Combinational Approach for Object Clipping Using GLIP and Protection Against Sql Injection Attacks

B.SRINIVASA RAO [#]1, MANOJ KUMAR YADAV[#]2"V. KRISHNA PRATAP[#]3,P.NARASIMHA RAO[*]4

[#]1Department of Computer Science & Information Technology, Sri Sarathi Institute of Engineering and Technology, Nuzvid, Jntuk, A.P, INDIA

#2Department of Computer Science & Engineering, Drona Charya Group Of Institutions,Greator Noida, M.T.U,U.P,

*3Department of Computer Science & Engineering, venkateswra Institute of science and Information Technology, Tadepalligudem, Jntuk, A.P,

*4Department of Computer Science & Engineering, Sri Sarathi Institute of Engineering and Technology, Nuzvid, Jntuk, A.P,

Abstract—

Multidimensional databases are being used in a wide range of applications. To meet this fast-growing demand,R+ trees were used that exhibit outstanding search performance. In order to support efficient concurrent access in multiuser environments, concurrency control mechanisms for multidimensional indexing have been proposed. However, these mechanisms cannot be directly applied to the R+-tree because an object in the R+-tree may be indexed in multiple leaves. This paper proposes a concurrency control protocol for R-tree variants(ZR+ trees) with object clipping, namely, Granular Locking for clIPping indexing (GLIP). GLIP is the first concurrency control approach specifically designed for the R+-tree and its variants, and it supports efficient concurrent operations with serializable isolation, consistency, and deadlock-free. A HTTP Analyzer tool is used in order to check whether the HTTP request packets sent by the client web application to the web server were according to the RFC specifications or not. SQL injection is a technique that can give attackers unrestricted access to the databases that underlie Web applications and has become increasingly frequent and serious. This paper presents a new highly automated approach for protecting Web applications against Data base vulnerabilities.

I. INTRODUCTION

In recent years, multidimensional databases have begun to be used for a wide range of applications, including geographical information systems (GIS), computer-aided design (CAD), and computer-aided medical diagnosis applications. As a result of this fast-growing demand for these emerging applications, the development of efficient access methods for multidimensional data has become a crucial aspect of database research. Many indexing structures have been proposed to support fast access to multidimensional data in relational databases. Among these indexing structures, the R+-trees has attracted significant attention as the tree structure is regarded as one of the most prominent indexing structures for relational databases. The R+-tree exhibits better search performance, making it suitable for applications where search is the predominant operation. However, the R+-tree is not suitable for use with current concurrency control methods because a single object in the R+-tree may be indexed in different leaf nodes. Special considerations are needed to support concurrent queries on R+-trees, while as far as we know, there is no concurrency control approach that specifically supports R+-trees. Furthermore, there are some limitations in the design of the R+-tree, such as its failure to insert and split nodes in some complex overlap or intersection cases. This paper proposes a concurrency control protocol for R-trees with object clipping, Granular Locking for clIPping indexing (GLIP). We also introduce the Zerooverlap R+-tree (ZR+-tree), which resolves the limitations of the original R+-tree by eliminating the overlaps of leaf nodes. GLIP, together with the ZR+-tree, constitutes an efficient and sound concurrent access model for multidimensional databases.

We are using a HTTP protocol analyzer tool which is responsible for analyzing the Data base access request packets sent by the client web application, whether they were according to the RFC 2616 specifications or not. If not, they were treated as abnormal requests and were not transferred to the Web server. Only well formed HTTP packets that were according to the specifications were sent to the server in order to be processed.

Before granting access to the underlying database resource, the request queries were throughly inspected, inorder to ensure that they were attack free and may not leave the database in an inconsistent state. The access to the database for the users is provided by the GLIP system which provides serializable isolation, consistency, and deadlock-free operations.

II. HTTP PROTOCOL ANALYZER TOOL

The proposed is trained before being placed in the operation.

