ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov840

Implementation of Service Oriented Architecture for Data Management Using Binary Search Trees

Zeinebou Mint Mohamed Lemin[1]

[1]Nouakchott, Mauritania

Publication Date: 2026/11/23

Abstract : The growth of digital data requires structures that support efficient insertion, retrieval and ordered traversal. This article presents a service oriented architecture in which each data service manages a binary search tree (BST), also called an arbre binaire de recherche (ABR). Records are organized as nodes containing a key, a value and links to left and right child nodes. Service contracts define key ranges and operations, while a service bus routes requests and coordinates results. The proposed design is described through node structures, insertion and search algorithms, and an illustrative example. Its efficiency depends on tree height, partitioning and available storage; it does not by itself establish unlimited scalability.

Keywords: *Big Data; Binary Search Tree; BST; ABR; Node; Subtree; SOA.*

How to Cite: Zeinebou Mint Mohamed Lemin (2026) Implementation of Service Oriented Architecture for Data Management Using Binary Search Trees. *International Journal of Innovative Science and Research Technology,* 11(11), 1176-1178. https://doi.org/10.38124/ijisrt/26nov840

I. INTRODUCTION

A binary search tree (BST, or ABR in French) organizes records by comparable keys. Each node stores a key, its associated data and at most two child links. All keys in its left subtree are smaller than its key, and all keys in its right subtree are greater. The first node is the root, child nodes form subtrees, and nodes with no children are leaves. This ordering guides insertion and search by comparing a requested key with one node at a time (Fig. 1). In this article, duplicate keys update the value of the existing node.

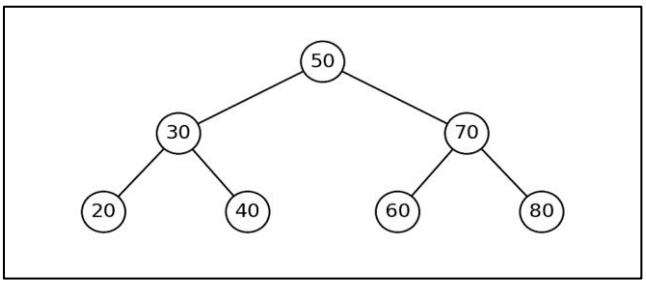


Fig 1 Structure of a Binary Search Tree

Data are represented by linked nodes rather than fixed-size storage blocks. A node contains the fields key, value, left and right. An empty child link is represented by NULL. Searching begins at the root: an equal key identifies the record; a smaller key selects the left child; a larger key selects the right child. Reaching NULL means that the key is absent. Insertion follows the same comparisons and attaches a new leaf at the first empty link. An in-order traversal visits the left subtree, the node and the right subtree to return records in increasing key order.

The structure distinguishes the root, internal nodes, child nodes, subtrees and leaves (Fig. 2). Each node can store a record or a reference to data maintained by its service. A search or insertion takes O(h) time, where h is the tree height. A balanced tree has height O(log n), whereas an ordinary unbalanced BST can become a chain and require O(n) time. Node links do not provide replication, persistence or fault recovery automatically; these functions must be implemented separately when required. Very large datasets therefore require explicit storage, balancing and partitioning policies.

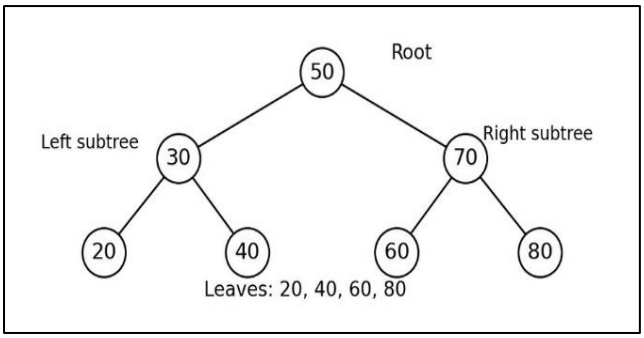


Fig 2 Root, Child Nodes, Subtrees and Leaves

II. MOTIVATION

The problem of Big Data is present in different domains. It is not social networks or large commercial enterprises only, but also in other areas such as sport, health, education, the Telecom Operators ….etc.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov840

The proposed design organizes data through several binary search trees exposed as independent services. Each service owns a defined key range and maintains its own root and child links. The service oriented architecture (SOA) specifies common message formats and operation contracts so that these services can exchange requests and results. This separation supports modular data management, while its usefulness at large scale depends on partition balance, storage capacity and communication costs.

