UDC 004.65:004.658.2:004.89

METHOD OF COMBINED INDEXING FOR EFFECTIVE SEARCH IN MULTI-ATTRIBUTE CATALOGS OF TECHNICAL COMPONENTS

Sotnik S. V. – PhD, Associate Professor, Associate Professor of Department of Computer-Integrated Technologies, Automation and Robotics, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine. ROR: https://ror.org/01ctj1b90. ORCID: https://orcid.org/0000-0002-6035-2388.

ABSTRACT

Context. Optimizing search in multidimensional catalogs of radio-electronic components (sensors, microcontrollers, communication modules) is a crucial task for computer-aided design systems, electronic component logistics management, and intelligent technical support systems. The complexity arises due to the high dimensionality of the parameter space (operating frequencies, power consumption, temperature ranges, etc.), data heterogeneity, and the high frequency of complex queries combining numerical constraints with categorical filters. Classic indexing algorithms from relational database management systems are inefficient for this specific domain, which slows down the performance of real-time information systems.

Modern indexing technologies for searching multi-attribute catalogs of technical components are moving away from the paradigm of universal one-dimensional structures in favor of specialized and hybrid approaches tailored to the nature of technical data. The focus has shifted from fast single-key search to efficient pruning of multidimensional parameter space. To achieve this, spatial indexes are actively used, interpreting each component as an object in an N-dimensional space, where each technical parameter is a separate axis. This allows a single query to the index to find all entries that fall within a specified multidimensional hyperrectangle.

Simultaneously, technologies that treat search as an information retrieval problem are evolving. Categorical attributes, such as interface type or manufacturer, are indexed using inverted indexes or compressed bitmap indexes, which provide ultra-fast execution of AND/OR operations over large sets.

A separate direction involves the use of vector representations (embeddings) of technical characteristics, obtained using machine learning models, followed by indexing using specialized structures for nearest neighbor search. This enables semantic search by technical description or finding analogous components.

A key issue in indexing for search within multi-attribute catalogs of technical components is the selection and combination of data structures that effectively prune the search space across all relevant dimensions simultaneously, minimizing the retrieval of irrelevant data in the early stages of query execution.

Therefore, the development of a new method that systematically combines the strengths of modern approaches within a unified adaptive architecture is a relevant scientific and technical problem. Its solution will significantly reduce the execution time of complex queries in key information systems for the fields of radio electronics, telecommunications, and automated engineering, addressing the challenges of production digitalization and intelligent data processing.

Objective. Development of a combined indexing method for efficient execution of complex search queries in multi-dimensional catalogs of technical components.

Method. A combined indexing method is proposed, which combines R-tree for multidimensional filtering of numerical parameters and inverted indexes for categorical features. The method for improving search efficiency is based on an adaptive query planner that dynamically chooses the optimal execution strategy (Index-First, Parallel-Merge or Full-Scan) based on an assessment of the selectivity of the conditions.

Results. The problem was formulated, and a combined indexing method was developed for multidimensional catalogs of radioelectronic components. In the course of the study, a formalized mathematical search model was created that takes into account the selectivity of numerical and categorical conditions, and algorithms for automatic distribution of attributes between index types and dynamic selection of the query execution strategy were developed. A hybrid index architecture was proposed that combines an R-tree for indexing numerical parameters and inverted indexes for categorical features, as well as mathematical models for assessing selectivity and optimizing the query execution plan. Experimental studies on a synthetic data set confirmed the effectiveness of the developed method, demonstrating a reduction in the execution time of complex queries compared to basic indexing based on B-trees.

Conclusions. The work develops a combined indexing method for efficient execution of complex multi-attribute queries in radioelectronic component catalogs. The method is based on a formalized mathematical search model, which allowed building a hybrid index architecture. This architecture integrates an R-tree for filtering numerical parameters, inverted indexes for categorical features, and an adaptive planner that dynamically selects the optimal query execution strategy (Index-First, Parallel-Merge, Full-Scan) based on an assessment of the selectivity of conditions. An algorithm for the automatic distribution of attributes between index types is developed. Experimental modeling confirmed the effectiveness of the method, showing a 35–55% reduction in the execution time of complex queries compared to traditional B-tree-based indexing, especially for queries typical of engineering component selection. The results obtained prove the feasibility of using combined indexing to increase the productivity of information systems working with multidimensional technical catalogs.

KEYWORDS: multi-attribute search, combined indexing, radio electronic components, query optimization, hybrid data structure.

ABBREVIATIONS

CAD – computer-aided design system;
DBMS – database management systems;
DDL – Data Definition Language;
DS – data sampling;
MBR – Minimum Bounding Rectangles;
PLM – Product Lifecycle Management;
R-MLGTI – R-Multi-Level Grid-Tree Index;
SPI – Serial Peripheral Interface.

DOI 10.15588/1607-3274-2026-2-18

NOMENCLATURE

A^{cat} – set of categorical attributes;

A^{num} – set of numeric attributes;

b – number of intervals (buckets);

c – separate component from the set C_1;

C^{best} – search result for the most selective index;

C^{cat} – set of records found by inverted indexes; $C(I,Q)$ – the set of candidates obtained after applying the index I;

C_j – domain of the categorical attribute A_j^{cat};

c_{ki} – components from the C catalog;

C_1 – intermediate set of candidates after the most selective index;

C^{num} – set of records found using the R-tree;

count – number of components in the catalog for which the value of attribute A_i^{num} lies in the range $[\min_i, \max_i]$;

$corr(A_q, A_k)$ – correlation function;

D_i – domain of numeric attribute $A_i^{num} \subset \mathbb{R}$;

$f_{range}(A_i^{num})$ – frequency of range requests;

H_i – histogram for attribute A_i^{num};

I – set of available indices;

i – numeric attribute index;

I^{cat} – set of categorical indices participating in the query;

I^{num} – set of numerical indices participating in a search query;

j – categorical attribute index;

l_i – lower bound of the range for the *i*th numeric attribute in the search query;

$\max_i$ – maximum value of the *i*th attribute among all components in the catalog;

$M_{\max}$ – resource limit: maximum available memory for the index;

$MBR(N)$ – minimum bounding rectangle for node N in an R-tree;

$M_{mem}(I)$ – memory occupied by the index structure;

$\min_i$ – minimum value of the ith attribute among all components in the catalog;

$P(C)$ – set of all subsets of the directory C (posting lists);

N – total number of entries in the directory;

P – probability;

$PF(I,Q)$ – pruning factor;

p_{qk} – correlation coefficient;

Q – search query;

Q^{cat} – part of the query with conditions for categorical attributes;

Q^{num} – part of the query containing numeric range conditions;

$\mathbb{R}^m$ – m-dimensional Euclidean space;

R_{cand} – intermediate set of candidates after intersection of index results (before exact verification);

$\mathrm{Re}\,s(Q)$ – final result of the query Q after exact verification of all conditions;

$|\mathrm{Re}\,s(Q)|$ – expected size of the result;

$s(Q)$ – overall query selectivity;

Sel% – maximum possible selectivity that a query can have after passing the strongest filter;

$s_i(Q)$ – selectivity for numeric attribute A_i^{num};

$s_j(Q)$ – selectivity for categorical attribute A_j^{cat};

s_j^{cat} – selectivity of the *j*-th categorical attribute;

s_i^{num} – selectivity of the *i*-th numerical attribute;

S_{total} – total selectivity;

$T(Q)$ – request execution time;

u_i – upper limit of the range for the *i*-th numeric attribute in the search query;

U_i – uniqueness coefficient;

$U(I,Q)$ – index utility coefficient;

V_i – set of actual unique values of attribute A_i^{num} in the catalog A_i^{num};

V_j – set of allowed values of categorical attribute A_j^{cat};

X – set of catalog components (data elements);

x_i – vector of numerical characteristics of the *i*-th component;

Γ – correlation matrix;

χ_j – inverted index;

κ – threshold parameter;

υ – attribute value for which the index stores a list of corresponding components;

π – order of its application index structure;

θ – threshold parameter;

θ_1 – lower selectivity threshold for strategy B;

θ_2 – upper selectivity threshold for strategy B;

σ – permutation function, which determines the order in which indices are applied;

σ^{thr} – threshold size (typically 64 bytes);

Ψ – set of all possible index structures.

INTRODUCTION

The digital transformation of industrial production, telecommunications, and engineering design is

DOI 10.15588/1607-3274-2026-2-18

accompanied by an exponential growth in the volume of technical information about radio-electronic components. Modern manufacturers of electronic products operate with catalogs containing from tens to hundreds of thousands of names of microcircuits, sensors, communication modules, and other components. Each position is characterized by numerous technical parameters that form a multidimensional information space of high complexity. Efficient and fast search in such spaces is a critically important task that is actively researched in modern information technologies [1–4].

Effective access to such information is a critical factor in the competitiveness of enterprises in high-tech sectors.

The speed of component selection directly affects the duration of the new product development cycle, the quality of technical solutions, and the overall productivity of engineering teams. CAD, PLM platforms, and intelligent logistics systems of the element base require high-performance search engines capable of processing complex multi-parameter queries in real time.

At the same time, traditional approaches to organizing searches in structured catalogs, developed mainly for general-purpose transactional systems, demonstrate limitations when working with the specifics of technical data. The nature of engineering queries, combining numerical ranges of parameters with categorical filters, requires specialized solutions that take into account the specifics of the subject area.

Scientific tasks include:

– analysis of modern indexing methods for multi-attribute search in technical component catalogs and determination of their limitations for processing mixed (numeric and categorical) queries;

– development of a formal mathematical model of multi-attribute search, including a description of the data structure, query model, selectivity criteria, and optimization formulation;

– development of a hybrid index architecture that combines an R-tree for multidimensional filtering of numerical parameters and inverted indexes for categorical features;

– formalization of an algorithm for automatic distribution of attributes between index types based on analysis of data characteristics and query patterns;

– development of a mechanism for dynamic selection of a query execution strategy that adaptively chooses between access strategies based on an assessment of the selectivity of conditions;

– experimental verification of the effectiveness of the proposed method on realistic datasets of radioelectronic component catalogs;

– comparative analysis of query execution time of the developed hybrid system relative to traditional indexing approaches.

The initial data for the study are the technical specifications of radio electronic components, the architecture of modern DBMSs, as well as the accepted requirements for the performance of engineering information systems.

The object of research is the process of multi-attribute search in structured catalogs of radio-electronic components.

The subject of research is algorithms and data structures for combined indexing aimed at optimizing this process.

The purpose of the work is to is the development of a combined indexing method for effective execution of complex search queries in multi-dimensional catalogs of technical components.

1 PROBLEM STATEMENT

In modern catalogs of technical components, numerical parameters and categorical characteristics (case type, interface, manufacturer, purpose, etc.) form a multidimensional data space. An increase in the number of attributes leads to a significant decrease in the efficiency of traditional indexing mechanisms of relational DBMSs, especially for combined queries that include both numerical range conditions and categorical filters.

In such conditions, classic B-trees or composite indexes provide an unacceptably large number of intermediate samples, which significantly affects the query processing time. High dimensionality of data also leads to the effect of the “curse of dimensionality”, in which spatial structures (R-tree, KD-tree) lose selectivity as the number of dimensions increases.

Thus, an unsolved problem is the construction of an index structure that simultaneously works with numerical and categorical attributes and provides consistent performance in high-dimensional parameter spaces.

Let us now describe the catalog data structure with which the index system will work. Let the set of catalog components be $X = \{x_1, ..., x_N\}$.

Then each element is described by a set of attributes of two types: $A^{num} = \{A_1^{num}, A_2^{num}, ..., A_m^{num}\}$, $i \in \{1, \ldots, m\}$, $A_i^{num} \in D_i \subset \mathbb{R}$; numerical attributes and $A^{cat} = \{A_1^{cat}, A_2^{cat}, ..., A_k^{cat}\}$ $j \in \{1, \ldots, k\}$, $A_j^{cat} \in C_j$ categorical attributes. Therefore, $x = (A_m^{num}, ..., A_k^{cat}) \in D_1 \times ... \times D_m \times C_1 \times ... \times C_k$.

Let’s move on to describing the query model. Then, we define the search query as a conjunction of conditions over the attributes: $Q = Q^{num} \wedge Q^{cat}$; numeric conditions $Q^{num}(x) = \wedge(l_i \le A_i^{num} \le u_i)$ at $i \in \{1, \ldots, m\}$ and $i \in I^{num}$; categorical conditions $Q^{cat}(x) = \wedge(A_j^{cat} \in V_j)$, $V_j \subseteq C_j$, at $j \in \{1, \ldots, k\}$ and $j \in I^{cat}$. Thus, the result of the query is $\mathrm{Re}\, s(Q) = \{x \in X \mid Q(x) = true\}$.