Training phase:

During training phase, the tool is trained with a valid set of HTTPmessages. These HTTP messages are those that will receive a positive response from the Web Server and were according to HTTP RFC 2616 specifications. They were captured by the tool and stored in a database. The Read header function obtains the header of the HTTP message. The Generate digest function obtains digest by comparing the header with 116 predefined set of HTTP headers. If there is a match, the corresponding bit is set to 1 in digest. Otherwise, the bit is set to 0. Thus a 116 bit message digest is obtained for each valid HTTP message and is stored in the database. Once it is trained, it ready for its operation.

During its operation, all the HTTP messages directed to the Server were captured by the tool were stored in the database. It removes the header of the HTTP message and then generates digest using the mechanism explained earlier. It then compares the digest generated for each message with the digest of the valid case messages that were already stored during the training phase. If no match is found it is treated as an abnormal packet. All such abnormal packet requests were blocked and were not sent to the Server. Only valid case HTTP messages/requests were forwarded to the Server for further processing.

III. SQL INJECTION DETECTION SYSTEM

Web applications interface with databases that contain information such as customer names, preferences, credit card numbers, purchase orders, and so on. Web applications build SQL queries to access these databases based, in part, on user-provided input. The intent is that Web applications will limit the kinds of queries that can be generated to a safe subset of all possible queries, regardless of what input users provide. However, inadequate input validation can enable attackers to gain complete access to such databases. One way in which this happens is that attackers can submit input strings that contain specially encoded database commands. When the Web application builds a query by using these strings and submits the query to its underlying database, the attacker's embedded commands are executed by the database and the attack succeeds. The results of these attacks are often disastrous and can range from leaking of sensitive data to the destruction of database contents.

The detection system perfomes 2 operations:

1. Character-level tainting
2. Syntax aware evaluation

Character-level tainting: We track taint information at the character level rather than at the string level. We do this because, for building SQL queries, strings are constantly broken into substrings, manipulated, and combined. By associating taint information to single characters, our approach can precisely model the effect of these string operations.

Syntax aware evalution: The key feature of syntax aware evaluation is that it considers the context in which trusted and untrusted data is used to make sure that all partsof a query other than string or numeric literals (for example, SQL keywords and operators) consist only of trusted characters. As long as untrusted data is confined to literals, we are guaranteed that no Injection attack can be performed. Conversely, if this property is not satisfied (for example, if a SQL operator contains characters that are not marked as trusted),we can assume that the operator has been injected by an attacker and identify the query as an attack.

Our Detection system performs syntax-aware evaluation of a query string immediately before the string is sent to the database to be executed. To evaluate the query string, the technique first breaks the string into a sequence of tokens that correspond to SQL keywords, operators, and literals. The technique then iterates through the tokens and checks whether tokens (that is, substrings) other than literals contain only trusted data. If all such tokens pass this check, the query is considered safe and is allowed to execute. If an attack is detected, it is not executed.

IV. ZR+ TREES

The R+-tree uses a clipping approach to avoid overlap between regions at the same level. As a result of this policy, a point query in the R+-tree corresponds to a single path tree traversal from the root to a single leaf. For search windows that are completely covered by the MBR of a leaf node, the R+-tree guarantees that only a single search path will be traversed. Examples of the R-tree and R+-tree are given in Fig. 1, where A and B are leaf nodes, and C, D, E, and F are MBRs of objects. Because objects D and E overlap with each other in the dataspace, leaf nodes A and B have to overlap in the R-tree in order to contain the objects. In contrast, in the R+-tree, leaf nodes do not have to cover an entire object, so object D can be included in both leaf nodes A and B. As a result, the R+-tree clearly has a better search performance compared to the R-tree.

The essential idea behind the ZR+-tree is to logically clip the data objects to fit them into the exclusive leaf nodes. There are two fundamental differences between the clipping techniques applied in the ZR+-tree and the R+-tree:

1. From the definition of the ZR+-tree, object clipping in the ZR+-tree must differentiate the MBRs of the segmented objects in leaf nodes ,while the clipping in the R+-tree retains the original MBRs.
2. In the ZR+-tree, each entry in a leaf node is a list of segmented objects that share the same MBR, while each leaf node entry in the R+-tree contains exactly one object.