➢ *Related Work*

Research on large-scale data processing provides the background for the original study. References [1]–[4] are retained as contextual sources on distributed processing; they are not evidence that the binary search tree design proposed here has been implemented or benchmarked. The growth of data motivates the search for suitable management structures [5]. Service oriented architecture offers loosely coupled components with defined service interfaces [6]. The present proposal combines that architectural approach with an ordered node structure: a service manages a BST and exposes insertion, search and traversal operations. The design described below is conceptual, and no measured performance comparison is claimed.

➢ *Contribution*

The contribution is a service oriented design built around binary search trees. Each service maintains nodes ordered by key and exposes operations through a common contract. The description specifies the node fields, request routing, insertion and retrieval algorithms, and the aggregation of ordered results. It also identifies the effect of tree height and service partitioning on performance. The proposal does not assume that a single tree, or a collection of trees, can manage arbitrary data volumes without additional engineering.

III. DESCRIPTION

Service oriented architecture can use an enterprise service bus or a message broker to route requests and coordinate services [7]. In the proposed design, the consumer sends an operation, a key and, for insertion, a value. A routing directory maps non-overlapping key ranges to services. Each service contract defines its range, comparison rule, duplicate-key policy, message format and supported operations. The service owns a root reference and nodes with key, value, left and right fields. Requests are executed by following child links within that service's tree; these links are logical data relationships, not network connections. Range queries that cross service boundaries are sent to each relevant service and their ordered results are merged. Persistence, concurrent access control, recovery and balancing must be defined by the implementation.

➢ *Node Structure and Implementation*

A Node contains key, value, left and right. Both child links are initially NULL. A leaf has no children. Each service initializes root to NULL and uses the following language-independent pseudocode. Begin and End delimit each algorithm.

```
Algorithm Insert(node, key, value)
Begin
  If node = NULL then
    Return new Node(key, value, NULL, NULL)
  End If
  If key < node.key then
    node.left ← Insert(node.left, key, value)
  Else If key > node.key then
    node.right ← Insert(node.right, key, value)
  Else
    node.value ← value
  End If
  Return node
End
Algorithm Search(node, key)
Begin
  current ← node
  While current ≠ NULL do
    If key = current.key then
      Return current.value
    Else If key < current.key then
      current ← current.left
    Else
      current ← current.right
    End If
  End While
  Return NOT_FOUND
End
Algorithm HandleRequest(operation, key, value)
Begin
  service ← Directory.FindOwner(key)
  If service = NULL then Return RANGE_ERROR
  If operation = INSERT then
    service.root ← Insert(service.root, key, value)
    Return SUCCESS
  Else If operation = SEARCH then
    Return Search(service.root, key)
  End If
  Return UNSUPPORTED_OPERATION
End
```

IV. RESULTS AND ANALYSIS

The figures illustrate the proposed architecture rather than experimental results. Each data service owns an independent binary search tree; the service bus routes requests using the key-range directory (Fig. 3). Within each tree, comparisons select the left or right subtree until the requested node or an empty link is reached. For example, inserting keys 50, 30, 70, 20, 40, 60 and 80 creates a root of 50, children 30 and 70, and leaves 20, 40, 60 and 80. A search for 60 visits 50, then 70, then 60. An in-order traversal returns 20, 30, 40, 50, 60, 70, 80.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov840

