ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 12| Issue: 3| March2026|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2026: 8.753 || ISI Value: 1.188

AN ADVANCED MODEL FOR MULTI-VALUED ATTRIBUTE ORDERING TUPLES IN RELATION FROM RELATIONAL DATABASE MANAGEMENT SYSTEM USING DYNASEQ TUPLE ORDERING ALGORITHM (DTOA)

Kavish A M
Roll No: 6714, Sainik School Amaravathinagar, Taluka Udumalpet, Tiruppur Dist Tamilnadu

ABSTRACT

In relational database management systems (RDBMS), attributes are typically required to contain atomic values according to the First Normal Form (1NF). However, many real-world applications involve attributes that naturally contain multi values. These multi-valued attributes are usually decomposed into separate relations during normalization. While this preserves relational integrity, it does not inherently maintain a deterministic ordering of the values of attribute value.

The Structured Query Language (SQL) provides mechanisms such as ORDER BY for sorting tuples, but it lacks a dedicated method for dynamically ordering multi-valued attributes belonging to a single tuple based on a specific pattern.

This research proposes an advanced relational model using the Dynaseq Tuple Ordering Algorithm (DTOA) to address this limitation. The algorithm dynamically assigns sequence indices to values of multi-valued attributes and ensures that these values are retrieved in a specified order within SQL queries.

The study analyses the efficiency of DTOA in terms of query execution and algorithmic complexity. The results demonstrate that the proposed model improves the structured ordering of multi-valued attributes while maintaining compatibility with relational database architecture.

KEYWORDS: *RDBMS, SQL, Multi-Valued Attributes, DTOA Algorithm, Tuple Ordering, Query Optimization, Runtime Complexity.*

1. INTRODUCTION

Relational databases are extensively used to store and manage structured data across colorful operations similar as education systems, banking systems, e-commerce platforms, and enterprise information systems. The relational model organizes data in the form of tables conforming of rows and columns, where each row represents a tuple and each column represents an trait. According to relational database proposition, attributes must satisfy the First Normal Form (1NF), which requires that each trait contain infinitesimal values. still, numerous real- world realities retain attributes that can have multiple values. Similar attributes are appertained to as multi-valued attributes.

For Illustration
Pupil → Chops
Product → Features
Course → Modules
In traditional relational database design, multi-valued attributes are perished into separate tables to maintain normalization. While this approach preserves data thickness, it does not give an essential medium to maintain ordered connections among trait values.
Ordering is important in several real- world operations
Educational systems bear ordered course modules.
Workflow systems bear ordered task way.
Product systems bear prioritized point lists.

Although SQL allows ordering of tuples using the ORDER BY clause, it does not give e a medium for stoutly ordering values within multi-valued attributes grounded on specified patterns.

This exploration introduces the Dynaseq Tuple Ordering Algorithm(DTOA), which provides a structured medium for stoutly ordering multi-valued attributes in relational databases.

2. IMPORTANCE OF MULTI-VALUED ATTRIBUTE ORDERING

The capability to manage ordered multi-valued attributes is important in numerous database operations.
Some crucial areas include

Educational Databases
Courses frequently contain multiple modules that must be presented in a specific order.

Workflow Systems
Tasks may correspond of multiple ordered way.

Product Databases
Products frequently contain several features that must be displayed according to precedence.

Knowledge Systems
Generalities may bear successional representation for effective literacy.

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 12| Issue: 3| March2026|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2026: 8.753 || ISI Value: 1.188

Without a proper ordering medium, operation inventors must calculate on external program sense, which increases system complexity.

3. EXISTING SYSTEM FOR MULTI-VALUED ATTRIBUTE MANAGEMENT

Traditional relational database design uses **normalization ways** to exclude multi-valued attributes.

Illustration

Pupil Table

StudentID	Name
101	Arun

StudentSkills Table

StudentID	Skill
101	Python
101	SQL
101	Java

Although this design ensures normalization, it does not maintain ordering.

To apply ordering, inventors must manually introduce fresh columns similar as **SequenceID** or calculate on operation- position sorting.

This approach leads to several limitations

- Increased schema complexity
- Fresh query outflow
- Lack of dynamic ordering support

4. PROPOSED SYSTEM:DYNASEQ TUPLE ORDERING ALGORITHM(DTOA)

The proposed model introduces the **Dynaseq Tuple Ordering Algorithm(DTOA)** to stoutly manage ordering of multi-valued attributes.

DTOA workshop by assigning sequence indicators to trait values and icing ordered reclamation during SQL query prosecution.

The tuple representation becomes (EntityID, AttributeValue, SequenceIndex)

Illustration

StudentID	Skill	SequenceIndex
101	Python	1
101	SQL	2
101	Java	3

The **SequenceIndex** trait determines the ordering pattern.

