www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 2 Issue 4 April, 2013 Page No. 1172-1183

Role Based Query Modification for Relational Database systems

Ahmed H. Arafa* , Nawal El-Fishawy, and Mervat M. Mousa
ahmed.hamdiey@yahoo.com.
* Department of Computer Science and Engineering.
Faculty of Electronic Engineering.
Menoufia University.
Menouf, Egypt.

Abstract

This paper presents a Role based query modification access control model that is implemented at database level by which fine-grained access control is handled by the underlying DBMS . The proposed model is a combination of the Role based access control model and the case statement query modification algorithm. While RBAC is used to specify the security policy for an organization , queries are modified to reflect the policy rules specified by RBAC model. The proposed model provides cell level granularity for relational database access control through a database level implementation that can't be bypassed and is independent of the underlying DBMS. It considers the insert, update and delete statements in the modification. It reduces the database size required to store the privacy meta data which will improve the performance by reducing the execution time of a given query. It also simplifies the security administration and the maintenance of users and their security policies.

Key words: Access controls, Role based access control, Case query modification, Outer join query modification.

Introduction : Access control in database is used to ensure that any access to database objects occurs according to the rules of the security policy. It is considered an important issue that is used to ensure confidentiality in database systems. Many models were proposed for controlling access in databases. These models are classified as data dependent or data independent models. Data dependent models which are derived from the database makes an access-control decision based on the content of data. One of the most popular data dependent access control models is the view based access control in which a complete view of the database is created for each user then the user queries are executed against the view instead of the base table. This technique only provides access control at the row level and is appropriate for small number of users only because the number of views will increase with increasing the number of users. Another data dependent access control model is the query modification that was first introduced by Michael Stonebraker et. al. in [1]. In this model the user queries are modified according the privacy policy rules then executed against the database but this is constrained by the features of the underlying DBMS . Data independent access control makes the access decision based on the user identity and privileges assigned to him such as discretionary access control (DAC) that was first proposed by Lampson in [2] but this model is susceptible to the Trojan horse problem and can't achieve cell level granularity for database systems. Another model in this family is the mandatory access control model (MAC) which was first introduced by LaPadula et. al. in [3] . In MAC both data and users are classified into security levels (Top Secret , Secret, Confidential and Unclassified that is ordered as TS >

S> C> U) then access to data is determined by comparing the security levels of both data and users. The MAC model may solve the Trojan horse problem through the star property but it increases the size of the database by using security class for each cell in the table and hence increases the retrieval time for queries against the database. Also MAC is suitable for military applications where the data can be easily classified that can't be achieved easily in the commercial systems . The last model in the family of data independent access control models is the Role-based access control (RBAC) that was introduced by Ferraiolo et. al. in [4]. It regulates the access of users to data on the basis of the responsibilities of the user in his organization. RBAC supports the least privilige and separation of duty principles and also simplifies the security administration , but it puts too much trust in the security administrator and can't individually achieve the cell level granularity for database systems. It is shown that none of the previous models can individually provide the finest level of security for an organization . Our objective is to propose a role based access control model by combining the NIST-RBAC model introduced in [10] and shown in figure 2 to the query modification model used in Hippocratic database that is introduced in [8]. The RBAC is used to specify the security policy by identifying both organizational hierarchy roles and administrative roles of an organization then all queries against the database will be modified to display only cells allowed by the roles assigned to the user. The work in this paper contributes in (1)- Reducing the database size required to store the privacy meta data by using the RBAC schema to store the privacy meta data instead of associating a disclosure choice table to each table in the schema. (2)-improving the performance by reducing the retrieval time of a query as a result of reducing the database size. (3)- It includes the Insert, Update, and Delete statements as well as the Select statement in the modification . (4)- It is independent of the underlying DBMS and can't be bypassed. The rest of this paper is organized as follow : Section 2 presents the related work. Section 3 illustrates the query modification models and its implementation in database systems .Section 4 presents the proposed Role based query modification model. Section 5 explains the performance study and analysis. Section 6 is the paper conclusion.

