Query Processing in Multilevel Secure Distributed Databases

Pooja Sapra
RESEARCH SCHOLAR, MRIU
mrs.sapra@gmail.com

Suresh Kumar
FET, MRIU
enthuvs@gmail.com

Rk Rathy
FET, MRIU
rkrathy.fet@mriu.edu.in

***Abstract*— Multilevel secure database systems are the systems, which each data item is assigned a security level or sensitivity level. The security classifications are assigned from complete relation to the individual data elements. To assign sensitivity levels, the capability of security constraints or classification rules is utilized. Security constraints assign the security levels on the basis of content, context, and time of data items. In this paper, we propose an integrated architecture for distributed query operations and coding of multi-levels for the purpose of providing extra security when it is stored on various sites.**

Keywords—component; formatting; style; styling; insert (key words)

I. INTRODUCTION

In multilevel secure database management systems, some part of the database is accessible at different levels of security, depending upon the clearance level of user. To assign the classifications, security constraints are used. They assign security levels on the basis of content, context, and time. In this paper, we propose an integrated architecture for distributed query operations and coding of multi-levels for the purpose of providing extra security. There are two basic security issues for providing the security to distributed database:

1. Secure data transmission
2. Secure data storage and access

The transmission of information from one site to another is protected through SSL and TLS. However, no sufficient support is present in storing and processing them in secure way, when the data is stored at the backend.
So integration of cryptographic support into MLS/DBMS is considered and for this purpose, the concept of coding and decoding of levels in secure dictionary has been also introduced. By integrating cryptography to DBMS, some of the major problems of MLS/DBMS may be solved.

The paper is organized as follows. Section 2 we discuss the literature review. Section 3 discusses the constraint processing during query operations. In section 4, proposed integrated architecture for MLS/DDBMS is given and the function of each module is given. Section 4 gives operations of MLS/DDBMS. Section 5 illustrates the operations with example. Section 6 gives the performance trade-off for adding cryptography and security, section 7 concludes the paper.

II. LITERATURE REVIEW

Security constraints are basically required to enforce the classification policy. These constraints can be considered as the integrity constraints. In non-multilevel relational database systems many techniques have been developed for handling integrity constraints by logic programmers[10]. The same techniques may be used for processing security constraints during query updates , query processing and database design. Fig. 1 shows the integrated architecture[1], which is influenced by the design of lock data views[15] for handling constraints during query, update and database design operations.

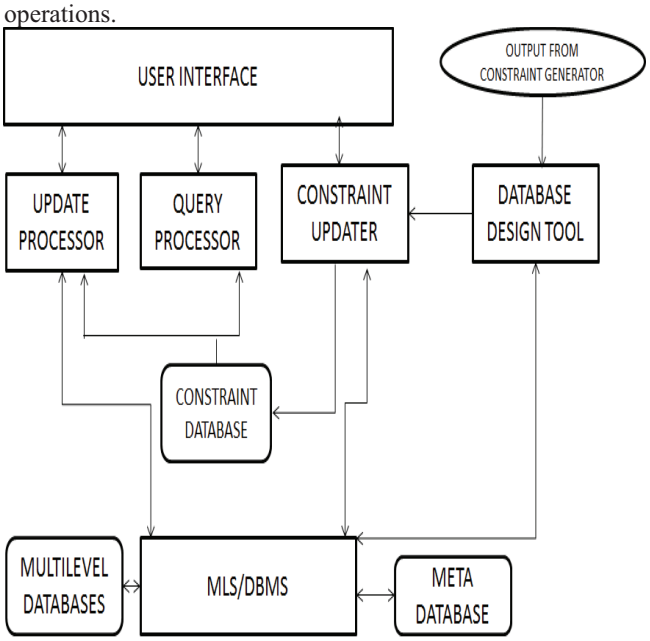

Fig. 1. Integrated Architecture for Centralized Databases [1]

This architecture can be divided into multilevel relational database management system and deductive manager.
The deductive manager is also known as the query/update constraint processor.

In this architecture, constraint generator produces the constraints and schema, which are fu rther processed by the database design tool. Then the constraint updater modifies the constraint database by considering t he modified constraints. Then the schema is stored in the meta-database. Constraint database is used by the query processor and update constraint processors. The query constraint processor checks all constraints and ensures that users do not get unauthorized data.