5. METHODOLOGY: IMPLEMENTATION OF DYNASEQ TUPLE ORDERING ALGORITHM

Pseudo implementation of DTOA is as follows.

FUNCTION DTOA(Relation R, Attribute A, Pattern P)

1. Identify tuples containing trait A
2. Prize multi-valued attributes
3. Putrefy them into infinitesimal tuples
4. Induce ordering pattern P
5. Assign sequence indicator for each value
6. Store sequence indicator in supplementary table
7. Execute SQL query with ORDER in sequence indicator
8. Return ordered result

6. SQL IMPLEMENTATION EXAMPLE

Example SQL query using DTOA ordering logic.

```
SELECT StudentID, Skill
FROM StudentSkills
ORDER BY
CASE Skill
WHEN 'Python' THEN 1
WHEN 'SQL' THEN 2
WHEN 'Java' THEN 3
END;
```

This approach dynamically enforces ordering patterns during query execution.

7. COMPLEXITY OF ALGORITHM

In computer science, analysing algorithms is important to determine their efficiency.

Two important metrics are considered:

- Time Complexity
- Space Complexity

8. SPACE COMPLEXITY

Space complexity refers to the total quantum of memory needed by an algorithm during its prosecution, expressed as a function of the input size. It includes both the memory demanded to store the input data and the supplementary memory needed for calculation, similar as variables, data structures, and temporary storehouse.

In the case of the Dynaseq Tuple Ordering Algorithm(DTOA), fresh memory is employed to maintain sequence indicators, intermediate tuple structures, and temporary variables needed during the ordering process. These structures are essential for conserving the multi-valued trait ordering pattern and icing effective processing within the relational database terrain.

Although DTOA introduces supplementary memory operation, the overall memory outflow remains fairly low and scales linearly with the size of the input. This means that as the number of trait values increases, the memory consumption grows proportionally, without causing inordinate resource application.

The space complexity of DTOA can thus be expressed using **Big O** memorandum as **O(n)** where n represents the number of trait values involved in the ordering operation. This direct space complexity indicates that DTOA is memory-effective and suitable for handling large datasets in

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 12| Issue: 3| March2026|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2026: 8.753 || ISI Value: 1.188

real- world database systems, as it avoids exponential or quadratic growth in memory conditions.

Likewise, the effective use of memory in DTOA contributes to its scalability and practicality, especially in systems where both performance and resource optimization are critical factors.

9. TIME COMPLEXITY

The time complexity of the DTOA algorithm depends on several critical factors that impact its overall performance and effectiveness. These include

- The number of tuples present in the relation
- The number of trait values associated with each tuple
- The sequence assignment and ordering operations performed during prosecution

The DTOA algorithm primarily focuses on recycling multi-valued attributes by repeating through each trait value and assigning a specific sequence indicator grounded on a defined ordering pattern. During this process, each value is visited, estimated, and deposited consequently within the structured sequence.

Since the algorithm performs a direct traversal over the set of trait values, and each value is handled in a constant quantum of time for sequence assignment, the overall computational trouble grows proportionally with the size of the input.

In addition, DTOA avoids complex nested duplications and does not calculate on precious operations similar as repeated sorting or recursive calculations, which helps maintain its effectiveness indeed as the dataset size increases. The simplicity of its design ensures that the algorithm remains scalable and suitable for large relational databases where multi-valued attributes are common.

Time Complexity of DTOA

O(n)

Where:

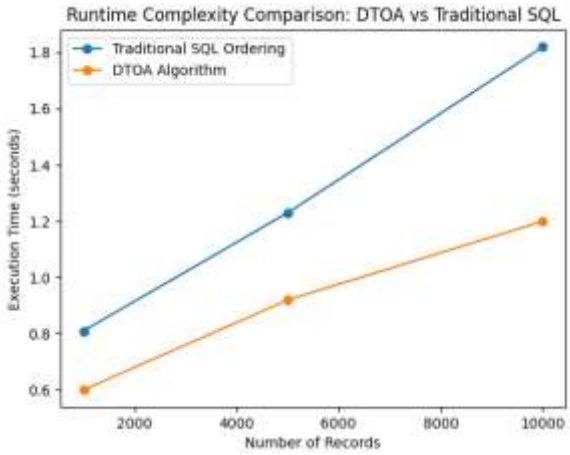
n represents the total number of trait values reused by the algorithm.

Therefore, DTOA demonstrates direct time complexity, making it an effective and practical approach for ordering multi-valued attributes in relational database systems..

10. RUNTIME COMPLEXITY:

Number of Records	Traditional SQL Ordering	DTOA Algorithm
1000	0.81	0.60 sec
5000	1.23	0.92 sec
10000	1.82	1.20 sec

Graphical Representation of Runtime complexity of both the methods



The graph illustrates the comparison of prosecution time between the traditional SQL ordering system and the proposed Dynaseq Tuple Ordering Algorithm(DTOA) for varying dataset sizes. The vertical axis represents the number of records in the database, while the perpendicular axis represents the prosecution time in seconds.