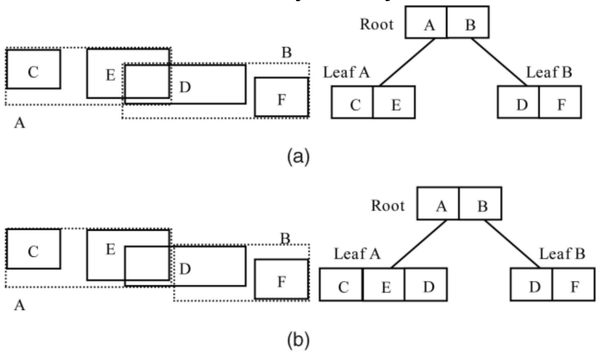


Fig. 1. Examples of R-tree and R+-tree. (a) An R-tree example. (b) An R+-tree example.

For example, in Fig. 2, the first entry in the leaf node A contains segmented objects O, P1, Q1, and R1, with the same MBR, and the second entry in the leaf node A contains segmented objects P2, Q2, and R2, with the same MBR. These segmented objects with the same MBR are combined into a single entry. These two features in the ZR+-tree can help to resolve the unable-to-split problem, as well as to reduce the number of leaf nodes after clipping objects. As the proposed object clipping ensures zero overlap in the entire search tree, the structure and the operations become more orthogonal. Furthermore, this zero-overlap design avoids the limitations associated with duplicating the links between objects. An example of the ZR+-tree that can be compared to the R-tree and the R+-tree in Fig. 1 is given in Fig. 3, where the object D is clipped into D1 and D2 to achieve zero overlap and avoid the construction limitations of the R+-tree.The definition of the ZR+-tree is given in the form of a revised version of the earlier definition of the R+-tree by modifying property 1 and 2 as follows:

1. A leaf node has one or more entries of the form (objectlist, RECT) where objectlist gives the identifiers for each object that completely encloses or covers RECT. Note that a single bounding rectangle with multiple object ids is still counted as a single entry, even though it requires extra space in the node. An alternative is to use a pointer as objectlist to the entry in a table that stores the corresponding object ids.
2. An internal node has one or more entries of the form (p, RECT) where p points to a ZR+-tree leaf or internal node R such that RECT is the MBR of all (p_i, $RECT_i$) in R. Thus, the definition of the ZR+-tree is more orthogonal as a result of eliminating the difference in rules for the MBRs of leaf nodes and internal nodes. However, the MBR of an object may be fragmented, such that the union of all the fragments equals the MBR of the object, and each of the fragments may be inserted into the same or different leaf nodes.

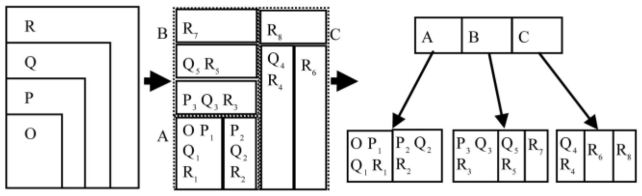


Fig.2. Object clipping in ZR+-tree

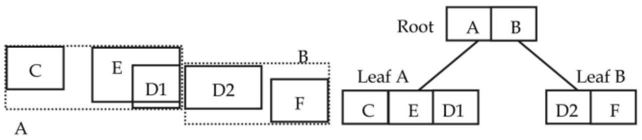


Fig. 3. An example of ZR+-tree for the data in Fig. 1.

V. CONCURRENCY CONTROL USING GLIP

The presence of a standard lock manager is presumed to support conditional and unconditional lock requests, as well as instant, manual, and commit lock durations in GLIP. A conditional lock request means that the requester will not wait if the lock cannot be granted immediately; an unconditional lock request means that the requester is willing to wait until the lock becomes grantable. Instant duration locks merely test whether a lock is grantable, and no lock is actually placed. Manual duration locks can be explicitly released before the transaction is completed. If they are not released explicitly, they are automatically released at the time of commit or rollback. Commit duration locks are automatically released when the transaction ends.