Related work: Many solutions have been proposed to enforce query modification in relational database systems. One of these solutions is DRQ-RBAC (Dynamic Rewriting of Queries for RBAC) which is presented in [5] and shown in figure 3 . In DRQ-RBAC the query rewriter adds rules in the form of WHERE clauses to create a new SQL statement that is then submitted to the DBMS to be executed. It also adds tables to the FROM clause and adds the appropriate joins to the WHERE clause in the select statement which reflects the policy rules specified for this user but this takes more time to execute the user query as a result of the join operation. Oracle has introduced the Virtual Private Database (VPD) that is based on the query modification to control database access at the row and cell level. VPD is based on creating a function to implement security policies, that returns a string which is dynamically added to the WHERE clause of the SQL statement [6]. The new rewritten SQL statement is then executed by the DBMS. VPD has some limitations such as it is not RBAC based which makes the security administration is very difficult. Also if the functions implementing the security returned a string that represent a table that doesn't exist in the original SQL query then the modified query won't be executed. Hippocratic database (HDB) which is developed at IBM's Almaden Research Center by Agrawal et al. [7][9] is another technique that limits disclosure to data for proper purposes only based on the query modification approach. HDB is developed as a middleware layer positioned between the database and any applications that may access it , so this middleware can be easily bypassed. Two algorithms modify queries in Hippocratic database which are Case statement and outer join query modification . Any of the two algorithms can be used to achieve cell level granularity based on the disclosure choices for each cell in the table and purpose recipient pairs for a given query but the Case statement always have a better performance compared to the outer join algorithm. In HDB according to the architecture shown in figure 1 , two methods are used to store the individual choices to disclose information in the database which are internal and external design. In internal design for each column , another column is associated with it in the same table that contain the disclosure choice for it. In external design , the choices for the table are stored in a separate table which means that for each table another table that contains the disclosure choices is

created. This will nearly duplicates the database size and hence increase the retrieval time for queries. Also it will increase the complexity of the database design and the possibility to change the database schema with the release of new requirements for the organization. The goal of this paper is to provide database level implementation which provides cell level access control for relational database systems based on the NIST- RBAC model and case statement algorithm where the RBAC schema will be used to store the privacy metadata and Case statement will modify the SQL queries to reflect the policy specified by the RBAC model.

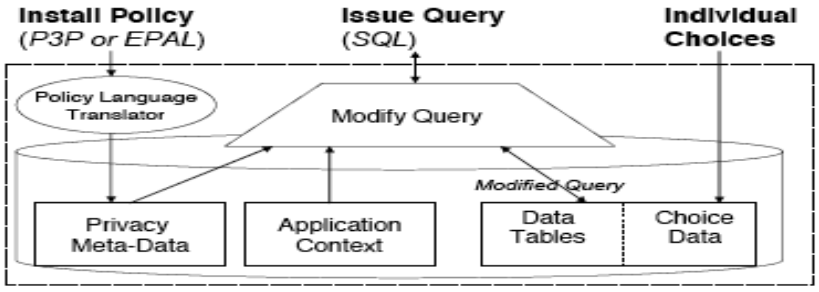

Figure 1 : Architecture overview of HDB.

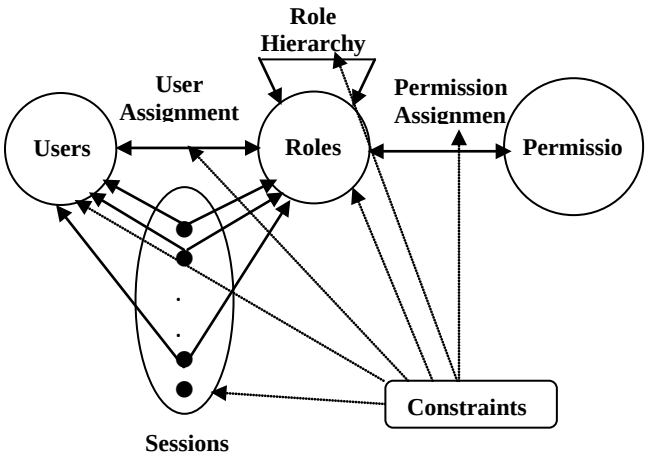

Figure 2 : NIST RBAC Model.