The evaluation of the efficiency of the index structure is inextricably linked to the concept of query selectivity, which determines the expected proportion of objects satisfying the given criteria. Let’s start with numerical attributes $s_i(Q) = P(l_i \le A_i^{num} \le u_i)$; for categorical

DOI 10.15588/1607-3274-2026-2-18

attribute $s_j(Q) = P(A_j^{cat} \in V_j)$; with independence $s(Q) = \prod_{i \in I^{num}} s_i(Q) \cdot \prod_{j \in I^{cat}} s_j(Q)$.

Then, the expected number of elements in the result is $\mathrm{Re}\, s(Q) \approx N \cdot s(Q)$.

Criteria for evaluating the selectivity of index structures. Let the efficiency of an index structure I for a given query Q be evaluated by the following metrics: the pruning factor shows what proportion of records the index allows to be screened out without checking: $PF(I,Q) = \frac{1 - |C(I,Q)|}{N}$; the index utility coefficient characterizes the accuracy of the cutoff: $U(I,Q) = |\mathrm{Re}\, s(Q)| / |C(I,Q)|$. Therefore, an ideal value $U(I,Q) = 1$ means the absence of false positives – the index returns only relevant records.

Based on the analysis of selectivity and the expected structure of queries, it is possible to formulate a universal optimization statement that will combine the problem of building an index and the problem of minimizing query execution time. Optimization statement. Let the query $T(Q, I, \pi)$ execution time depend on the index structure I and the order of its application π. It is necessary to find such an index structure $I \in \Psi$ and the order of its application π when executing the query Q, such that $T(Q, I, \pi) \to \min$ subject to resource constraints: $M_{mem}(I) \le M_{\max}$.

As a result, a formal mathematical model of multi-attribute search was obtained, which includes: a description of the catalog data structure, a search query model, criteria for evaluating the selectivity of index structures, and a universal optimization formulation that combines the task of constructing an index structure and the task of minimizing its execution time for specific queries. Based on this formulation, a combined indexing method with a hybrid architecture was developed that combines an R-tree for numerical attributes and inverted indexes for categorical features, ensuring efficient execution of complex queries in high-dimensional catalogs of technical components through adaptive strategy selection based on condition selectivity analysis.

2 REVIEW OF THE LITERATURE

The problem of efficient search in multi-attribute catalogs is actively studied in the context of optimizing access to structured data. Classical approaches are based on B-trees [5, 6], which provide efficient one-dimensional search and range queries, but show limitations when processing multidimensional queries that combine conditions on different attributes. Composite indexes based on B-trees can improve the situation for specific combinations of attributes, but their efficiency drops sharply with increasing dimensionality and for queries that do not correspond to the given order of attributes in the index [5, 6].

To overcome the limitations of one-dimensional structures, spatial indexes such as the R-tree and its variants (R*-tree, R+-tree) have been developed. These structures organize multidimensional data into a hierarchy of bounding boxes, which allows for efficient filtering of regions of space that do not meet the range conditions of the query. However, for high-dimensional data, they suffer from the “curse of dimensionality” – a decrease in node selectivity and an increase in the intersections of bounding boxes, which leads to poor performance [7, 8]. In parallel, inverted indexes and bitmap indexes are widely used for indexing categorical attributes. Inverted indexes, borrowed from the field of information retrieval, provide fast access to records by exact attribute values and efficient execution of Boolean operations (AND, OR) over large sets of identifiers [9]. Bitmap indexes are especially effective for attributes with low cardinality, allowing filtering using bitwise operations [10].

A separate direction is hybrid approaches that try to combine the advantages of different data structures. The integration of bitmap indexes with B-trees to accelerate range queries is being investigated [11], as well as the combination of spatial indexes with inverted indexes for processing mixed data types [12]. Adaptive methods for index selection based on query statistics and data characteristics are being developed [13].

Existing hybrid approaches, for example, the combination of R*-tree and k-d trees [14], effectively solve the search problems in distributed graph data (e.g., Linked Open Data); they are not specialized for processing complex multi-attribute queries in structured technical catalogs. In contrast, the proposed combined indexing method is specially designed for processing complex multi-attribute queries in structured technical catalogs, where the key innovation is the adaptive mechanism of dynamic query planning.

As for other areas of hybrid indexing, such as the previously discussed methods for graph or semantic data, modern research in the field of spatial indexing also actively uses hybrid approaches to overcome the limitations of traditional structures, in particular for working with unevenly distributed data. A striking example is the R-MLGTI method [15]. This approach is interesting in that, like the presented development, it uses a combination of structures (multi-level grid and R-tree) to adapt to the characteristics of the data. However, their fundamental goals and subject areas differ significantly. Unlike R-MLGTI, which is optimized for accelerating spatial range queries in a two-dimensional geographic coordinate system and struggles with the problem of uneven density of objects in physical space, the proposed method is focused on a logical multidimensional space of technical parameters. Thus, while R-MLGTI effectively determines “where the object is located”, the proposed system solves the search problem according to the criteria “what properties it possesses”, which requires a fundamentally different architecture and optimization mechanisms, in particular, the introduction of inverted indexes and an adaptive query planner.

DOI 10.15588/1607-3274-2026-2-18

A somewhat different but related approach to hybrid spatial indexing is presented in [16], where a combination of a grid and an R-tree is used to speed up queries on geographic data on car sales, demonstrating an advantage over a B-tree. Although this and the previously considered methods (R-MLGTI, R-tree + k-d trees) are effective in their subject areas – physical spatial indexing and graph search – none of them is designed to optimize complex multi-attribute queries in structured technical catalogs. This is the niche that the presented work fills.

Therefore, this work focuses on the development of a combined indexing method and its comparative analysis with traditional approaches. The experimental modeling performed allowed for a comprehensive assessment of the key performance characteristics of the proposed solution, including its performance in processing different types of search queries, the efficiency of cutting off unnecessary data, and the overall impact on the response time of the information system.

3 MATERIALS AND METHODS

To solve the formulated problem, a combined indexing method was developed, which is based on a hybrid architecture that synthesizes the advantages of spatial (R-tree) and inverted index structures.

The basis of the method is four interrelated components:

1. A hybrid index structure that integrates an R-tree for multidimensional filtering of numerical parameters and a system of inverted indexes for categorical features. This structure is managed through a single metadata layer that ensures the coordinated operation of both types of indexes.

2. An algorithm for automatic distribution of attributes between index types, which, based on the analysis of data characteristics (type, cardinality, selectivity) and historical query patterns, determines the optimal belonging of each attribute to a spatial or inverted index structure.

3. A procedure for building a combined index that implements the stages of creating and synchronizing the R-tree and inverted indexes according to the distribution obtained from the automatic distribution algorithm, ensuring the integrity of the hybrid structure.

4. A mechanism for dynamically selecting a query execution strategy that analyzes the selectivity of each specific query condition and adaptively chooses between the “index-first”, “parallel merge”, and “full scan” strategies to minimize response time.

Each component of the method is described in detail below.

The developed combined indexing method is based on a hybrid architecture that synthesizes the advantages of spatial and inverted index structures for efficient processing of mixed queries in multi-attribute catalogs of radio-electronic components. The main idea is to use specialized indexes for different data types in parallel with subsequent intelligent merging of intermediate results.

Let’s start with a description of the hybrid index architecture (structure).

The hybrid index system consists of two main components that function independently but in a coordinated manner:

1. Spatial indexing component (R-tree).

2. Categorical indexing component (inverted index).

The first component is the spatial indexing component (R-tree), which is needed for indexing numerical attributes: $A^{num} = \{A_1^{num}, A_2^{num}, ..., A_m^{num}\}$ a modified R-tree structure is used, which interprets each component c_i as a point or bounding rectangle in m-dimensional Euclidean space $\mathbb{R}^m$. Each numerical attribute corresponds to a separate coordinate axis of this space.

The vector of numerical characteristics of a component is represented as:

$$x_i = (x_{i1}, x_{i2}, ..., x_{im},) \in \mathbb{R}^m . \quad (1)$$

An R-tree organizes these vectors into a hierarchical structure, where each node contains the minimum bounding rectangles (MBRs) that encompass all of its children. For a node N, the MBR is defined as:

$$MBR(N) = \prod_{i\in 1}^{m} \left[\min_i \max_i \right]. \quad (2)$$

Categorical indexing component (inverted index). A system of inverted indexes is used for categorical attributes $A^{cat} = \{A_1^{cat}, A_2^{cat}, ..., A_k^{cat}\}$. For each categorical attribute A_j, a separate inverted index χ_j is constructed, which is a mapping of:

$$\chi_j : D_j \to \mathrm{P}(C) . \quad (3)$$

An inverted index is a data structure that allows you to quickly find all documents (or records) that contain a particular word or value.

For each value $\upsilon \in D_j$, an inverted index stores an ordered list of component identifiers:

$$\chi_j(\upsilon) = \{c_{k1}, c_{k2}, ..., c_{kp}\} . \quad (4)$$

Therefore, c_{ki} there are components for which the attribute A_j^{cat} takes the value υ.

In this study, there is a need for a metadata layer for optimization. The effectiveness of a hybrid index system critically depends on the ability to make informed decisions about the order in which indexes are applied. Since different queries have different selectivity for R-tree and inverted indexes, the system needs additional statistical information to predict the performance of each

DOI 10.15588/1607-3274-2026-2-18

approach. Without such information, the query optimizer would be forced to use heuristics or random selection, which would lead to suboptimal performance.

Thus, to ensure intelligent query execution planning, the system includes a metadata component that stores statistical information about the distribution of attribute values in the catalog:

– histograms of the distribution of numeric attributes – allow you to estimate how many components fall into a given range $[l_i, u_i]$ without actually scanning the R-tree. The histogram for an attribute A_i^{num} divides its range into b intervals (buckets), for each of which the number of records is stored:

$$H_i = \{([\min_1, \max_1], count_1), ..., ([\min_b, \max_b], count_b)\}; \quad (5)$$

– cardinality of categorical attributes – for each $A_j^{cat} \in A^{cat}$, $|D_j|$ (total number of unique values) and frequency distribution of individual values are stored. This allows to quickly assess the selectivity of the condition $A_j^{cat} \in V_j$:

$$S_j^{cat} = \frac{\sum_{\upsilon \in V_j} |\chi_j(\upsilon)|}{N}; \quad (6)$$

– correlation matrix between attributes – if there is a statistical dependence between attributes (for example, high frequency components usually have a smaller temperature range), the independent selectivity model $S_{total} = \prod S_j$ gives inaccurate estimates. The correlation matrix Γ stores the correlation coefficients p_{qk} between pairs of attributes:

$$\Gamma = [p_{qk}], \text{ at } p_{qk} = corr(A_q, A_k), \quad (7)$$

this information is used to correct selectivity estimates for combined queries.

The above metadata describes the properties of the data, but query optimization also depends on user behavior. In real systems, the distribution of queries is uneven, so the choice of the optimal index must take into account the most likely search scenarios.

In this regard, the metadata component additionally stores query frequency statistics.

Query frequency statistics – the system stores aggregated data about typical query patterns: which attributes are most often used together, which value ranges are popular, what is the average selectivity of different query types. This information allows you to adaptively adjust index parameters (for example, cache frequently requested posting lists or optimize the R-tree structure for typical ranges).

Next, we present a diagram of the interaction between the R-tree and the inverted index.

The interaction between the components of the hybrid system is implemented through a coordination module – the query planner (Query Planner), which analyzes the structure of the incoming query $Q = (Q^{num}, Q^{cat})$ and determines the optimal execution strategy.

Stage 1. Query analysis and selectivity assessment. When a query arrives, the planner decomposes it into a numerical and categorical part, and the numerical part Q^{num} defines a hyperrectangle in space $\mathbb{R}^m$:

$$Q^{num} = \prod_{i=1}^{m} [l_i, u_i]. \quad (8)$$

or equivalent

$$Q^{num} = \{x \in \mathbb{R}^{|m|} \mid l_i \le x_i \le u_i, \forall i \in m\}. \quad (9)$$

Categorical part Q^{cat} defines the set of allowed values for each categorical attribute:

$$Q^{cat} = \{V_j \subseteq C_j \mid j \in k\}. \quad (10)$$

or predicate

$$Q^{cat} = \{c \in C \mid \forall j \in k : \upsilon_j(c) \in V_j\}. \quad (11)$$

or in terms of outcome sets:

$$\text{Re}\, s(Q) = \{c \in C \mid c \in Q^{num} \ i \ c \in Q^{cat}\}. \quad (12)$$

For each index, selectivity is calculated – the expected proportion of records that satisfy the conditions:

– for R-tree:

$$s_i^{num} = \prod_{i=1}^{n} \frac{u_i - l_i}{\max_i - \min_i}; \quad (13)$$

– for categorical attributes:

$$s_j^{cat} = \frac{|V_j|}{|D_j|}; \quad (14)$$

– overall selectivity (attribute independence):

$$s(Q) = \prod_{i \in I^{num}} s_i^{num} \cdot \prod_{j \in I^{cat}} s_j^{cat}. \quad (15)$$

DOI 10.15588/1607-3274-2026-2-18

Stage 2. Selection of execution strategy. Based on the calculated selectivities, the planner chooses one of three strategies:

Strategy A (Index-First). If at least one of the selectivities is very high: $s^{num} < \theta$ or $s^{cat} < \theta$ where $\theta = 0.01,\ \theta = 0.01\ (1\%)$, then choose "Strategy A (Index-First)" and then search for the most selective index:

– if $s^{num} < s^{cat}$ → search the R-tree

– if $s^{cat} < s^{num}$ → search the inverted indices.