III. SECURITY CONSTRAINTS

Security constraints are the rules used for assigning security levels to the data. The various types of security constraints include [1]:

1) Simple constraints: these are the constraints that are required to classify a database, relation, or an attribute.
2) Content based constraints: these constraints saggregates some part of t he database depending on the value of data.
3) Event Based Constraints: th ey are required to classify part of the database depending on some real-world event.
4) Association based constraints: Constraints that are required to classify associations between tuples, attributes, elements, etc.
5) Release Based Constraints: they help to extract the database depending on the previously released information. Release-based constraints may be general release constraint which classifies an entire attribute or the individual release constraint which classifies a Value of an attribute.
6) Aggregate Constraints: these are the constraints that classify the collections of data.
7) Level Based Constraints: these constraints classify database on the basis of the security level of some data.
8) Logical Constraints: Constraints which defines the inferences by using implications.
9) Meta constraints: Constraint which are required to define the constraints about constraints and metadata.

IV. INTEGRATED ARCHITECTURE FOR MLS/DDBMS

In an MLS/DDBMS, users share a MLS database that is distributed at various sites without compromising the security. The integrated architecture of MLS/DDBMS consist various homogeneous nodes that are interconnected by a trusted network. Each node has the multilevel data and an MLS/DBMS to manage the local multilevel database as shown in Fig 2.

For distributed processing, each node has a component called the Secure Distributed Processor (SDP), which further has following components for its operations:

Distributed Query update Processor (DQUP): Th e DQUP consists of Distributed Query Processor (DQP), which is responsible for distributed query processing and Distributed Update Processor (DUP), which handles the distributed updates. The DTM is responsible for distributed transaction management.

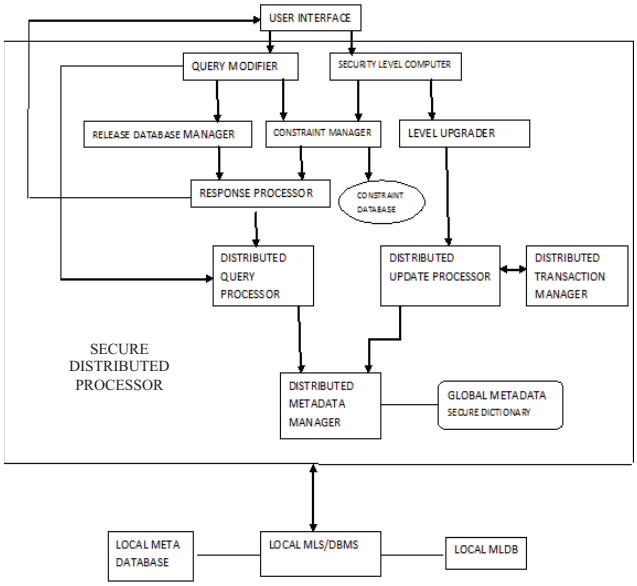

Fig. 2. Integrated Architecture for MLS/DDBMS

Distributed Metadata Manager (DMM): It manages the global metadata, which includes the information on schemas, how they are distributed and fragmented among various sites, their locations and the enforced constraints.

Distributed Constraint Processor (DCP): it handles security constraints during query and update operation with the help of Distributed Query Constraint Processor (DQCP) and Distributed Update Constraint Processor(DUCP), to process constraints during the query operation and update operation respectively.

Secure Dictionary (SD): it prevents the unauthorized access to important information like clearance level and classification level of user and t ransactions. It is also termed as security *catalogue that* is replicated at each site. For a bet ter security environment and its full usage, it must not be updat ed by anyone and its access must be controlled by strict policies.

V. OPERATIONS OF MLS/DDBMS

The constraints are processed by either the constraint processor or update processor.

For processing the constraints during query processing the following modules come into action:

The User Interface Manager (UIM) accepts the query, checks it syntactically, parses it and submits it to que ry modifier/security level computer. It also releases its response to the user.