Conventionally, five types of locks, namely, S (shared locks), X (exclusive locks), IX (Intention to set X locks), IS (Intention to set S locks), and SIX (Union of S and IX locks) are used. In the proposed protocol, only S and X locks are used to support concurrent operations with relatively simple maintenance processes. The lock manager in GLIP is presumed to support the acquisition of multiple locks as an atomic operation. If this is not the case, such a procedure can be conveniently implemented by acquiring the first lock in a list unconditionally and all subsequent locks conditionally, with the procedure releasing all the acquired locks and restarting if any of the conditional locks cannot be acquired. Furthermore, a transaction can place any number of locks on the same granule as long as they are compatible. The lock manager will place separate locks for each granule, and each lock will be distinct even if the lock modes are the same. When releasing manual duration locks, both the lock granule and lock mode must be specified.

Each leaf node in the ZR+-tree is defined as a lockable granule(ext(T)). In order to reduce the overhead associated with lock maintenance, objects are not individually lockable. The clip array introduced as an auxiliary structure to store the object clipping information does not need to be locked because the locking strategies on leaf nodes ensure the serializability of access for the same object, and updating one object will not affect the other objects.

Selection Operation:

The select operation returns all object ids given a search window W. It is necessary to place locks on all granules that overlap with the search window in order to prevent writers from inserting into or deleting from these granules until the transaction is completed.

Selection starts by checking whether the search window overlaps with ext(T). If so, a shared lock is placed on ext(T), thus preventing a writer from inserting data into this space. A breadth-first traversal is then performed starting from the root node and traversing each node whose MBR overlaps with the search window. For each internal node that overlaps with W, an S lock is placed on its external area. This lock is released when all of its child nodes and its external granular have been inspected and locked if necessary. For each internal node, if the MBRs of its children do not fully cover the search window W, an S lock will be kept on the external granule for the node in order to prevent writers from modifying this region. This ensures consistency within the tree, as it prevents writers from modifying the internal node until all the child nodes have been properly inspected and protected.

As discussed earlier, in order to reduce the number of locks that must be placed and released, we neither perform object-level locking, nor lock the corresponding objects in the clip array for the select operation. Instead, shared locks are placed on the leaf nodes that overlap withW. Since the same object id may recur in the same leaf node or across different leaf nodes, a set of object ids is maintained to avoid returning the same object id more than once. This is consistent with the expected result from a select statement. Finally, all the locks on the granules that overlap with W are released once the search is complete.

Example:

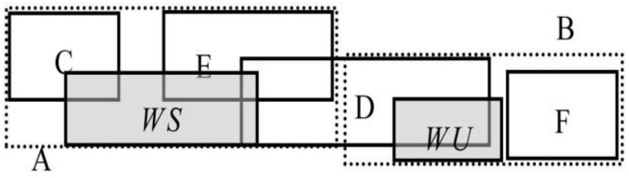
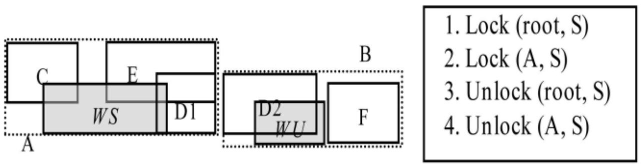


Fig.4. Example operations

In Fig. 4, assuming A and B are leaf nodes, the search window WS requires shared locks to be placed on the lockable granules A, whereas the update window WU requires exclusive locks to be placed on B.

Fig. 5 illustrates the lock management for the window query in Fig. 4. For a search window WS that overlaps with C, E, and D, initially, an S lock will be placed on the root. An S lock is then placed on the leaf node A and the lock on the root is released. This prevents any other transactions from modifying the root (by placing an X lock on it) until all its children have been inspected. After the lock on the root has been released, the entry for node B in the root can be modified as long as the modification does not result in overlap with A. Thus, manual duration S locks are used tomaintain consistency while at the same time maximizing the degree of concurrency



1. Lock (root, S)
2. Lock (A, S)
3. Unlock (root, S)
4. Unlock (A, S)

Fig.5. Locking sequence for WS in Fig. 4.