The result is a set of candidates C^{best}. Definition C^{best}:

– if $s^{num} < s^{cat}$ → the most selective numerical index → $C^{best} = C^{num}$;

– if $s^{cat} < s^{num}$ → the most selective categorical index → $C^{best} = C^{cat}$. Next, the candidates C^{best} are filtered. For each candidate $c \in C^{best}$, all the remaining conditions are checked:

– if $C^{best} = C^{num}$ → check categorical conditions Q^{cat};

– if $C^{best} = C^{cat}$ → check numerical conditions Q^{num}.

The final result is formed by:

$$Res(Q) = \{c \in C^{best} \mid c \text{ satisfies all conditions } Q\}.$$

If the condition that at least one of the selectivities is very high is not met, then the following condition is checked: "Do both indices have average selectivity?", that is, $\theta_1 \le s^{num} \le \theta_2$ and $\theta_1 \le s^{cat} \le \theta_2$ where $\theta_2 = 0.3,\ \theta_2 = 30\%$.

Strategy B (Parallel-Merge). It is used if both selectivities lie in the middle range $\theta_1 < s_i^{num} < \theta_2$ and $\theta_1 < s_j^{cat} < \theta_2$, for example, $0.01 < s_i^{num} < 0.3$ and $0.01 < s_j^{cat} < 0.3$:

– search in parallel by R-tree → C^{num};

– search in parallel by inverted indices → C^{cat};

– calculate the intersection $R_{cand} = C^{num} \cap C^{cat}$.

Strategy C (Full-Scan). If both selectivities are very low $s_i^{num} > \theta_2$ and $s_j^{cat} > \theta_2$, for example, $s_i^{num} > 0.3$ and $s_j^{cat} > 0.3$ or any index returns a significant part of the catalog, the planner chooses a full scan of the table:

– sequentially check all catalog components;

– apply all query conditions without indexes;

– return a set:

$$\text{Re}\, s(Q) = \{c \in C \mid \text{satisfies all conditions } Q\}.$$

Strategy C is important when: data distribution is uniform; indexes are degraded; query is "everything there is"; category or range is too broad.

Stage 3. Execution and merging of results. The selected strategy is executed by the corresponding components:

– R-tree processes the numerical part, returning a set of candidates;

– inverted indexes process categorical conditions through posting list intersection operations:

$$C^{cat}(Q) = \bigcap_{j=1}^{m} \bigcup_{\upsilon \in V_j} \chi_j(\upsilon). \quad (16)$$

Thus, the result is formed as:

$$\text{Re}\, s(Q) = C^{num}(Q) \cap C^{cat}(Q) \cap \{\text{accurate check}\}, \quad (17)$$

where the exact verification stage eliminates false positives that could arise due to the approximate nature of the MBR in the R-tree.

Let's move on to the principles of attribute distribution between index types.

The effectiveness of a hybrid system critically depends on the correct distribution of attributes between the R-tree and inverted indexes. A system of rules has been developed that automates this process.

Rule 1. Data type. The basic criterion is the natural type of the attribute:

$$A_i^{num} \in A^{num} \Leftrightarrow domain(A_i^{num}) \subseteq \mathbb{R}, \quad (18)$$

$$A_j^{cat} \in A^{cat} \Leftrightarrow domain(A_j^{cat}) = \text{discrete set}. \quad (19)$$

Rule 2. Cardinality. For attributes with low cardinality, even numeric values should be indexed with an inverted index:

$$\text{if } |D_i| < \kappa \text{ and } A_i^{num} \in A^{num} \Rightarrow A_j^{cat} \to A^{cat}.$$

let the threshold parameter κ be typically equal to 20 for catalogs (e.g. electronic components).

Example: the attribute "number of housing pins" can take the value $\{4, 8, 14, 16, 20, ...\}$. In this case $|D_i| < 20$, it is more efficient to index it as categorical.

Rule 3. Query type. Query history analysis determines which operations are performed more often:

– if the attribute A_i^{num} is mainly used in range queries $(l_i \le A_i^{num} \le u_i) \to$ R-tree;

– if the attribute (numeric or categorical) is mainly used in exact matches $(A = \upsilon)$ or multiple selection $(A \in V)$ → inverted index.

For numeric attributes, the frequency of range queries is calculated:

$$f_{range}(A_i^{num}) = \frac{\text{number of range queries per } A_i^{num}}{\text{total number of requests } A_i^{num}}. \quad (20)$$

DOI 10.15588/1607-3274-2026-2-18

Criterion $f_{range}(A_i^{num}) > 0.3 \Rightarrow$ R-tree.

Rule 4. Selectivity (uniqueness coefficient). The rule applies to numeric attributes, since it is for them that the efficiency of indexing in the R-tree critically depends on the ability to divide the space. Therefore, the uniqueness coefficient:

$$U_i = \frac{|V_i|}{N}. \quad (21)$$

If $U_i > 0{,}5$, the attribute has high discriminative ability → R-tree.

For categorical attributes A_j^{cat} where $j \in \{1, \dots, k\}$, high uniqueness often indicates proximity to an identifier (e.g., a numeric attribute, a serial number).

Thus, if a numeric attribute has a high uniqueness coefficient, it is advisable to index it using an R-tree. In this case, it is more efficient to use an inverted index (according to Rule 5 if the size of the values is large, or Rule 2 if the cardinality is low).

Rule 5. Data size. For attributes that take up a lot of memory (text fields, long strings).

If $size(A_q) > \sigma^{thr} \Rightarrow$ inverted index with compression.

Rule 5 is especially important for categorical attributes with long text values, as well as for numeric attributes with high precision (for example, serial numbers).

Thus, the above rules allow you to classify attributes and determine the preferred type of index for them. However, in practice, there are situations when the characteristics of an attribute satisfy several criteria at once or change over time (for example, due to catalog updates or changes in the nature of queries).

In such cases, it is advisable to use combined indexing of the same attribute.

Hybrid indexing of individual attributes. In some cases, one attribute can be indexed in both ways:

– numeric attribute with frequent exact queries → additionally an inverted index is created for popular values;

– categorical attribute with natural ordering → can be included in an R-tree with categories mapped to a numeric axis.

The architecture of a hybrid index system is proposed.

The developed method is based on a hybrid architecture that combines three key components: a spatial indexing component based on an R-tree for efficient processing of numerical attributes, a categorical indexing component using inverted indexes for fast search by discrete features, and a metadata layer for intelligent optimization of query execution. Each component specializes in processing a specific type of data, which allows achieving maximum performance when performing complex multi-attribute queries in radio electronic component catalogs.

Spatial indexing component (R-tree). The spatial indexing component is designed to efficiently process numerical attributes of technical components and perform range queries. It is based on a modified R-tree structure, which interprets each component as a point in m-dimensional Euclidean space, where each numerical attribute corresponds to a separate coordinate axis (detailed mathematical definitions are given in formulas 1–2).

The R-tree organizes these vectors into a hierarchical structure, where each node contains minimum bounding boxes (MBRs) that encompass all of its children. This organization allows for efficient pruning of large regions of the parameter space early in the search, minimizing the number of component checks. This structure provides logarithmic search complexity for range queries of the type “find all components with supply voltages from 3 to 5,5 V and operating temperatures from –40 to +85 °C,” which are typical of engineering component selection problems. Spatial indexing is particularly effective for attributes with high cardinality and uniform distribution of values, where traditional B-trees exhibit suboptimal performance due to the need to check multiple indices sequentially.

Categorical indexing component (inverted indexes).

The categorical indexing component implements fast, accurate search by discrete attributes of technical components, such as case type, communication interface, manufacturer, or purpose. For each categorical attribute, a separate inverted index is built (formulas 3–4).

An inverted index is a classic data structure from information retrieval theory that allows you to quickly find all records containing a certain attribute value. Unlike a direct index, which stores a list of its attributes for each component, an inverted index inverts this dependency. An inverted index stores an ordered list of component identifiers (the so-called posting list).

The advantages of inverted indexes are the constant complexity of accessing the posting list by attribute value (O(1) when using hash tables) and the ability to efficiently implement Boolean operations AND, OR, NOT through operations on ordered sets. This makes them ideal for processing queries of the type “find all microcontrollers manufactured by STMicroelectronics with SPI interface and QFN package”, where it is necessary to quickly calculate the intersection of several posting lists.

Metadata layer for optimization. The effectiveness of a hybrid index system critically depends on the ability to make informed decisions about the order in which indexes to use and the optimal query execution strategy. Since different queries have different selectivity for R-trees and inverted indexes, the system needs additional statistical information to predict the performance of each approach. Without such information, the query optimizer would be forced to use heuristics or random selection, which would lead to suboptimal performance, especially for complex multi-parameter queries.

Thus, to ensure intelligent query execution planning, the system includes a metadata component that stores and

DOI 10.15588/1607-3274-2026-2-18

constantly updates statistical information about the distribution of attribute values in the catalog and the characteristics of typical queries. The metadata layer includes: histograms of the distribution of numeric attributes (formula 5), cardinality of categorical attributes (formula 6), correlation matrix between attributes (formula 7), and query frequency statistics.

The hybrid index system is presented in Fig. 1.

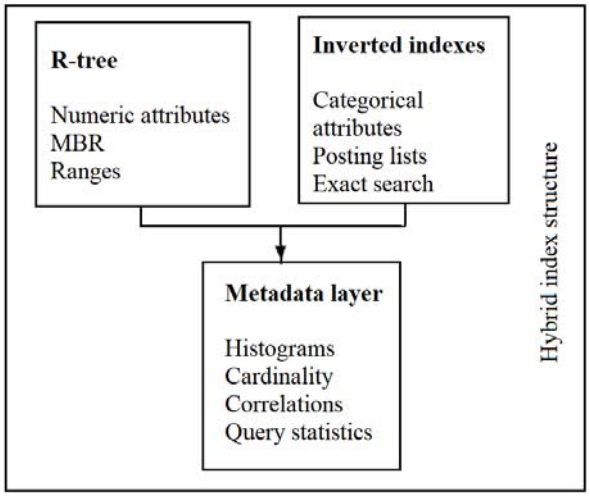

Figure 1 – Hybrid index system

The presented architecture demonstrates the structural organization of the system, but the key aspect of efficiency is the mechanism of coordinated interaction between components. However, for this interaction to be as productive as possible, it is necessary to determine in advance which of the components – R-tree or inverted index – is best suited for indexing each specific attribute. This fundamental task is solved by the algorithm for automatic distribution of attributes between index types. This algorithm, analyzing both the inherent characteristics of the data (type, cardinality, selectivity) and the patterns of their practical use from the history of queries, accurately determines the optimal belonging of each attribute to the spatial or inverted index structure, laying the foundation for further effective interaction between them.

An algorithm for automatic distribution of attributes between index types is proposed, which consistently applies the formulated rules 1–5 (Fig. 2).

Stage 1. Initialization. At this stage, data structures are created that will be used during the algorithm:

– A_num (numeric indices, A^{num}) – initially an empty set of attributes. Attributes will be added here for which the algorithm will determine that it is most efficient to use indices optimized for numerical ranges. This usually applies to data on which comparison operations (>, <, BETWEEN), sorting or range search are performed. As an example, in this study, these are supply voltage, input/output signal level, exact measurement timestamp, etc.;

– A_cat (categorical indexes, A^{cat}) – also an initially empty set. Here will be attributes for which indexes optimized for working with categorical or discrete values are better suited. As an example, in this study – this is the type of component, synchronization status, etc.;

– conflicts – a set for resolving conflicts. During the analysis, some attributes may demonstrate features suitable for both types of indexes. Instead of being immediately selected, they are placed in this temporary set for further, deeper analysis based on additional criteria (for example, analysis of HisQ query history.

Stage 2. Analyze each individual attribute for a primary, formal classification based on the mathematical nature or declared data type of the attribute.

Stage 2.1. Define the base data type. The domain of an attribute (domain(A_i) or $domain(A_i^{num})$) is the set of all valid values that this attribute can take. In practice, the algorithm determines this not abstractly, but by analyzing the metadata (DDL) of the table (column type: INT, FLOAT, VARCHAR, etc.) and/or directly by inspecting the DS data sample.

In practice, access to the theoretical domain of the attribute (domain(A_i)) is not possible. Therefore, in stage 2.1, the algorithm uses the data sample DS as an operational approximation of the domain. It inspects the values of the attribute A_i in the sample DS and its metadata to make an initial inference about the underlying type. This inference is a first approximation and can be refined later.

