distributed and Parallel Databases*, 43, article number 8. doi: 10.1007/s10619-024-07451-7.
[28] Tian, B., Liu, H., Duan, Z., Liao, X., Jin, H., & Zhang, Y. (2024). Scalable billion-point approximate nearest neighbor search using SmartSSDs. In *Proceedings of the 2024 USENIX annual technical conference* (pp. 1135-1150). San Francisco: USENIX Association.
[29] Udell, M., & Frangella, Z. (2023). *Randomized numerical linear algebra for optimization*. Retrieved from https://web.stanford.edu/~udell/doc/udell2022_randnla4opt.pdf.

[30] Yin, H., Wen, D., Li, J., Wei, Z., Zhang, X., Huang, Z., & Li, F. (2024). Optimal matrix sketching over sliding Windows. *Proceedings of the VLDB Endowment*, 17(9), 2149-2161. doi: 10.14778/3665844.3665847.
[31] Zeng, X., Hui, Y., Shen, J., Pavlo, A., McKinney, W., & Zhang, H. (2023). An empirical evaluation of columnar storage formats. *Proceedings of the VLDB Endowment*, 17(2), 148-161. doi: 10.14778/3626292.3626298.
[32] Zhang, Z., Liu, F., Huang, G., Liu, X., & Jin, X. (2024). Fast vector query processing for large datasets beyond GPU memory with reordered pipelining. In *Proceedings of the 21st USENIX symposium on networked systems design and implementation* (pp. 23-40). Santa Clara: USENIX Association.

Застосування методів лінійної алгебри для оптимізації обробки та зберігання даних у сучасних інформаційних системах

Олександр Крайчук
Кандидат фізико-математичних наук, професор
Рівненський державний гуманітарний університет
33028, вул. Степана Бандери, 12, м. Рівне, Україна
https://orcid.org/0000-0002-5987-0460
Сергій Крайчук
Кандидат технічних наук, доцент
Рівненський державний гуманітарний університет
33028, вул. Степана Бандери, 12, м. Рівне, Україна
https://orcid.org/0000-0001-9756-1979
Наталія Остапчук
Кандидат педагогічних наук, доцент
Рівненський державний гуманітарний університет
33028, вул. Степана Бандери, 12, м. Рівне, Україна
https://orcid.org/0000-0001-8827-8741

Анотація. Метою дослідження було узагальнити підходи до узгодження рішень на етапах планування, виконання та фізичного рівня подання даних в інформаційних системах за домінування ресурсних обмежень під час оперування масивними наборами. Методологія базувалася на системно-аналітичній формалізації, порівняльно-аналітичному узагальненні та типологізації, порівнянному аналізі, систематизації та інтегративному синтезі. Встановлено, що лінійна алгебра формалізує подання даних як матриць/векторів і обчислення як послідовність перетворень. Узагальнення операцій (напівкільця) поширює ці примітиви на графові та булеві задачі, а Graph Basic Linear Algebra Subprograms стандартизує їх виконання для розріджених структур. Ефективність визначається узгодженим ланцюгом "переписування → вартісна модель → фізичне" виконання: переписування зменшує проміжні дані, а вартісна модель обирає план за введення/виведення даних, пам'яттю, розрідженістю та паралелізмом, без неї лишаються евристики. Виявлено, що факторизовані подання зменшують надлишковість і тиск на пам'ять та введення/виведення даних (для приєднань/групувань) за підтримки системою. Randomized Numerical Linear Algebra і скетчинг знижують витрати та керують компромісом "точність-ресурси" в потоковій обробці. Компресія матриць (низькорангове подання, знижена точність) зменшує пам'ять і трафік та прискорює обчислення за контрольованої втрати точності. У векторних системах керування базами даних і розподілених системах вузьким місцем є введення/виведення даних і перерозподіл між вузлами, тому DShuffle переносить частину обробки ближче до даних, підкреслюючи роль інфраструктурної оптимізації. Лінійно-алгебраїчне подання обчислень створює основу для узгодженої оптимізації переписування, вибір планів і виконання, зменшуючи матеріалізацію та переміщення даних за обмежень введення/виведення, пам'яті й пропускної здатності. Практична значимість результатів полягає у можливості їх застосування розробниками та інженерами інформаційних систем для узгодженого вибору форматів даних, планів виконання й обчислювальних примітивів, щоб зменшити матеріалізацію та витрати на введення/виведення, пам'ять і мережевий обмін

Ключові слова: операції; матриці; ресурсні обмеження; розрідженність; матеріалізація; бази даних