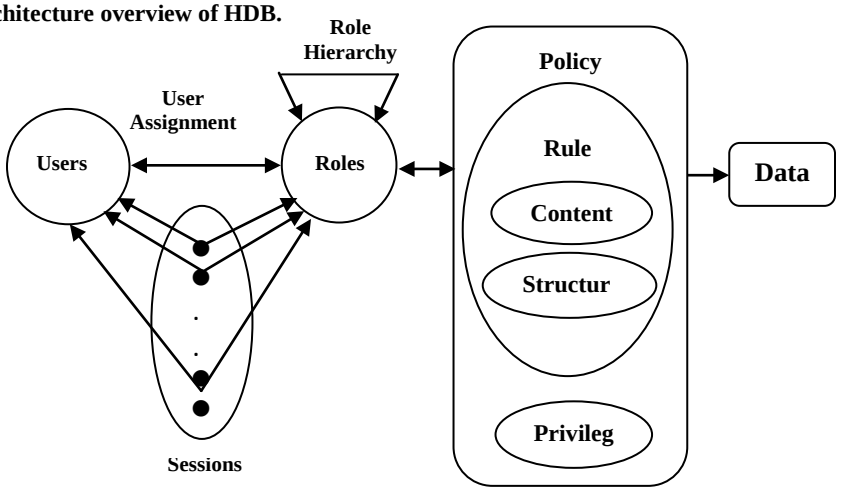

Figure 3 :DRQ-RBAC Model

3. Query modification models and its implementation: Query modification models enforces access control in database systems by modifying the user query to an equivalent query that satisfy the specified security policy. One of these models is the outer join query modification and the other is the case-statement query modification model. The two algorithms are used to modify the Select statement to limit disclosure of data to the cell level. To implement Query modification algorithms in databases , there are two approaches are used. The first one is the application level access control as shown in figure 4. In this approach, the access control is implemented as a middleware between the user and the DBMS. When a user issues a query , the middleware modifies it to another query which ensures that only authorized cells will be accessed. The modified query is then passed to the DBMS for execution. This approach is easy to be deployed in existing DBMS without the need to change the underlying DBMS. But , it has some disadvantages which are (1) it is constrained by the features of the

underlying DBMS , (2) the modified queries may be too expensive to be executed , (3) if the user has a direct access to relations in the DBMS then he can bypass the middleware which means that access policies will not be enforced. The other approach is the database level access control as shown in figure 5. In this approach, the DBMS executes the queries considering the policy rules. This can be achieved by changing the query evaluation engine of the DBMS by adding special functionalities to it. This approach it is harder to bypass the access control policies. The major drawback of this approach is that the DBMS must be changed to support the access control.

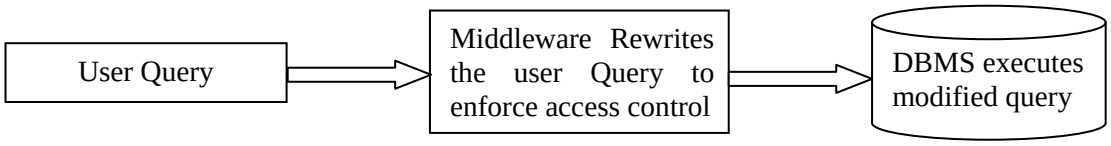

Figure 4 : Application level access control.

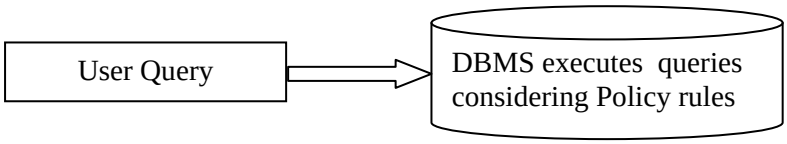

Figure 5 : Database level access control.