Condition check. If domain(A_i) is numbers/digits etc., then base type (base_type) is numeric (NUMERIC), otherwise, attribute is categorical type (CATEGORICAL). Proceed to next stage.

After formal classification (base_type), it examines the real statistics of the attribute in the data sample DS.

Stage 2.2. Calculation of the attribute characteristics on the DS sample. The obtained metrics will become objective arguments for clarifying or correcting the initial assumption.

The algorithm scans all rows of the DS sample for the attribute A_i and forms a set of its unique values D_i or D_i. The result is an unordered collection without repetitions.

Based on the set of unique values D_i, a key metric is calculated – the cardinality of the attribute (|D_i| or $|D_i|$). It is an absolute number that shows how many different variants of the values of the attribute A_i actually exist in the analyzed data sample DS. Calculation – determining the size (number of elements) of the set D_i.

Low cardinality (e.g., |D_i| < 20) is a strong indicator of categorical nature, even for numeric attributes. High cardinality (close to N_ds) is a sign of a unique identifier or numeric range.

Next, the algorithm determines the amount of data available for analysis.

DOI 10.15588/1607-3274-2026-2-18

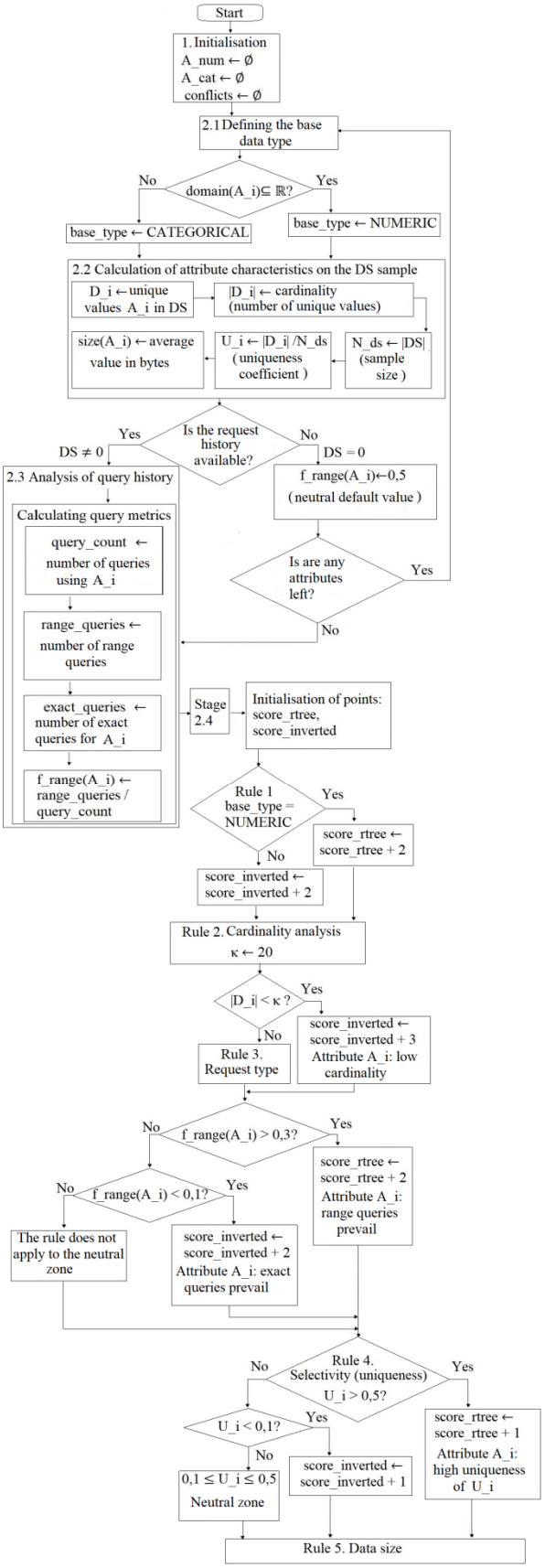

Figure 2 – Algorithm for automatic distribution of attributes between index types (Part 1)

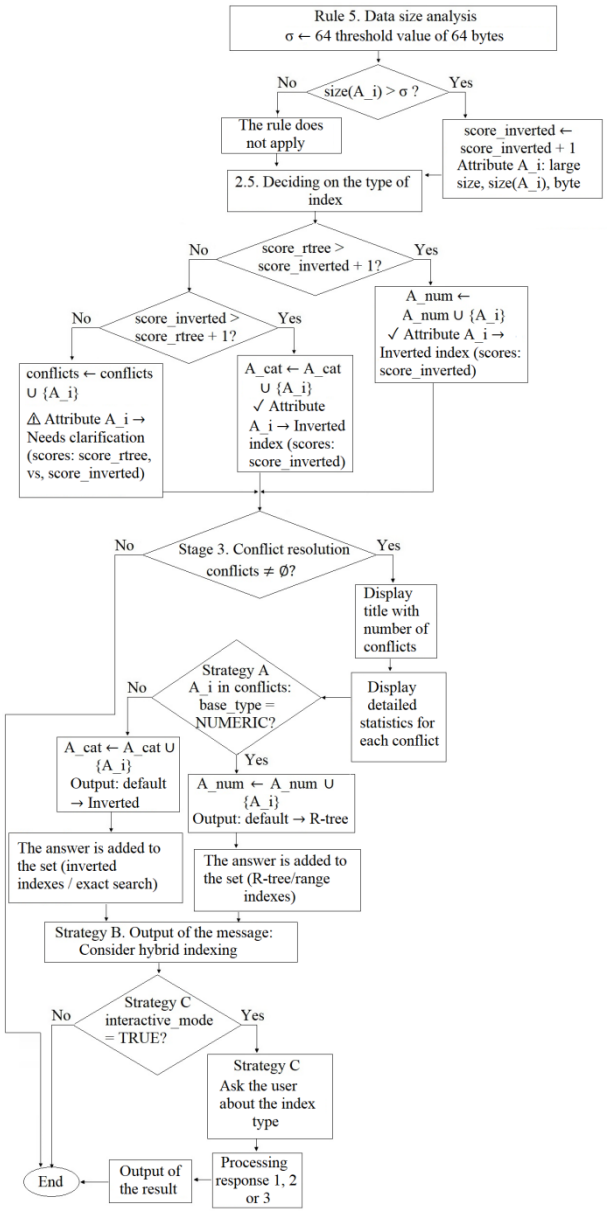

Figure 2 – Algorithm for automatic distribution of attributes between index types (Part 2)

The variable N_ds receives the value of the number of rows in the DS sample. This value is critically important for the subsequent calculation of relative characteristics (for example, the uniqueness coefficient), since it allows you to evaluate statistical indicators not in absolute numbers, but in the context of the total sample volume. The larger and more representative the sample (the larger N_ds), the more reliable the metrics calculated at this stage will be. That is, the variable N_ds records the size (volume) of the DS data sample.

|DS| to denote the cardinality of the set, i.e. the number of elements in it. Since DS is a sample of data (a set of records, rows of a table), |DS| denotes the total number of records (rows, objects) in this sample.

Next, the uniqueness coefficient (U_i or U_i) is determined – a measure of the relative diversity of

DOI 10.15588/1607-3274-2026-2-18

attribute values. It shows what proportion of the total number of records are unique values. A coefficient close to 1 indicates an almost unique attribute (identifier), while a value close to 0 indicates high repeatability (a typical categorical variable).

Definition of the uniqueness coefficient: U_i = number of unique values of the attribute A_i / total number of rows in the DS sample.

The uniqueness coefficient is a key heuristic indicator that helps to detect a discrepancy between the formal data type (base_type) and its actual use.

Then the average size of the attribute value is calculated. At this stage, the metric size(A_i) is calculated for the attribute A_i – the average size of an individual value of this attribute in bytes. This characteristic helps to estimate the "weight" of the data to be indexed.

Although the average size does not directly determine the choice between numeric (A_num) and categorical (A_cat) index types, it is a critically important additional optimization criterion.

After calculating the statistical characteristics, the algorithm obtains an objective picture of the data structure of the attribute A_i:

– formal type (base_type);

– nature of value distribution (|D_i|, U_i);

– technical parameters (size(A_i)).

However, to make an index decision, it is not enough to know just what data is stored. It is critical to understand how that data is used in real-world scenarios. Therefore, the next logical step is to analyze query history.

So, the check block "Is the query history available?" comes. If available, then go to stage 2.3, otherwise if the history is absent or empty DS = 0 the algorithm has to act under conditions of uncertainty regarding the load. In such a situation, it cannot assume which type of search (range or exact) will prevail. To avoid biased selection, the algorithm sets f_range(A_i) to a neutral default value of 0,5. The next stage is to check: "Is are any attributes left?", if "Yes", then go to stage 2.1 "Definition of the base data type", otherwise go to stage 2.3.

Stage 2.3. Query history analysis. If the query history DS ≠ 0 is available and not empty. This stage is the heart of pragmatic optimization. The algorithm stops looking at the data as a static object and starts analyzing the dynamics of its use. The goal is to measure the real operational need for different types of searches for the attribute A_i.

The algorithm filters the entire DS history, selecting only those queries in which the attribute A_i appears: *query_count* ← *number of queries using A_i*.

The result is an integer *query_count* – the total number of relevant queries.

query_count is a metric that shows the operational importance of an attribute.

A high *query_count* means that the attribute is actively used, and choosing the optimal index for it will have a significant impact on the performance of the entire system.

Next, among the selected relevant queries, the algorithm counts those that contain range operations with the A_i attribute. range_queries ← *number of range queries* for A_i. *range_queries* ← *number of range queries for A_i*.

Range queries are considered conditions with comparison operators (>, <, >=, <=, BETWEEN), as well as ORDER BY A_i and LIMIT ... OFFSET ... constructs, which implicitly require ordering by attribute values.

The result will be an integer number of *range_queries*.

range_queries directly measures the need for range access, which is the main scenario for using numeric indexes (A_num). Similarly, the algorithm counts queries with exact match conditions for A_i. *exact_queries* ← *number of exact queries for A_i*

Exact queries are considered conditions with equality operators (=), as well as IN (list_of_values) and IS NULL.

The result will be an integer number of *exact_queries*.

exact_queries measures the need for fast exact searches, which is a universal scenario but is particularly well implemented by categorical indexes (A_cat) such as hash indexes. The final stage is to calculate the proportion of range queries. *f_range(A_i)* ← *range_queries / query_count* – this is the ratio of the number of range queries to the total number of relevant queries.

Range and interpretation:

– f_range ≈ 1 – almost all queries use range operations. This is a direct and strong signal in favor of assigning the attribute to A_num (numeric indices).

– f_range ≈ 0 – almost all queries are exact matches. This is a direct and strong signal in favor of A_cat (categorical indices).

– 0,2 < f_range < 0,8 – mixed picture. This value indicates a potential conflict between the data structure and usage patterns. The attribute will become a prime candidate for detailed consideration and conflict resolution in the next stages of the algorithm.

As a result of analyzing the DS history for the attribute A_i, a key metric f_range(A_i) is formed, which quantifies the operational requirement for the indexing type. This metric will become one of the main arguments in the final decision rule, allowing the algorithm to recommend the index type that best matches the real workload.

The transition to stage 2.4 is the transition from the fact-gathering and evaluation phase to the rule-based decision-making phase.

Stage 2.4. Applying index type selection rules. The goal of this stage is not simply to choose a "numeric" or "categorical" type, but to assess the suitability of the attribute for specific indexing technologies that best implement these types. In this example, the algorithm considers two competing types of indexes:

– R-tree (or B-tree/range index) – optimal for range search and ordering, corresponding to the abstract category A_num;

– inverted index – optimal for fast precise search and work with texts corresponding to the A_cat category.

DOI 10.15588/1607-3274-2026-2-18

To make an objective choice, a scoring system (score_rtree, score_inverted) has been introduced. Each rule (criterion) adds a certain number of points to a particular type of index, reflecting the strength of the corresponding argument. The starting hypothesis base_type = NUMERIC is a "zero approximation". That is, the start is made with the most obvious assumption-test: for numerical data, range indexes (like R-tree/B-tree) are more convenient, and for categorical ones – indexes for precise search (inverted, hash indexes).

"Rule 1 (Natural data type)", described in stage 2.4, is the simplest and most basic. It formally awards a point contribution to the type of index that logically corresponds to the formal nature of the data:

– if the numeric type (NUMERIC) receives +2 points for the R-tree (*score_rtree* ← *score_rtree* + 2);

– if the categorical type (CATEGORICAL) receives +2 points for the inverted index (*score_inverted* ← *score_inverted* + *2*).

However, this is just the beginning. This rule will be adjusted or strengthened by the following, more subtle rules that will take into account:

After applying "Rule 1", which is based on the formal data type, the algorithm moves on to a deeper analysis – studying the real data structure revealed in the DS sample.