Then Query Modifier accepts the parsed query from DQP.
Constraint manager manipulates the constraints and communicates with both QM/RP. It also interacts with constraint manager of other sites to access constraint.
Response processor takes the response from DQP, then sends a message to C M and RDM to access relevant released constraints and information that has been released. Then the response is sent to UIM.
For processing the constraints during update processing the following modules come into action:
UIM, accepts the update request, parse that and gives that to security level computer (SLC).
Security Level Computer takes the parsed request, communicates with CM and computes the security level of update request. Then it sends the information to level upgrade.
Level Upgrade (LU) creates the processes for each (old-data, old-level, new-data, new-level) tuple and gives the corresponding data to these processes. The DUP deletes the old data and inserts new data into database.

VI. ILLUSTRATIVE EXAMPLE

Consider a database of a bank that is distributed at two sites. It consists of two relations ACCOUNT and ASSESTS at each site (Table 1). We assume that all the tuples are at unclassified level.
The constraints are as follows:

1. ACCOUNT. LOCATION = Y
   =>LEVEL(ACCOUNT.LOCATION)=TS
2. ASSETS.ASSETSVALUES>20,000
   =>LEVEL(ASSETS.ASSETSVALUE)=TS
3. ASSETS.ASSETSVALUES<20,000
   =>LEVEL(ASSETS.ASSETSVALUE)=U

TABLE 1: MULTILEVEL SECURE DISTRIBUTED DATABASE

SITE 1:

ACCOUNT:

ACC#	NAME	BALANCE	LOCATION
101	A	1000	X
102	B	2000	X
104	D	4000	Y
106	F	6000	Y
108	H	8000	X
110	J	10,000	Y

ASSETS:

LOCATION	ASSETS VALUES
X	11,000
Y	20,000

SITE 2:

ACCOUNT:

ACC#	NAME	BALANCE	LOCATION
103	C	3000	X
105	E	5000	Y
107	G	7000	X
109	I	9000	X
111	K	11000	Y
112	L	12,000	Y

ASSETS:

LOCATION	ASSETS VALUES
X	19,000
Y	28,000

Constraints are classified at level unclassified and replicated at both the sites.

Now, consider the following query that has been issued at site 1 by user with classification level unclassified:
"SELECT LOCATION, ASSETSVALUE FROM ASSETS"

Execution of query will be as follows:

STEP NO.1:
UIM parses the query and pass it to DQCP.

STEP NO.2:
QM accepts the constraints from CM and modifies the query.

STEP NO.3:
DQP at site 1, encrypts the levels by using security dictionary which is available at each site.

STEP NO.4:
It sends the subqueries to MLS/DBMS and DQP at site 2.

STEP NO.5:
DQP at site 2 does the decoding of the levels with the help of security dictionary available at this site.

STEP NO.6:
Query is evaluated at both the sites.

STEP NO.7:
DQP at site 2 again encode the level and send the result to the query issuing site.

STEP NO.8:
Decoding of levels at site 1 is done.

STEP NO.9:
Result is merged at site 1 by DQP.

Let the total no. of sites be n, and then the operations required for each step are as follows:

1. Parse tree generation: 1
2. Modification of the query:1
3. Coding of levels:1
4. Generation and sending of sub-queries to different DQP at diff sites : n
5. Decoding of levels at various sites: n
6. Evaluation of queries at various sites: n
7. Encoding of level of result query: n
8. Decoding of levels at issuing site: 1
9. Merging of results: 1

So, total no. of operations required are 4n+5.

Interpretation:
As shown in Table 1, with an extra overhead of 2n+2 operations in coding and decoding of various levels the database c an be m ore secure.

TABLE II: NUMBER OF SITES VS. OPERATIONS AND OVERHEADS REQUIRED WITH AND WITH OUT SECURE DICTIONARY

number of sites	number of operations required (with secure dictionary)	number of operations required (without secure dictionary)	OVERHEADS
2	13	7	6
3	17	9	8
5	25	13	12
10	45	23	22
20	85	43	42
30	125	63	62
50	205	103	102
100	405	203	202