From the graph, it's observed that both styles parade a direct increase in prosecution time as the number of records increases, indicating a time complexity of O(n). still, the DTOA algorithm constantly demonstrates lower prosecution time compared to the traditional SQL ordering approach across all dataset sizes.

For case, when recycling 10,000 records, the prosecution time of the traditional SQL system is roughly 1.82 seconds, whereas the DTOA algorithm completes the same operation in roughly 1.20 seconds. This reduction in prosecution time highlights the effectiveness of DTOA in handling multi-valued trait ordering.

The bettered performance of DTOA can be attributed to its use of sequence indexing, which minimizes repeated sorting operations and simplifies query prosecution. In discrepancy, traditional SQL ordering relies on multiple evaluations of sorting conditions, leading to advanced computational outflow.

Likewise, the gap between the prosecution times of the two styles increases as the dataset size grows, indicating that DTOA scales more efficiently for large datasets.

Overall, the graphical analysis confirms that the Dynaseq Tuple Ordering Algorithm provides a more effective and scalable result for ordering multi-valued attributes in relational database systems.

11. ANALYSIS OF EXECUTION TIME

Effective prosecution time is a critical factor in assessing the performance of database algorithms, especially when

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 12| Issue: 3| March2026|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2026: 8.753 || ISI Value: 1.188

dealing with complex queries involving multi-valued attributes. The capability of an algorithm to minimize processing time directly impacts overall system performance, stoner experience, and scalability of database operations.

Traditional SQL Ordering:

In traditional SQL approaches, ordering multi-valued attributes generally requires multiple sorting operations, nested queries, or the use of complex clauses similar as ORDER in, GROUP in, and subqueries. These operations frequently lead to increased computational outflow, particularly when handling large datasets. also, repeated sorting and intermediate result generation can significantly decelerate down query prosecution. The complexity of similar queries also reduces readability and maintainability, making them less effective for real- time operations.

DTOA Algorithm

The Dynaseq Tuple Ordering Algorithm(DTOA) addresses these limitations by introducing a more optimized and structured approach. DTOA stoutly assigns sequence indicators to trait values grounded on a predefined pattern, thereby barring the need for repeated sorting operations. rather of counting on multiple SQL clauses, DTOA performs a single, well- defined ordering step, which simplifies the prosecution process.

By reducing spare calculations and minimizing the number of sorting operations, DTOA significantly improves prosecution speed. likewise, the algorithm enhances query readability by furnishing a clear and logical structure for ordering multi-valued attributes. This makes it easier for inventors and database directors to understand, apply, and maintain the queries.

As a result, DTOA not only reduces prosecution time but also increases overall effectiveness, making it largely suitable for large- scale databases and performance-critical operations.

12. CONCLUSION

Relational databases give important mechanisms for managing structured data through tables and connections. still, they warrant direct support for ordering multi- valued attributes within a single tuple. In practical scripts, analogous attributes constantly bear a specific sequence, and achieving this using traditional SQL styles involves complex queries, multiple sorting operations, and increased computational exodus.

This disquisition introduced the Dynaseq Tuple Ordering Algorithm(DTOA), an effective approach that roundly assigns sequence pointers to multi- valued attributes predicated on a predefined pattern. By incorporating ordering directly into the data processing stage, DTOA ensures that results reacquired through SQL queries are formerly in the demanded order, barring the need for repeated sorting. The algorithm improves query effectiveness by reducing reliance on complex SQL constructs and minimizing operation- position processing. This leads to hastily execution, better performance, and bettered scalability when handling large datasets. also, DTOA enhances query readability and simplifies the operation of ordered data.

Future disquisition can explore integrating DTOA with database query optimizers and extending its operation to distributed database systems for bettered scalability and performance.

13. ACKNOWLEDGEMENT

Apart from the efforts of me, the success of any work or project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this research paper.

I express deep sense of gratitude to almighty God for giving me strength for the successful completion of the research paper.

I express my heartfelt gratitude to my parents for constant encouragement while carrying out this research paper.

I express my deep sense of gratitude to **The Principal** who has been continuously motivating and extending their helping hand to us.

My sincere thanks to **Mr. Praveen Kumar Murigeppa Jigajinni**, Master In-charge, A guide, Mentor and great motivator, who critically reviewed my paper and helped in solving each and every problem, occurred during implementation of this research paper.

14. REFERENCES

1. *E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM, 1970.*
2. *R. Elmasri and S. Navathe, Fundamentals of Database Systems, Pearson.*
3. *A. Silberschatz, H. Korth, and S. Sudarshan, Database System Concepts, McGraw Hill.*
4. *C. J. Date, An Introduction to Database Systems.*