"Rule 1" provides an initial hypothesis. However, this hypothesis can be refuted or strengthened by the values that the attribute takes in practice. A numeric attribute (for example, status_code INT) can have only 3–4 different values, which makes it inherently categorical. Conversely, a text field can contain almost unique values (for example, the email of the manufacturer of technical component elements), which brings it closer to the properties of a numeric identifier. That is why the logical next stage is "Rule 2" – cardinality analysis, which allows you to detect such a semantic contradiction between the data type and its real distribution.

Threshold is set (κ ← 20) – the operation of setting the threshold value for low cardinality.

Checking the key condition: is the number of unique attribute values less than the threshold?

If the condition is met, then (*score_inverted* ← *score_inverted* + 3), that is, "attribute A_i: low cardinality". *score_inverted* ← *score_inverted* + 3 – the main action of the rule: a strong increase in the weight of the inverted index.

If the condition is not met, then go to "Rule 3". The purpose of "Rule 3" is to use information from the query history (DS) to quantify the dominant access patterns to the attribute A_i and to prefer the index type that best serves them. "Rule 3" is based on the value of f_range(A_i) – the fraction of range queries, calculated in stage 2.3. It has two active branches and one implicit (neutral) branch.

The condition is checked: f_range(A_i) > 0.3?

If the condition is met, then output "Attribute A_i: range queries prevail". Then go to "Rule 4".

If the condition is not met, then check the following condition: f_range(A_i) < 0.1?

If the condition "f_range(A_i) < 0.1?" is met, then (*score_inverted* ← *score_inverted* + *2*) output "Attribute A_i: exact queries prevail". Then go to "Rule 4".

If the condition "f_range(A_i) < 0.1?" is not met, then "Rule 3" does not apply because it is a neutral zone. Then go to "Rule 4".

"Rule 4" – Selectivity (uniqueness). The purpose of this rule is to assess how "selective" an attribute is, that is, what proportion of table rows it is able to filter on average.

The peculiarity is that while "Rule 2" also analyzed the number of unique values (|D_i|), "Rule 4" works with a relative indicator U_i = |D_i| / N_ds – the uniqueness coefficient. This indicator better characterizes the "rarity" of values in the context of the entire data set.

The condition is checked: U_i > 0.5?: "Is the attribute highly unique?".

If the condition "U_i > 0.5?" is met, then (*score_rtree* ← *score_rtree* + *1*) calculate +1 for the R-tree, output the log and output "Attribute A_i: high uniqueness of U_i". Checking U_i > 0.5? is a check whether the attribute is highly unique? Then go to "Rule 5".

If the condition "U_i > 0,5?" is not met, then the next condition "U_i < 0,1?" is checked. Checking U_i < 0.1? is a check whether the attribute is low-unique? If this condition is met, then (*score_inverted* ← *score_inverted* + *1*) calculate +1 for the inverted index, go to "Rule 5".

If the condition "U_i < 0,1?" is not met, then this is the neutral zone "0.1 ≤ U_i ≤ 0.5". No actions are performed in the neutral zone. Then go to "Rule 5".

After analyzing all semantic and operational characteristics, we move on to the last technical criterion – the physical size of attribute values. This rule makes adjustments, taking into account the limitations and features of implementing different types of indexes in database management systems.

First σ ← 64 – sets the size threshold to 64 bytes. This is a typical technical threshold related to the size of memory pages in the DBMS, buffer sizes, and comparison efficiency. The value can be customized for a specific DBMS.

First, check the condition: size(A_i) > σ ?

If the average size of an attribute value (in bytes) exceeds the technical threshold (64 bytes), the attribute is classified as "large". Typical examples are long text descriptions, etc.

If the condition is met, then (*score_inverted* ← *score_inverted* + 1) print "Attribute A_i: large size, size(A_i), byte". Then go to stage 2.5.

If the condition is not met, the rule does not apply. Proceed to stage 2.5 – making a decision based on all points.

"Rule 5" completes the analysis by ensuring that the recommendation is not only semantically correct but also technically feasible. It prevents the creation of inefficient indexes for big data and guides the choice towards specialized structures optimized for working with large values.

Stage 2.5. Deciding on the type of index.

DOI 10.15588/1607-3274-2026-2-18

After completing all five evaluation rules, the algorithm receives two final numerical indicators for the "attribute A_i: score_rtree and score_inverted". These scores are the sum of all arguments "for" the R-tree (range index) and "for" the inverted index (exact search), respectively. Now it is time to compare these results and make a final decision. The goal of the stage is to determine, based on the accumulated scores, which category (A_num or A_cat) the attribute should be assigned to, or to recognize it as an ambiguous case that requires additional analysis.

Checking the condition: *score_rtree* > *score_inverted* + 1*?*

If this condition is met, then (A_num ← A_num ∪ {A_i}) the attribute is officially added to the set for numeric/range indices. Output: "✓Attribute A_i → R-tree (scores: score_rtree)". A clear advantage of the R-tree (range index). So, for the R-tree to win, it is not just more points that are needed, but an advantage of more than 1 point. This "margin" (+1) is a confidence threshold. It is needed to avoid random selection at a minimal difference that may arise due to the specificity of the rule weights.

A clear, positive message is generated for the log or user interface. A check mark (✓) symbolizes a successful selection, and displaying the number of points makes the process transparent.

If this condition is not met, then the transition to a new condition: score_inverted > score_rtree + 1? This is a symmetric test for the inverted index. The inverted index must also have an advantage of more than 1 point for a clear win.

Satisfying this condition will lead to A_cat ← A_cat ∪ {A_i}, i.e. adding the attribute to A_cat. Output a positive message. Output "✓ Attribute A_i → Inverted index (scores: score_inverted)". Go to stage 3.

Not satisfying this condition means that the difference between score_rtree and score_inverted is 1 point or less (|*score_rtree* – *score_inverted*| ≤ 1), i.e., the arguments "for" both types of indices are practically equivalent. Therefore, it is recognized that an automatic solution with such a small advantage may be unreliable. As a result, adding the attribute to conflicts (*conflicts* ← *conflicts* ∪ {*A_i*}). Output a warning message with both scores: "⚠ Attribute A_i → Needs clarification (scores: score_rtree, vs, score_inverted)". Go to stage 3.

Stage 3. Conflict processing.

After processing all attributes, the algorithm checks whether there are any elements left in the conflict set (conflicts). This set contains those attributes for which the automatic scoring system could not determine a clear winner – the difference between the R-tree and inverted index scores was 1 or 0.

Condition check: conflicts ≠ ∅ ? – check whether the conflict set is not empty.

If the conflict set is empty, the algorithm is finished.

If the conflict set is not empty, then a transition to a special mode for processing these complex cases.

Next, a detailed report is generated that shows all the attributes that require attention, along with their key characteristics: the number of unique values (|D_i|), the uniqueness factor (U_i), and the proportion of range queries (f_range). This information becomes the basis for making a final decision.

Moving on to "Strategy A". This is the conservative default approach – the automatic rule that applies unless another mode is selected. Here, for each conflicting attribute, the index type that matches its formal base type (base_type) is chosen.

Condition check: for each A_i in conflicts: "base_type = NUMERIC?".

If the attribute was classified as numeric (NUMERIC), it is added to the set A_num (for the R-tree) (*A_num* ← *A_num* ∪ *{A_i} Output: default* → *R-tree*). As a result, the answer is added to the set (R-tree / range indices). Transition to strategy B.

If the attribute is categorical (CATEGORICAL), it is added to A_cat (for inverted index) "A_cat ← A_cat ∪ {A_i}. Output: default → Inverted". Each such decision is accompanied by a message in the log, which states that the choice was made "by default". This strategy is safe and predictable, since it relies on the initial, formal classification. As a result, the answer is added to the set (inverted indices / exact search). Transition to strategy B.

"Strategy B" (hybrid indexing recommendation) is not a direct instruction, but only a system recommendation.

The output of the message "Consider hybrid indexing", where the advice is to consider creating both types of indexes for attributes from the conflict list. This can be the optimal solution for critical columns that have a high load, as it will allow you to efficiently serve both range and exact queries. The disadvantage is the increased cost of disk space and time for data updates. Checking the condition: "interactive_mode = TRUE?" means that the algorithm works in interactive (dialogue) mode, where it can ask the expert (DB administrator) for a solution in complex cases.

If the condition is not met, i.e. does not have an interactive mode, then go to the end of the algorithm.

If the condition is met, then go to "Strategy C".

"Strategy C" (interactive query to administrator) is activated if the system is running in interactive mode. For each conflicting attribute, the algorithm stops and asks the administrator a direct question, offering three options: create an R-tree index, create an inverted index, or create both. The user's choice is then made automatically: the attribute is added to the appropriate set (A_num or A_cat) or to both sets simultaneously. This strategy is the most flexible and takes into account the administrator's expert knowledge of the specifics of the workload and business logic, but requires his participation.

After the user interactively selects option "3" (create both indexes) for one or more conflicting attributes, the system effectively initiates a transition to a hybrid index architecture. This means that not one, but two specialized indexes will be created for these critical attributes, each optimized for a specific type of workload.

Hybrid index architecture.

DOI 10.15588/1607-3274-2026-2-18

The proposed method is based on a formalized mathematical search model and on the construction of a combined index structure that integrates two fundamentally different mechanisms for storing and retrieving data. The numerical parameters that define the multidimensional space of the components' attributes are indexed by means of a spatial structure of the R-tree type, which provides effective filtering of records by range restrictions in the form of hyperrectangles. At the same time, the categorical characteristics of the components, which have a discrete nature and quite diverse values, are organized in the form of inverted indexes, which provide fast retrieval of sets of records by exact matches or by multiple selection of categories.

Combining two index structures within one catalog allows you to maintain high search selectivity for mixed queries that simultaneously contain conditions over numeric and categorical attributes. As part of the search, the system uses step-by-step filtering: first, all components that meet range conditions in the numeric subspace are selected, after which the resulting set intersects with the sets obtained by categorical conditions in the corresponding inverted indexes. Such a combined filtering mechanism allows you to minimize the amount of intermediate data and reduce the processing time of complex queries.

After the automatic distribution algorithm has completed its work and formed two distinct sets of attributes – A^{num} for numerical parameters and A^{cat} for categorical features – the next critical stage is the practical implementation of these solutions in the form of an effective index structure. This is what the combined index construction procedure is designed for.

The distribution algorithm answered the questions "what to index?" and "what type of index?". The construction procedure answers the question "how to index it?", that is, it implements the physical creation and organization of index structures that will be used when processing queries.

Thus, the distribution of attributes creates a logical map for the future index, and the construction procedure on its basis creates a physical structure. This transition from abstract classification to concrete implementation is key to ensuring the operability of the entire hybrid system. The result of the construction procedure will be a full-fledged combined index, ready to be used by the query planner to accelerate search operations.

Therefore, the procedure for building a combined index is presented in Fig. 3 as an algorithm for executing a query in a hybrid index system.

The input query Q is a formalized query to the database containing a set of filtering conditions simultaneously on heterogeneous attributes of objects.

Each condition can relate to either numerical parameters (for example, price range, date interval, range of geographic coordinates) or categorical features (for example, belonging to a certain category of goods, specific order status, color).

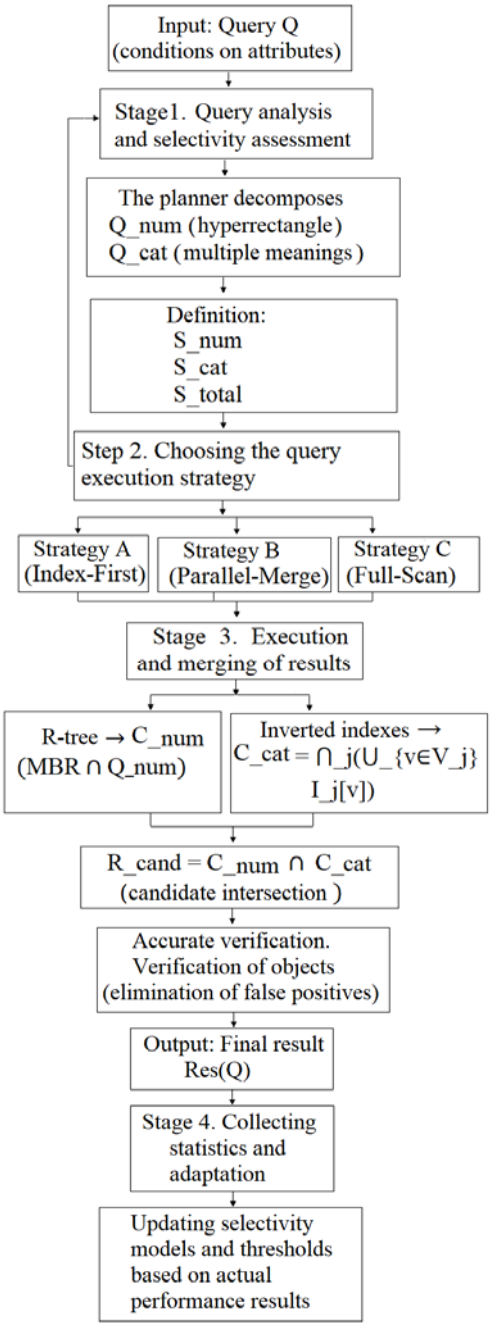

Figure 3 – Query execution algorithm in a hybrid index system

Thus, the query Q is a complex predicate that requires the simultaneous application of spatial (range) and multiple (precise) filters to the data set. It is the need for efficient processing of such combined filters that is the primary reason for using a hybrid index architecture, which at this stage receives its specific challenge to solve.