This can also be shown by the curve shown in Fig.3

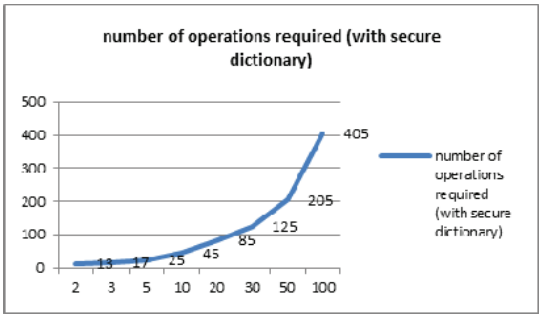

Fig.3. Number of Operations required with Secure Dictionary

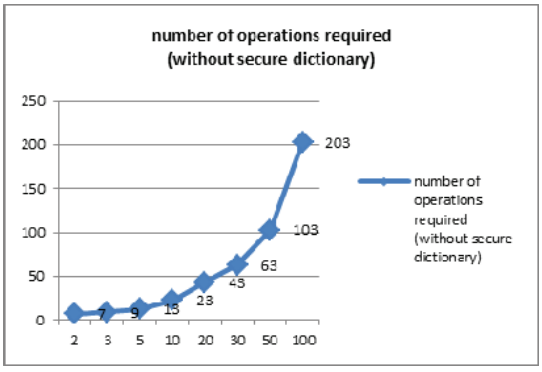

Fig.4. Number of Operations required without Secure Dictionary

Further, if we increase the number of security levels then the time taken for execution of queries also increases as shown in figure. The experiment was done with the set of 200 tuples in a database with the schema as follows:
Locations (LOCATION_ID, STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID)

The set-up time was very large in case of secure databases and it increased with the increase in sec urity levels as shown in Table 3 to Table 5.

TABLE III: SET UP TIME FOR INSECURE DATABASE

CREATION OF A DATABASE:	00.00.00.32
INSERTION OF ROWS:	00.00.00.01 per row

TABLE IV: SET UP TIME FOR SECURE DATABASE(2-LEVELS FOR SECURITY):

CREATION OF A DATABASE:	00.00.00.32
INSERTION OF ROWS	00.00.00.01 per row
CREATION OF USERS AND ROLES:	00.00.00.48
CREATION OF A POLICY	00.00.03.18
GRANTING ROLES:	00.00.00.15
CREATION OF LEVELS:	00.00.00.24
CREATION OF LABELS:	00.00.00.91
SETTING USER LABELS:	00.00.00.40
SET USER PRIVILIGES:	00.00.00.04
APPLY POLICY:	00.00.01.23
ADD LABEL COLUMN:	00.00.00.50
CREATION OF INDEX:	00.00.00.30
REVOKING ACCESS:	00.00.00.05
CLEANING UP:	00.00.16.29

TABLE V: SET UP TIME FOR SECURE DATABASE(3-LEVELS FOR SECURITY)

CREATION OF A DATABASE:	00.00.00.32
INSERTION OF ROWS:	00.00.00.01 per row
CREATION OF USERS AND ROLES:	00.00.00.52
CREATION OF A POLICY	00.00.04.94
GRANTING ROLES:	00.00.00.25
CREATION OF LEVELS:	00.00.00.36
CREATION OF LABELS:	00.00.00.97
SETTING USER LABELS:	00.00.00.40
SET USER PRIVILIGES:	00.00.00.04
APPLY POLICY:	00.00.01.93
ADD LABEL COLUMN:	00.00.00.63
CREATION OF INDEX:	00.00.00.30
REVOKING ACCESS:	00.00.00.06
CLEANING UP:	00.00.18.13

Firstly, we performed the set of all queries on insecure database and then the same queries were performed to the multi-level secure database with two levels for security and then 3 - levels for security.

TABLE VI: AVERAGE TIME TAKEN FOR DIFFERENT OPERATIONS

QUERIES	AVERAGE TIME TAKEN		
	EXECUTION TIME(INSECURE DATABASE)	EXECUTION TIME(SECURE DATABASE(2-LEVELS))	EXECUTION TIME(SECURE DATABASE(3-LEVELS))
SELECT QUERIES	0.22	0.23	0.37
DELETE QUERIES	0.01	0.07	0.09
ALTER QUERIES	0.27	1.33	1.75
UPDATE QUERIES	0.04	0.22	0.36
DROP SCHEMA	0.53	0.73	2.71

Table 6 shows the average time taken for different set of queries and it is also shown diagrammatically by Fig.5.