VI. PROPOSED SYSTEM SCHEME OF OPERATION

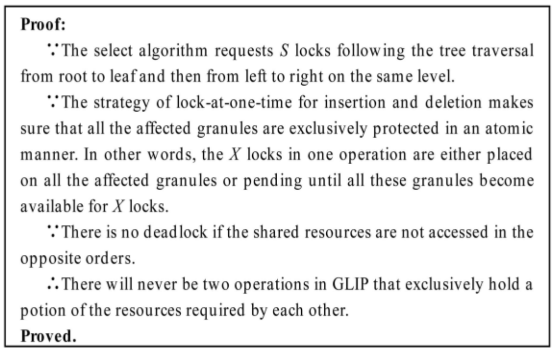
The architecture of the proposed scheme is as shown in Fig. 6. The operation of the system is explained in terms of the following steps:

1. The web requests for database access from multiple users directed to the web server were taken by the HTTP Protocol Analyzer tool.
2. The tool then checks whether the HTTP request packets sent by the clients were according to the RFC specfications or not. If not they were simply discarded. If valid, they were forwarded to the web server.
3. The data base access query sent by the client web application is carefully examined by the Data base Vulnerability Detection System.
4. The DB Vulnerability Dection System extracts each individual character, groups them into tokens and compares them with the metasymbol library(i.e valid SQL keywords, literals etc). This technique then iterates through the tokens and checks whether tokens

(that is, substrings) other than literals contain only trusted data. If all such tokens pass this check, the query is considered safe and is allowed to execute.

5. The Database server consisting of multidimensional databases uses Granular Locking for clIpping Protocol. The locking mechanism provided by the GLIP ensures consistency. It supports efficient concurrent operations with serializable isolation, consistency, andndeadlock-free

Deadlock free in GLIP:



Proof:

∵The select algorithm requests S locks following the tree traversal from root to leaf and then from left to right on the same level.

∵The strategy of lock-at-one-time for insertion and deletion makes sure that all the affected granules are exclusively protected in an atomic manner. In other words, the X locks in one operation are either placed on all the affected granules or pending until all these granules become available for X locks.

∵There is no deadlock if the shared resources are not accessed in the opposite orders.

∴There will never be two operations in GLIP that exclusively hold a potion of the resources required by each other.

Proved.

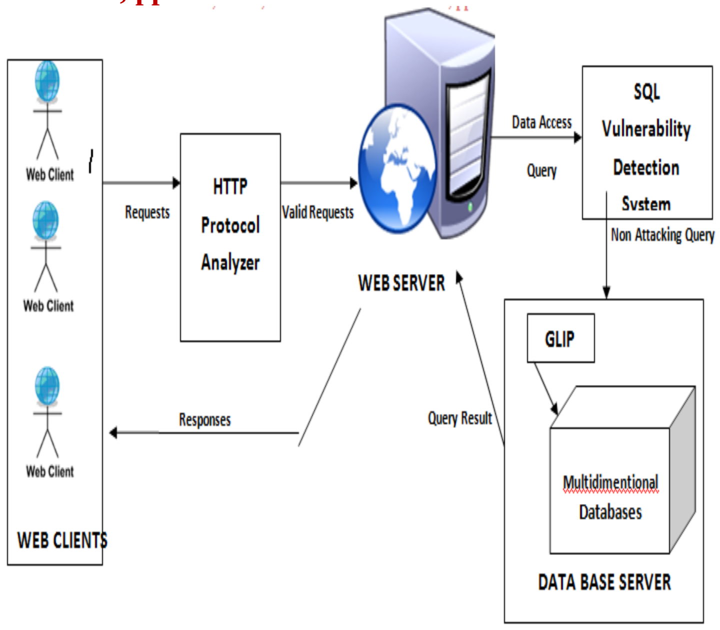


Fig .6. Architecture of the proposed system

VII. ANALYSIS

Based on the proposed GLIP protocol, ZR+-tree operations meet the requirements of serializable isolation, consistency, and no additional deadlocks. Specifically, serializable isolation is guaranteed by the strategy of requesting Shared locks on reading and Exclusive locks at the same time on updating. These locks are granted on the affected granules before the actual actions and provide protection until the process is complete. Therefore, the intermediate status of one operation cannot be exposed to any other operations. The consistency requirement is ensured by implementing version checking and restarting the insertion or deletion when the version does not match. This version checking prevents the update operations from modifying a version of the ZR+-tree that differs from the one investigated.