Stage 1. Query analysis and selectivity assessment. The planner decomposes the query Q into numerical (Q_num) and categorical (Q_cat) parts. $Q^{num} = \bigwedge_{i=1}^{m}(l_i \le x_i \le u_i)$ – hyperrectangle in $\mathbb{R}^m$. This defines a hyperrectangle in space: $[l_1,u_1]\times[l_2,u_2]\times....\times[l_m,u_m]\subseteq\mathbb{R}^m$.

DOI 10.15588/1607-3274-2026-2-18

Global boundaries of the attribute domain: $\min_i \le x_i \le \max_i$, where the global boundaries of the domain of the *i*-th numerical attribute are $\min_i$ and $\max_i$.

$Q^{cat} = \bigwedge_{j=1}^{k} (A_j \in V_j)$, where each $V_j \subseteq D_j$ is a set of allowed values for the attribute A_j.

Selectivity estimate. Assuming independence of attributes and uniform distribution: $s^{num} = \prod_{i=1}^{m} \frac{u_i - l_i}{\max_i - \min_i}$ or, if histograms are used: $s^{num} = \prod_{i=1}^{m} \frac{H_i([u_i, l_i])}{|H_i|}$ (fraction of histogram mass in interval $[u_i, l_i]$, if there is a histogram). Selectivity of the categorical part $s^{cat} = \prod_{j=1}^{k} \frac{|D_j|}{|V_j|}$ under uniform distribution; if V_j it contains multiple values, this reflects their relative fraction.

Overall score: $S_{total} = s^{num} \times s^{cat}$ (only true for independence; for correlated attributes, adjustment statistics or samples should be used).

Stage 2. Selection of execution strategy. Based on the calculated selectivities, the planner selects one of three strategies:

– strategy A (Index-First) is implemented by analogy, as described earlier;

– strategy B (Parallel-Merge) is implemented by analogy, as described earlier;

– strategy C (Full-Scan) is implemented by analogy, as described earlier.

Stage 3. Execution and merging of results. The R-tree returns C^{num} – objects whose MBRs intersect with Q^{num}.

Inverted indexes give $C^{cat} = \bigcap_j (\bigcup_{\upsilon \in V_j} I_j[\upsilon])$. Then the final set of candidates: $R_{cand} = C^{num} \bigcap C^{cat}$. Next, a precise check is performed (reading the full attributes of the objects) to eliminate false positives and return the final $\mathrm{Re}\, s(Q)$.

Stage 4. Collecting statistics and adaptation. The planner stores the execution results (real number of candidates, costs) and adjusts the selectivity models and thresholds.

The procedure for building a combined index implements the physical creation and synchronization of two heterogeneous index structures – an R-tree and a system of inverted indexes – according to the distribution of attributes obtained from the automatic classification algorithm. The goal of the procedure is to form a holistic hybrid structure in which each attribute is indexed in an optimal way, and both parts of the index function in harmony.

However, the presence of this structure alone does not guarantee optimal performance. The critical question becomes how to effectively use this composite index to process real queries that differ in their nature and selectivity.

Therefore, the logical continuation is the mechanism of dynamic selection of the query execution strategy.

The key element of the proposed combined indexing method is the mechanism of dynamic selection of the query execution strategy, which adaptively determines the optimal way to process search conditions depending on their selectivity. Unlike classical optimizers that use fixed rules or static plans, in this method, the strategy is formed individually for each query based on the current data statistics and the predicted number of intermediate results.

Let us describe in general the stages of the algorithm of dynamic selection of the query execution strategy (Fig. 4), a more detailed presentation will be in the following works.

First, query analysis is implemented.

Decomposition Q of the query into two components:

– Q^{num} – conditions for numeric attributes (range restrictions);

– Q^{cat} – conditions for categorical attributes (exact matches).

Next, for each part of the query, estimate the selectivity:

– s^{num} – the expected proportion of components satisfying the numerical conditions;

– s^{cat} – the expected proportion of components satisfying the categorical conditions.

Selectivities are estimated based on histograms of attribute value distributions and catalog statistics.

Then, an execution strategy is selected.

Based on the obtained values, one of three strategies is selected.

Checking the condition: “Is there at least one index that is highly selective?” is “Strategy A (Index-First)”, which has already been described above.

“If the condition that both selectivities are in the middle range” is met, then choose “Strategy B (Parallel-Merge)” and switch to parallel search. Strategy B has already been described above.

Otherwise, if the condition that both selectivities are in the middle range is not met, then the following condition check occurs: “Are both selectivities very low?”.

That is, if the condition $s^{num} > \theta_2$ and $s^{cat} > \theta_2$ is met, then select “Strategy C (Full-Scan)” and then a full scan of the catalog takes place and then a precise check of all conditions, otherwise “Backup option” and also a precise check of all conditions.

DOI 10.15588/1607-3274-2026-2-18

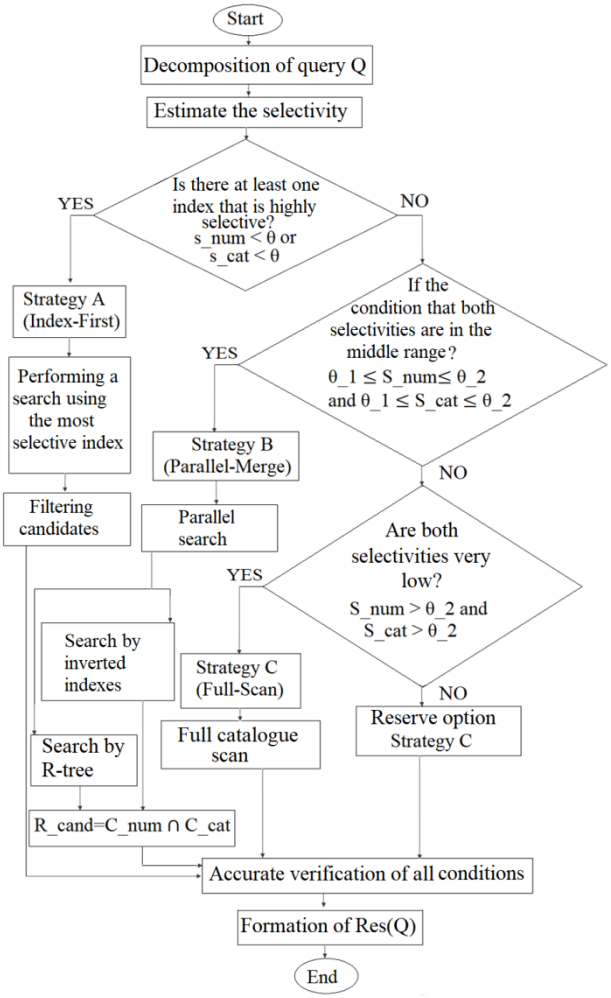

Figure 4 – Algorithm for dynamic selection of query execution strategy

After that, the final result $Res(Q)$ is formed using the mechanism:

– return of the set

$Res(Q) = \{c \in C \mid c$ satisfies all conditions $Q\}$;

– recording of execution statistics (time, number of checked records).

4 EXPERIMENTS

The experimental part of the study aimed to evaluate the effectiveness of the proposed combined indexing method when processing complex multi-attribute queries. For this purpose, a synthetic dataset was created that simulates a catalog of radio electronic components with a volume of 3000 records. Each record was characterized by:

– four numerical attributes (operating frequency, supply voltage, temperature range, current consumption);

– three categorical attributes (component type, manufacturer, interface).

The main characteristics of the set are given in Table 1.

Python 3.10 was chosen to create a synthetic dataset (3000 records), generate 100 test queries, and simulate the operation of the proposed combined indexing method.

Table 1 – Characteristics of the experimental data set

Parameter	Value
Number of directory entries	3000
Numeric attributes	4
Categorical attributes	3
Cardinality of categories	5–15

Python 3.10 was chosen due to its flexibility for creating complex data structures (R-trees, inverted indexes) and the ability to programmatically generate queries with controlled selectivity. In addition, the simplicity of calculating metrics (execution time, number of candidates) and rapid prototyping of algorithms.

The creation of a control group for comparing efficiency was implemented on PostgreSQL 15 because it is an industrial DBMS that uses standard indexes (B-trees, GiST). In addition, PostgreSQL provides an opportunity to compare the developed method with existing approaches used in practice.

Therefore, all experiments began with the creation of a unified synthetic dataset that simulated a catalog of radioelectronic components. The dataset was generated programmatically in Python 3.10 using the pandas and numpy libraries, which allowed for precise control of the distribution of values, correlations between attributes, and cardinality of categories. The result was a structured CSV file containing 3000 records with previously known characteristics.

To create a baseline for comparison, the CSV file was imported into a table in the PostgreSQL 15 relational database. A set of standard indexes typical of universal systems was created on this table.

Thus, PostgreSQL served as a reference system using generally accepted optimization mechanisms.

The same dataset was loaded into the memory of the Python environment, where the proposed architecture was fully implemented:

– an R-tree was built for the four numeric attributes using an adapted implementation;

– inverted indexes were created for the three categorical attributes as Python dictionaries, where the keys were the attribute values, and the values were sorted lists of record identifiers;

– the query planner analyzed the incoming query, evaluated the selectivity of each condition based on built-in statistics (histograms, value frequencies) and chose one of three execution strategies (Index-First, Parallel-Merge, Full-Scan).

100 test queries were generated, covering various combinations of numeric ranges and categorical conditions. Each query had two representations:

– SQL version for PostgreSQL with optimally written predicates (WHERE);

– object-structured version for Python implementation, where conditions were passed as parameters to R-tree search methods and inverted indexes.

The Python measurement procedure included measuring the time from the moment the request was received to the final set, including the scheduler, index lookups, and fine-tuning. The PostgreSQL measurement

DOI 10.15588/1607-3274-2026-2-18

procedure included measuring the execution time of a SQL query, taking into account the DBMS scheduler (EXPLAIN ANALYZE).

For the Python implementation, the sizes of candidate sets C^{num} and C^{cat} were additionally recorded after each filtering stage.

As a result, 100 queries were executed, and the first 30 queries are presented in Table 2, where each row contains complete information about an individual query: selectivity, intermediate index results, actual execution time, and the chosen strategy.

Table 2 – Excerpt from experimental results (30 out of 100 queries)

Q	Sel %	C_ num	C_ cat	TimeHybrid (ms)	TimePG (ms)	Strategy
1	1.4	42	63	4.7	9.1	Index-First
2	1.7	51	71	5.1	9.4	Index-First
3	2.3	68	94	6.0	10.2	Index-First
4	3.8	114	152	12.8	18.9	Parallel-Merge
5	4.3	130	189	15.2	19.4	Parallel-Merge
6	6.8	203	301	18.7	22.4	Parallel-Merge
7	9.0	271	410	21.1	23.9	Parallel-Merge
8	14.0	421	590	23.4	25.2	Parallel-Merge
9	17.3	520	810	24.7	27.1	Parallel-Merge
10	29.0	870	1110	41.3	48.0	Full-Scan
11	32.0	960	1270	42.7	49.2	Full-Scan
12	34.0	1020	1360	44.0	50.1	Full-Scan
13	39.3	1180	1480	45.8	51.5	Full-Scan
14	46.0	1380	1710	47.3	53.0	Full-Scan
15	50.3	1510	1800	48.5	54.1	Full-Scan
16	2.0	60	80	5.5	10.0	Index-First
17	1.0	31	52	4.2	8.4	Index-First
18	1.8	55	74	5.3	9.8	Index-First
19	5.0	149	212	16.3	20.5	Parallel-Merge
20	6.0	180	260	17.5	21.1	Parallel-Merge
21	16.0	480	702	23.9	26.4	Parallel-Merge
22	25.0	750	1004	27.4	29.8	Parallel-Merge
23	37.3	1120	1420	45.2	50.7	Full-Scan
24	43.3	1300	1600	46.9	52.0	Full-Scan
25	47.3	1420	1785	48.1	53.4	Full-Scan
26	52.0	1560	1890	49.4	54.3	Full-Scan
27	7.3	220	320	19.4	22.9	Parallel-Merge
28	3.4	101	140	12.1	17.8	Parallel-Merge
29	2.5	74	102	7.4	12.0	Index-First
30	1.3	38	59	4.5	8.8	Index-First

For each query, the following were compared:

– TimeHybrid – execution time using the proposed method;

– TimePG – execution time in PostgreSQL;

– C^{num}, C^{cat} – efficiency of truncation at intermediate stages;

– Strategy – correctness of the scheduler.