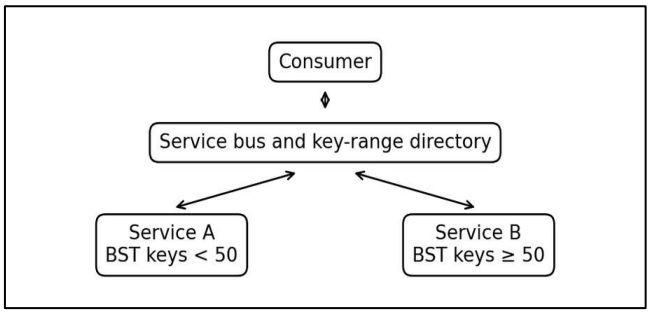


Fig 3 Service Oriented Architecture with Binary Search Tree Services

The bus selects the service that owns the requested key and returns that service's response (Fig 4). For insertion, the service updates its root reference after the algorithm returns. A new key becomes a leaf, while an existing key keeps its position and receives an updated value. A new operation requires an explicit contract update; the bus does not change contracts automatically. The local cost of insertion and search is O(h), in addition to routing and communication overhead. No timing measurements or capacity guarantees are reported.

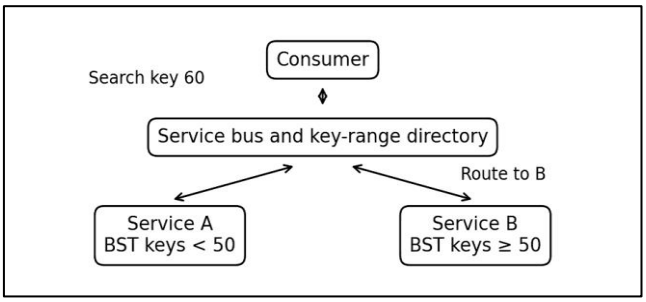


Fig 4 Routing a Request to the Appropriate Binary Search Tree

V. CONCLUSIONS AND FUTURE WORK

This article proposes a service oriented architecture in which binary search trees organize data as a root, internal nodes, left and right subtrees, and leaves. Insertion and search follow key comparisons, and in-order traversal produces sorted results. The service bus routes requests to services with defined key ranges. The design supports modular organization, but its performance depends on tree height, data distribution and storage choices. Future work includes a working prototype, balancing, persistence, concurrency control, recovery and experiments on representative datasets.

ACKNOWLEDGEMENT

We would like to thank the reviewers for their valuable comments, which helped us to revise the paper thoroughly.

REFERENCES

[1]. C. Uzunkaya, T. Ensari, and Y. Kavurucu, "Hadoop Ecosystem and Its Analysis on Tweets," Procedia - Soc. Behav. Sci., vol. 195, pp. 1890–1897, 2015.

[2]. O. Romero, V. Herrero, A. Abelló, and J. Ferrarons, "Tuning small analytics on Big Data: Data partitioning and secondary indexes in the Hadoop ecosystem," Inf. Syst., pp. 1–21, 2014.

[3]. A. Ben Ayed, M. Ben Halima, and A. M. Alimi, "MapReduce Based Text Detection in Big Data Natural Scene Videos," Procedia Comput. Sci., vol. 53, pp. 216–223, 2015.

[4]. A. O'Driscoll, V. Belogrudov, J. Carroll, K. Kropp, P. Walsh, P. Ghazal, and R. D. Sleator, "HBLAST: Parallelised sequence similarity - A Hadoop MapReducable basic local alignment search tool.," J. Biomed. Inform., vol. 54, pp. 58–64, 2015.

[5]. S. Erevelles, N. Fukawa, and L. Swayne, "Big Data consumer analytics and the transformation of marketing," J. Bus. Res., vol. 69, no. 2, pp. 897–904, 2015.

[6]. M. B. Carvalho, F. Bellotti, R. Berta, A. De Gloria, G. Gazzarata, J. Hu, and M. Kickmeier-Rust, "A case study on Service-Oriented Architecture for Serious Games," Entertain. Comput., vol. 6, pp. 1–10, 2015.

[7]. A. Cameron, M. Stumptner, N. Nandagopal, W. Mayer, and T. Mansell, "Rule-based peer-to-peer framework for decentralised real-time service oriented architectures," Sci. Comput. Program., vol. 97, Part 2, no. 0, pp. 202–234, 2015.