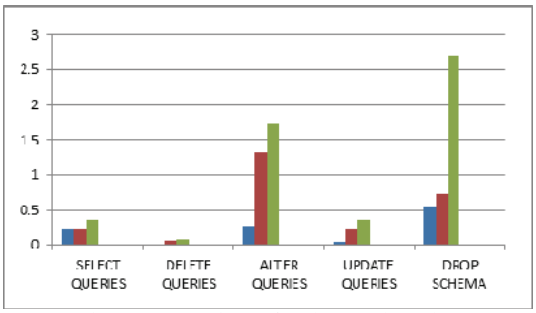

Fig.5. Average Time Taken for Set of Queries

VII. CONCLUSIONS

In MLS/DDBMS, the storing of classification labels in the records is not enough to prevent top secret data, from reaching an unclassified user; the database system may simply change the label or merge top secret data into a record marked unclassified. To protect changes against the classification labels, the concept of coding and decoding with the help of secure dictionary is proposed**.** An MLS/DDBMS, users share a MLS database that is distributed at various sites without compromising the security.

VIII. FUTURE SCOPE

The secure dictionary may also be changed by intruders, so as to avoid the changes to this dictionary it may also be coded or encrypted by using different encryption techniques.

REFERENCES

[1] B. thuraisingham, W.ford, "Security Constraint Processing in a Multilevel Secure Distributed Database Management System",IEEE Transactions on know ledge and data engineering, vol.7, no.2, April, 1995.

[2] D. E. Denning, Cryptography and Data Security, Addison-wesley, Reading mass., 1982.

[3] J.He, M.Wang, "Cryptography and Relational Database Management System", Database Engineering and Applications, pp 273-284.

[4] D. Bell and L. La Padula, Secure Computer Systems: Unified Exposition and Multics Interpretation, Technical Report NTIS AD-A023588. Bedford, Mass.: MITRE Corporation, July 1975.

[5] U. Chakravarthy, "Foundations of Semantic Query Optimization", ed. by J. Minker, Foundations of Deductive Databases and Logic Programming.San Francisco: Morgan Kaufmann, 1988.

[6] M. Collins, "Design and Implementation of Secure Update Processor", Technical Report MTR 10977. Bedford, Mass.: MITRE Corporation,Oct. 1990.

[7] D. E. Denning, S.G. AH, M. Morgenstem, P.G. Neumann, R.R. Schell and M. Heck, "Views as a Mechanism for Classification in Multilevel Secure Database Management Systems", Proc. IEEE Symp. Security and Privacy, Oakland, Calif., 1986.

[8] P. Dwyer, G. Jelatis and B. Thuraisingham, "Multilevel Security in Database Management Systems", Computers and Security, vol. 6, no. 3, pp, 252-260, June 1987.

[9] W. Ford, J. O'Keeffe, and B. Thuraisingham, "Database Inference Controller: An Overview", Technical Report MTR 10963, vol. 1, MITRE Corporation, Bedford, Mass., Aug. 1990.

[10] H. Gallaire and J. Minker, "Logic and Databases", New York: Plenum Press, 1978.

[11] T. Hinke, "Inference Aggregation Detection In Database Management Systems", Proc. IEEE Symp. on Security and Privacy, Oakland, Calif., Apr. 1988

[12] Honeywell Inc., "Security Policy for Lock Data Views", Interim Report for RADC, Mar. 1987.

[13] T. Keefe, B. Thuraisingham and W. Tsai, "Secure Query Processing Strategies'', Computer, vol. 22, no. 3, pp. 63-70, Mar. 1989.

[14] H. Rubinovitz and B. Thuraisingham, "Design And Implementation Of A Query Processor For A Trusted Distributed Database Management System", J.Systems and Softvare, vol. 21, no. 1, Apr. 1993.

[15] P. Stachour and B. Thuraisingham, "Design of LDV-A Multilevel Secure Relational Database Management System'', IEEE Trans. On Knowledge and Data Eng., vol. 2, no. 2, June 1990.