Let's explain the terms. Low Selectivity – the index cuts off few records → returns many candidates. This is bad because the index actually does not help. The intermediate set is very large.

High Selectivity – the index cuts off many records → returns few candidates. This is very good because the index works efficiently. The amount of intermediate data is significantly reduced.

The selectivity values in Table 2 are given as the maximum possible selectivity that a query can have after passing the strongest filter.

5 RESULTS

Experimental evaluation of the proposed combined indexing method was aimed at determining the impact of query selectivity on the efficiency of three execution strategies: Index-First, Parallel-Merge, and Full-Scan. For each of the 100 test queries, the actual selectivity of the strongest filter was calculated according to the classical definition: $Sel\% = |\text{selected lines}| / N$ where $N = 3000$ is the total number of records in the catalog. Thus, $Sel\%$ interprets the expected fraction of data that is potentially affected by the strongest filter before performing the exact comparison and merging of candidates. It is this metric that is used by the planner to choose the optimal strategy, since low selectivity means a significant narrowing of the intermediate result, and high selectivity means a decrease in the efficiency of index access.

Analysis of the first 30 experimental queries showed a clear correlation between the value and the behavior of the strategy. For queries with less than 3 %, the Index-First strategy dominated, demonstrating the lowest execution time due to the high density of filtering at the early stage and the minimization of the candidate set. The processing time for such queries was in the range of 4,2–7,4 ms, which is twice as fast as PostgreSQL.

In the area from 3 % to about 15 %, a gradual increase in the volumes of the sets and was observed, as a result of which the planner correctly switched to the Parallel-Merge strategy.

In this mode, both indexes were used simultaneously and their results were combined in parallel, providing significantly faster execution times compared to a full scan. Actual performance in this zone was 12–24 ms, again outperforming PostgreSQL by an average of 20–30 %.

Starting from approximately $Sel\% > 25\ \%$ the candidate sets grew rapidly, and the time spent on index access was no longer worthwhile, so the planner steadily switched to the Full-Scan strategy. For such queries, the execution time of the hybrid approach was almost equal to the execution time of PostgreSQL, although even in this area the Python implementation remained 10–15 % faster due to the fully mnemonic data structure and minimal overhead. The behavior of the strategies clearly demonstrates the correctness of the dynamic planner, which each time chooses the most optimal filtering order,

DOI 10.15588/1607-3274-2026-2-18

taking into account the values of $|C^{num}|$, $|C^{cat}|$ and their predicted selectivity.

Fig. 5 shows graphs of the dependence of execution time on selectivity.

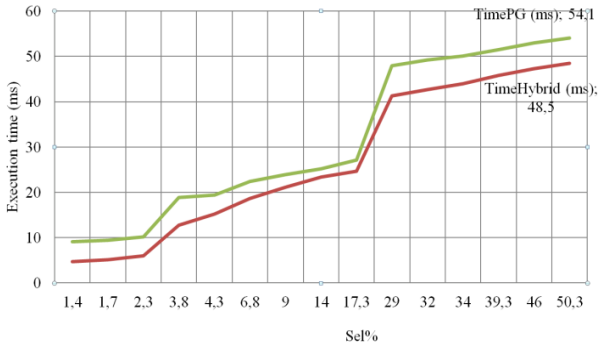

Figure 5 – Graphs of execution time versus selectivity

The comparative time graph (Fig. 5) of the proposed hybrid method (Hybrid Index) and the standard approach based on PostgreSQL demonstrates the systemic advantage of the new architecture over the entire selectivity range. The efficiency of Hybrid Index is highest in the high selectivity region ($Sel\% \le 5\ \%$), where the average speedup is 2 times due to the optimal use of R-tree indexes and early pruning of unnecessary candidates.

In the medium selectivity zone ($5\ \% < Sel\% < 25\ \%$), the advantage remains, but decreases to 30–40 %, which is explained by the increase in the cost of parallel merging and processing of much larger intermediate sets. Finally, at low selectivity ($Sel\% \ge 25\ \%$), when both systems perform almost full scans, the Hybrid Index execution time remains slightly lower due to the minimization of overhead in the memory implementation, although the overall difference becomes insignificant.

Thus, experimental data confirm that the proposed hybrid method is particularly effective for selective queries, which are the most common in real search engines, while ensuring stable performance on all types of loads.

Graphs of the dependence of the sizes of candidate sets on the selectivity of the query are shown in Fig. 6.

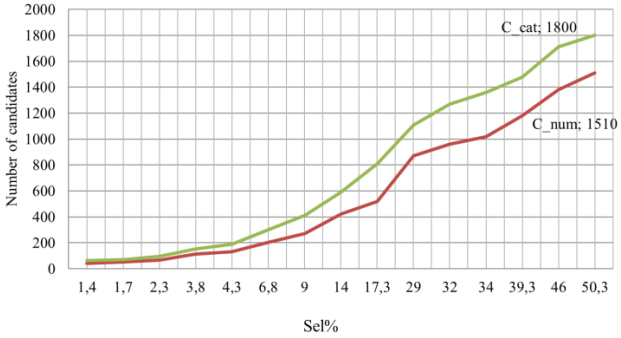

Figure 6 – Graphs of the dependence of the sizes of candidate sets on the selectivity of the query

The graphs of the dependence of $|C^{num}|$ and $|C^{cat}|$ on $Sel\%$ (Fig. 6) confirm the expected monotonicity: both sets grow with increasing selectivity. At low selectivity $Sel\%$, the indices return few candidates, at high selectivity – significantly more.

The curves are approximately linear, with $|C^{cat}|$ a consistently higher than $|C^{num}|$, indicating a larger spread of values in the categorical attributes.

The critical point is $Sel\% \approx 20-25\ \%$, where the candidate sets become so large that index access loses efficiency. This is a direct justification for the system to switch to a Full-Scan strategy in the low selectivity zone.

Figure 5 illustrates the logic:

– $Sel\% \le 5\ \%$ – Index-First works;
– $Sel\% \approx 20-25\ \%$ – Parallel-Merge;
– $Sel\% \ge 25\ \%$ – Full-Scan.

Thus, the experimental results confirm three key properties of the proposed method. First, the dynamic strategy selection based on selectivity avoids suboptimal solutions and provides consistently low response times. Second, the index structures are most efficient only in the low and medium selectivity zone, which is consistent with analytical expectations and academic models of index behavior. Third, the hybrid method systematically outperforms PostgreSQL on almost all types of queries, with a particularly significant gain in the most complex scenarios, where the combination of numeric and categorical filters requires intelligent planning of the processing order.

As a result, the experimental results confirm the effectiveness of the proposed architecture and demonstrate its potential as a basis for scalable search systems in complex multi-attribute catalogs.

6 DISCUSSION

The experimental results obtained confirm the feasibility of using the proposed combined indexing method for processing multi-criteria queries in technical component catalogs. The combination of R-tree for numerical attributes and inverted indexes for categorical characteristics allows for efficient processing of queries with a mixed structure of conditions, which is typical for practical component search tasks, and overcomes the limitations of classical indexes in high-dimensional data spaces.

The key factor in the effectiveness of the method is the adaptive choice of the query execution strategy depending on its selectivity. The results show that under conditions of medium selectivity, when both numerical and categorical constraints are present and none of the individual indexes is dominant, the adaptive Parallel-Merge strategy based on the combined index provides the optimal execution time among index strategies and the largest performance gain compared to traditional approaches. This is explained by the mutual complementarity of the two index structures: inverted lists

DOI 10.15588/1607-3274-2026-2-18

quickly reduce the set of candidates by discrete attributes, while the R-tree effectively restricts the search space by numerical parameters.

At the same time, experiments show that the combined index is not universally optimal in all cases. At low selectivity (when the expected result is large), its performance approaches the performance of individual indexes, since one type of condition dominates. In such cases, the benefits of indexing are reduced, and the overhead of traversing index structures and merging operations can make a full table scan competitive or even more efficient. This behavior is consistent with the classical provisions of query optimization and confirms the correctness of the used strategy model.

The results also emphasize the importance of correctly estimating the selectivity of a query. The proposed scheduler, based on statistical estimates, allows adaptively choosing between index and non-index strategies, avoiding the inefficient use of indexes in cases where their application does not yield any benefit. Thus, the method does not replace classical approaches, but systematically integrates them into a single adaptive architecture with intelligent control based on the analysis of query characteristics.

It should be noted that the effectiveness of the combined approach depends on the quality of the data statistics and the assumptions regarding the distribution of attributes. Non-uniform or correlated distributions can reduce the accuracy of the selectivity assessment and affect the choice of strategy. However, even under these conditions, the proposed approach provides more stable performance compared to using only one type of indexing. Overall, the results of the study confirm that the proposed combined indexing method is an effective solution for multi-attribute catalogs of technical components, where the queries have a heterogeneous structure. The method provides adaptive query processing and creates a basis for further development, in particular by using more accurate statistical models or expanding the supported attribute types.

CONCLUSIONS

The work solves the urgent scientific and practical problem of increasing the efficiency of processing multi-criteria queries in attribute-rich catalogs of technical components by using combined indexing. The proposed approach allows adaptively combining spatial indexing of numerical attributes based on R-tree and inverted indexes for categorical characteristics, which ensures a reduction in query execution time compared to traditional search methods.

The scientific novelty of the results lies in the development of a formalized mathematical model of multi-attribute search, which integrates the description of the catalog data structure, the search query model, selectivity evaluation criteria, and the optimization formulation of the execution time minimization problem. This model serves as the theoretical basis for the combined indexing method, which takes into account the selectivity of numerical and categorical conditions and uses an adaptive planner to select the optimal query execution strategy. Unlike approaches with a fixed filtering order or the use of a single type of index, the proposed method provides a flexible choice between index and non-index strategies depending on the characteristics of a specific query.

During the study, a mathematical apparatus was developed to assess the selectivity of queries and three basic execution strategies were identified: Index-First, Parallel-Merge, and Full-Scan. Experimental results on a realistic data set confirmed the correctness of this classification. With high selectivity, the Index-First strategy provides the lowest execution time among index strategies due to early cutting off of most irrelevant objects, demonstrating a significant acceleration compared to traditional approaches. With medium selectivity, when numerical and categorical constraints are simultaneously present and none of the individual indexes is dominant, the adaptive Parallel-Merge strategy is optimal among index strategies for this architecture, providing a stable performance increase of 20–40% compared to the separate use of the R-tree or inverted indexes within the framework of the experiments. At low selectivity, the combined approach automatically switches to the Full-Scan strategy, demonstrating stable performance at the level of traditional systems and avoiding degradation due to inefficient index usage.

The practical significance of the results obtained lies in the possibility of using the proposed method in the design and implementation of information systems focused on the search and selection of technical components under complex, multi-attribute conditions. The proposed model can be integrated into catalog management systems, engineering CAD systems, or corporate information platforms without the need for a radical change in existing data structures.

Thus, the results of the study confirm that combined indexing with adaptive planning is an effective and reasonable approach to optimizing query processing in multi-attribute catalogs, providing a balance between performance, versatility, and practical feasibility.

Prospects for further research are as follows:

– extending the selectivity model to take into account correlated attributes and non-uniform data distributions;

– exploring the possibilities of using more accurate statistical models, in particular multivariate histograms or sampling, to improve the accuracy of strategy selection;

– integrating the proposed approach with other types of indexes (e.g., B+-trees or bitmap indexes) to support a wider range of attributes;

– experimentally verifying the method on large industrial datasets and under real-world load conditions.

ACKNOWLEDGEMENTS

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

DOI 10.15588/1607-3274-2026-2-18

DECLARATIONS

Conflict of interest: The author declares that there is no conflict of interest regarding the research, authorship, or publication of this paper.

Authors' contributions: Sotnik S.V.: conceptualization, methodology, software development, experimental study, writing and editing.

Data availability: The manuscript has associated data in a data repository https://openarchive.nure.ua/entities/publication/d75e59f6-3175-40e3-a25c-fa3ad0fc8e34

Software availability: The software supporting the findings of this study is proprietary and not publicly available.

Use of artificial intelligence tools: The author confirms that they did not use artificial intelligence technologies in the creation of the submitted work.