VIII. CONCLUSION

This paper proposes a new concurrency control protocol, GLIP, with an improved spatial indexing approach, the ZR+-tree. GLIP is the first concurrency control mechanism designed specifically for the R+-tree and its variants. It assures serializable isolation, consistency, and deadlock free for indexing trees with object clipping. The ZR+-tree segments the objects to ensure every fragment is fully covered by a leaf node. This clipping-object design provides better indexing structures. Several structural limitations of the R+-tree are overcome in the ZR+-tree by the use of a nonoverlap clipping. Furthermore, the usage of Database Vulnerability Detection System ensures the protection of underlying databases from vulnerable attacks that may otherwise leave the database in an inconsistent state, leaking of sensitive data and sometimes to the destruction of database contents.

Extending GLIP and the ZR+-tree to perform spatial joins, KNN-queries, and range aggregation offer further attractive possibilities.

REFERENCES

[1]. N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger, “The R_-Tree: An Efficient and Robust Access Method for Points and Rectangles,” Proc. ACM SIGMOD ’90, pp. 322-331, 1990.

[2]. A. Biliris, “Operation Specific Locking in B-trees,” Proc. Sixth Int’l Conf. Principles of Database Systems (PODS ’87), pp. 159-169, 1987.

[3]. J.K. Chen, Y.F. Huang, and Y.H. Chin, “A Study of Concurrent Operations on R-Trees,” Information Sciences, vol. 98, nos. 1-4, pp. 263-300, May 1997.

[4]. V. Gaede and O. Gunther, “Multidimensional Access Methods,” ACM Computing Surveys, vol. 30, no. 2, pp. 170-231, June 1998.

[5]. C. Anley, “Advanced SQL Injection In SQL Server Applications,” white paper, Next Generation Security Software, 2002.

[6]. S.W. Boyd and A.D. Keromytis, “SQLrand: Preventing SQL Injection Attacks,” Proc.

Second Int'l Conf. Applied Cryptography and Network Security, pp. 292-302, June 2004.

B.Srinivasa Rao received his B.Tech degree in Computer Science and Information Technology from Dr.Paul Raj Engineering College, Jawaharlal Technological University, A.P India.The M.E. degree in Computer science and engineering from Anna University, Tamilnadu, and India. He was worked as an assistant professor in the department of computer science and engineering in various engineering colleges. At present he is working as an assistant professor in the department of computer science and information technology in sri sarathi institute of engineering and technology ,Jawaharlal technological university kakinada,A.P India. His research interests include information security, networking techniques, formal languages and automata theory, cloud computing.

MANOJ KUMAR YADAV received his M.C.A degree from BSA college of Engineering &Technology, Uthtar Pradesh Technical University, U.P, India. The M.Tech. Degree in Computer science and Engineering from Jawaharlal technological university, Hyderabad, A.P India. He was worked as an assistant professor in the department of computer science and engineering in sri sarathi institute of engineering and technology ,Jawaharlal technological university, kakinada,A.P India . At present he is working as an assistant professor in the department of computer science and Engineering, Dronacharya Group of Institution,U.P, India. His research interests include information security, networking techniques, formal languages and automata theory.

Mr. V. Krishna Pratap received his M.Tech , in CSE from JNTU KAKINADA University , AP and India in the year of 2011. He studied B.Tech in the Specialization of INFORMATION TECHNOLOGY from JNTU KAKINADA University, AP and India. He was worked as Asst. Professor in the Computer Science and Engineering in Sri Sarathi Institute of Engineering and Technology; Nuzvid, AP and India At present he is working as a asst.prof in the dept of CSE in venkateswra institute of science and information technology ,Tadepalligudem,A.P,India. His research interests include information security, networking techniques, formal languages and automata theory, cloud computing.