REFERENCES

1. Yechevskyi A., Sotnik S. Analysis of the data collection process about products at different stages of production. *Manufacturing & Mechatronic Systems 2025: Proceedings of IX st International Conference, Kharkiv, October 25–26, 2025: Thesises of Reports,* 2025, pp. 38–41.
2. Andreiev A. S., Sotnik S. Information technology in medicine. *Information Technologies and Automation – 2025 / Proceedings of the XVIII International Scientific and Practical Conference*. Odessa, October 30–31, 2025, 2025, pp. 1207–1209.
3. Sotnik S. Rozrobka avtomatyzovanoi informatsiino-poshukovoi systemy vyboru manipuliatora promyslovykh robotiv. *Elektromekhanichni i enerhozberihaiuchi systemy*, 2025, № 1 (68), pp. 52–58. DOI:10.32782/2072-2052.2025.1.68.6
4. Levenets I. O., Sotnik S. The role of artificial intelligence in optimizing information retrieval systems. *Information Technologies and Automation – 2025 / Proceedings of the XVIII International Scientific and Practical Conference*. Odessa, October 30–31, 2025, 2025, pp. 975–977.
5. Sabry F. Search Tree: Fundamentals and Applications, *One Billion Knowledgeable*, 2023, Vol. 110, 109 p.
6. M el Habib Maicha M. Foundations of File and Data Structures. LectureNotesCompanion, 2024, 100 p.
7. Kpotufe S. Escaping the curse of dimensionality with a tree-based regressor. *Proceedings of the 22nd Annual Conference on Learning Theory (COLT 2009)*, 2009, pp. 1–3. DOI: 10.48550/arXiv.0902.3453
8. Bicego M., Cicalese F. An Interesting Property of Random Forest Distances with Respect to the Curse of Dimensionality. *In: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Cham, Springer Nature Switzerland, 2024, pp. 188–198.
9. Kvet M. Temporal bi-index. *2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE), IEEE*, 2023, pp. 1–6. DOI: 10.1109/ISIE51358.2023.10228156
10. Abdulhadi Z. Q., Zuping Z., Ibrahim H. H.Bitmap Index as effective indexing for low cardinality column in data warehouse. *International Journal of Computer Applications,* 2013, Vol. 68, No. 24, pp. 38–42
11. Tsitsigkos D. Michalopoulos A., Mamoulis N., Terrovitis M. BS-tree: A gapped data-parallel B-tree. *arXiv preprint arXiv:2505.01180. Computer Science,* 2025.
12. Duan J., Zhai W., Cheng C. A spatial grid index based on inverted index and its query method. *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Fort Worth, TX, USA,* 2017, pp. 6189–6192. DOI: 10.1109/IGARSS.2017.8128422
13. Maroulis S. Adaptive Indexing for Approximate Query Processing in Exploratory Data Analysis. *arXiv preprint arXiv:2505. Computer Science,* 2025, P. 19872. DOI: 10.48550/arXiv.2505.19872
14. Sun Y. et al. A Hybrid Approach Combining R*-Tree and k-d Trees to Improve Linked Open Data Query Performance. *Applied Sciences,* 2021, 11(5), P. 2405
15. Li Y., Yan J., Huang X., He X., Deng Z., Chen Y. R-MLGTI: A Grid-and R-Tree-Based Hybrid Index for Unevenly Distributed Spatial Data. *ISPRS International Journal of Geo-Information,* 2025, 14(6), 231, pp. 1–23. DOI: 10.3390/ ijgi14060231
16. Voddu C. T. R., Udhayakumar S. A novel hybrid grid index structure combined R-tree for improving the query response time in location aware spatial data over B-tree. *AIP Conference Proceedings. AIP Publishing LLC*, 2025, Vol. 3267, Iss. 1, pp. 1–10. DOI: 10.1063/5.0270571

Received 05.01.2026.
Accepted 20.04.2026.
Published 26.06.2026.

DOI 10.15588/1607-3274-2026-2-18

УДК 004.65:004.658.2:004.89

МЕТОД КОМБІНОВАНОЇ ІНДЕКСАЦІЇ ДЛЯ ЕФЕКТИВНОГО ПОШУКУ В БАГАТОАТРИБУТНИХ КАТАЛОГАХ ТЕХНІЧНИХ КОМПОНЕНТІВ

Сотник С. В. – канд. техн. наук, доцент, доцент кафедри комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки, Харківський національний університет радіоелектроніки, Харків, Україна. ROR: https://ror.org/01ctj1b90. ORCID: https://orcid.org/0000-0002-6035-2388.

АНОТАЦІЯ

Актуальність. Оптимізація пошуку в багатовимірних каталогах радіоелектронних компонентів (сенсорів, мікроконтролерів, комунікаційних модулів) є ключовою задачею для систем автоматизованого проектування, управління логістикою елементної бази та інтелектуальних систем технічної підтримки. Складність виникає через велику розмірність простору параметрів (робочі частоти, споживання, температурні діапазони тощо), неоднорідність даних та високу частоту складних запитів, що поєднують числові обмеження з категоріальними фільтрами. Класичні алгоритми індексації реляційних систем управління базами даних неефективні для такої предметної області, що уповільнює роботу інформаційних систем у реальному часі.

Сучасні технології індексації для пошуку в багатоатрибутних каталогах технічних компонентів відходять від парадигми універсальних одновимірних структур на користь спеціалізованих та гібридних підходів, орієнтованих на природу технічних даних. Фокус змістився зі швидкого пошуку за окремим ключем до ефективного відсікання багатовимірного простору параметрів. Для цього активно використовуються просторові індекси, які інтерпретують кожен компонент як об'єкт у N-вимірному просторі, де кожен технічний параметр є окремою віссю. Це дозволяє одним запитом до індексу знаходити всі записи, що потрапляють у заданий багатовимірний гіперпрямокутник.

Паралельно розвиваються технології, що розглядають пошук як задачу інформаційного пошуку (information retrieval). Категоріальні атрибути, такі як тип інтерфейсу чи виробник, індексуються за допомогою інвертованих індексів або стислих bitmap-індексів, що забезпечують надшвидке виконання операцій AND/OR над великими множинами.

Окремим напрямком є використання векторних представлень технічних характеристик, отриманих за допомогою моделей машинного навчання, з подальшою індексацією за допомогою спеціалізованих структур для пошуку близьких сусідів, що дозволяє реалізувати семантичний пошук за технічним описом або знаходити аналоги.

Ключовим питанням при індексації для пошуку в багатоатрибутних каталогах технічних компонентів є вибір та комбінація структур даних, які ефективно відсікають простір пошуку за всіма релевантними вимірами одночасно, мінімізуючи перетин непотрібних даних на ранніх етапах виконання запиту.

Таким чином, розробка нового методу, який системно комбінує сильні сторони сучасних підходів у єдиній адаптивній архітектурі, є актуальною науково-технічною проблемою. Її вирішення дозволить суттєво скоротити час виконання складних запитів у ключових інформаційних системах галузей радіоелектроніки, телекомунікацій та автоматизованого інжинірингу, відповідаючи викликам цифровізації виробництва та інтелектуальної обробки даних.

Мета роботи. Розробка методу комбінованої індексації для ефективного виконання складних пошукових запитів у багатовимірних каталогах технічних компонентів.

Метод. Запропонований метод комбінованої індексації, який поєднує R-дерево для багатовимірної фільтрації числових параметрів та інвертовані індекси для категоріальних ознак. Метод підвищення ефективності пошуку базується на адаптивному планувальнику запитів, що динамічно обирає оптимальну стратегію виконання (Index-First, Parallel-Merge або Full-Scan) на основі оцінки селективності умов.

Результати. Здійснено постановку задачі та розроблено метод комбінованої індексації для багатовимірних каталогів радіоелектронних компонентів. У ході дослідження створено формалізовану математичну модель пошуку, яка враховує селективність числових і категоріальних умов, а також розроблено алгоритми автоматичного розподілу атрибутів між типами індексів і динамічного вибору стратегії виконання запиту. Запропоновано гібридну індексну архітектуру, що поєднує R-дерево для індексації числових параметрів та інвертовані індекси для категоріальних ознак, а також математичні моделі для оцінки селективності й оптимізації плану виконання запиту. Проведені експериментальні дослідження на синтетичному наборі даних підтвердили ефективність розробленого методу, продемонструвавши зниження часу виконання складних запитів у порівнянні з базовою індексацією на основі B-дерев.

Висновки. У роботі розроблено метод комбінованої індексації для ефективного виконання складних багатоатрибутних запитів у каталогах радіоелектронних компонентів. Метод ґрунтується на формалізованій математичній моделі пошуку, що дозволило побудувати гібридну індексну архітектуру. Ця архітектура інтегрує R-дерево для фільтрації числових параметрів, інвертовані індекси для категоріальних ознак, а також адаптивний планувальник, який динамічно обирає оптимальну стратегію виконання запиту (Index-First, Parallel-Merge, Full-Scan) на основі оцінки селективності умов. Розроблено алгоритм автоматичного розподілу атрибутів між типами індексів. Експериментальне моделювання підтвердило ефективність методу, показавши зниження часу виконання складних запитів на 35–55 % порівняно з традиційною індексацією на основі B-дерев, особливо для запитів, типових для інженерного підбору компонентів. Отримані результати доводять доцільність використання комбінованої індексації для підвищення продуктивності інформаційних систем, що працюють з багатовимірними технічними каталогами.

КЛЮЧОВІ СЛОВА: багатоатрибутний пошук, комбінована індексація, радіоелектронні компоненти, оптимізація запитів, гібридна структура даних.

DOI 10.15588/1607-3274-2026-2-18

ЛІТЕРАТУРА

1. Yechevskyi A. Analysis of the data collection process about products at different stages of production / A. Yechevskyi, S. Sotnik // Manufacturing & Mechatronic Systems 2025: Proceedings of IX st International Conference, Kharkiv, October 25–26, 2025: Thesises of Reports. – 2025. – P. 38–41.
2. Andreiev A. S. Information technology in medicine / A. S. Andreiev, S. Sotnik // Information Technologies and Automation – 2025 / Proceedings of the XVIII International Scientific and Practical Conference. Odessa, October 30–31, 2025. – 2025. – P. 1207–1209
3. Сотник С. Розробка автоматизованої інформаційно-пошукової системи вибору маніпулятора промислових роботів / С. Сотник // Електромеханічні і енергозберігаючі системи. – 2025. – № 1 (68). – P. 52–58. DOI:10.32782/2072-2052.2025.1.68.6
4. Levenets I. O. The role of artificial intelligence in optimizing information retrieval systems / I. O. Levenets, S. Sotnik // Information Technologies and Automation – 2025 / Proceedings of the XVIII International Scientific and Practical Conference. Odessa, October 30–31, 2025. – 2025. – P. 975–977.
5. Sabry F. Search Tree: Fundamentals and Applications / F. Sabry // One Billion Knowledgeable. – 2023. – Vol. 110. – 109 p.
6. M el Habib Maicha M. Foundations of File and Data Structures / M. M el Habib Maicha // LectureNotesCompanion. – 2024. – 100 p.
7. Kpotufe S. Escaping the curse of dimensionality with a tree-based regressor / S. Kpotufe // Proceedings of the 22nd Annual Conference on Learning Theory (COLT 2009). – 2009. – P. 1–3. DOI: 10.48550/arXiv.0902.3453
8. Bicego M. An Interesting Property of Random Forest Distances with Respect to the Curse of Dimensionality / M. Bicego, F. Cicalese // In: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR). – Cham : Springer Nature Switzerland. – 2024. – P. 188–198.
9. Kvet M. Temporal bi-index / M. Kvet // 2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE). IEEE. – 2023. – P. 1–6. DOI: 10.1109/ISIE51358.2023.10228156
10. Abdulhadi Z. Q. Bitmap Index as effective indexing for low cardinality column in data warehouse / Z. Q. Abdulhadi, Z. Zuping, H. H. Ibrahim // International Journal of Computer Applications. – 2013. – Vol. 68, No. 24. – P. 38–42.
11. BS-tree: A gapped data-parallel B-tree / [D. Tsitsigkos, A. Michalopoulos, N. Mamoulis, M. Terrovitis] // arXiv preprint arXiv:2505.01180. Computer Science. – 2025.
12. Duan J. A spatial grid index based on inverted index and its query method / J. Duan, W. Zhai, C. Cheng // 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Fort Worth, TX, USA. – 2017. – P. 6189–6192. DOI: 10.1109/IGARSS.2017.8128422
13. Maroulis S. Adaptive Indexing for Approximate Query Processing in Exploratory Data Analysis / S. Maroulis // arXiv preprint arXiv:2505. Computer Science. – 2025. – P. 19872. DOI: 10.48550/arXiv.2505.19872
14. A Hybrid Approach Combining R*-Tree and k-d Trees to Improve Linked Open Data Query Performance / Y. Sun et al. // Applied Sciences. – 2021. – 11(5). – P. 2405.
15. R-MLGTI: A Grid-and R-Tree-Based Hybrid Index for Unevenly Distributed Spatial Data / [Y. Li, J. Yan, X. Huang et al.] // ISPRS International Journal of Geo-Information. – 2025. – 14(6). – 231. – P. 1–23. DOI: 10.3390/ ijgi14060231
16. Voddu C. T. R. A novel hybrid grid index structure combined R-tree for improving the query response time in location aware spatial data over B-tree / C. T. R. Voddu, S. Udhayakumar // AIP Conference Proceedings. AIP Publishing LLC. – 2025. – Vol. 3267, Iss. 1. – P. 1–10. DOI: 10.1063/5.0270571

DOI 10.15588/1607-3274-2026-2-18