3.1 Left outer join query modification : This model is dependent on the concept of the left outer join used in the database theory . The incoming query is replaced by another query that enforces the access control to the cell level through a series of left outer joins among the attributes contained in the query. To illustrate this model, consider some query Q and a table T referenced by Q and contains some columns A1 ... An. Let k represent the primary key of T, and assume that the primary key is comprised of just one column then replace Q's reference to T with the following:

[σ k="allowed" (Π k(T))] ⟕ $1=$1[σ A1="allowed" (Π k,A1 (T))]... ⟕ $1=$1 [σ An="allowed" (Π k, An (T))]

Where " ⟕ " denotes the left outer join operator.

3.2 Case statement query modification: This model is dependent on the case function that is used in the SQL which facilitates conditional queries by doing the work of an IF-THEN-ELSE statement. The case statement has the following format :

```
CASE expr  WHEN comparison_expr1 THEN return_expr1  [ .....WHEN comparison_exprn THEN return_exprn
ELSE else_expr]
END
```

In this model , for each column in the project list of the received query, a case statement will be added to the from clause of the query . The case statement will test each cell in the specified field then it will return the same data contained in the database field if the cell is allowed to be disclosed by the user or it will return a null value if the user is not allowed to disclose the value contained in this filed. After adding the case statement , a where clause is added for implementing the conditions associated with each column in the role.

4. RBQM (Role Based Query Modification Model) : A model that enforces RBAC at the database level is proposed . As shown in figure 6 , RBQM takes a submitted SQL statement and modifies it according to the user's security policy which is easily specified by the RBAC model . The query modification takes place through a series of steps according to the type of the received SQL statement. This model reduces the database size required to store the privacy meta data by using the RBAC schema instead of associating disclosure choice table with each table in the database schema as used in Hippocratic database. Also it takes advantages of the RBAC model

such as simplifying the security administration and maintenance of users and their policies and supporting the separation of duty principle.

4.1 Privacy meta data in RBQM: The work done in Hippocratic database is based on the concept of the Purpose -Recipient pairs to disclose the required information for a user . The policy rules always take the form <data ,purpose-recipient pair, condition> . To store the privacy meta data, a Policy table is created to store the purpose-recipient information associated with the predicates of the rules as shown in table 1, table 2. Also for each table in the schema , another table is created to store the disclosure information for that table. As shown in the tables below , table 4 is created to contain the disclosure choices for employees information shown in table 3. This means that number of tables created in the database are duplicated which may duplicates the database size as shown in the tables below.

Table 1 :Rule table that contains Policy Rules.

Rule_id	Policy	Purpose	Recipient	Table	Column	Cond_id
R1	P1	Insurance	Billing office	Emp	phone	C1
R2	P1	Counting	Accountant	Emp	salary	-

Table 2 : Constraint.

Cond_id	Predicate
C1	Dno=2
C2	Salary>2500

Table 3 : Emp table that contains full data of Employees .

EID	ENAME	EMAIL	PHONE	HIRE_DATE	JOB_ID	SALARY	DNO
1101	mostafa	mostafa@yahoo.com	0111456223	30/1/1994	Sales_rep	9850	30
1102	Khaled	khaled@hotmail.com	0102254663	22/7/1996	receptionist	3560	40
1103	Ali	ali@yahoo.com	0125462311	30/8/2001	accountant	4500	50

Table 4 :Emp_choice table that contains Employees choices for disclosure.

EID	EIDC	ENAMEC	EMAILC	PHONEC	HIRE_DATEC	JOB_IDC	SALARYC	DNOC
1101	1	1	1	1	1	1	1	1
1102	1	1	0	0	0	0	0	1
1103	1	0	0	0	0	0	0	0

According to the architecture overview of RBQM shown in figure 7 and the Entity Relationship Diagram (ERD) schema shown in figure 6 which is corresponding to the RBAC model , A set of tables corresponding to this schema are created in the database to store the privacy meta data. For example a table is created to store the permission's information such as the permission type that reflects the operation that the user can perform on a specific field in the database such as Insert , Update , Delete or Select operation. For example permission type = 0 is used for a select statement, permission type =1 is used for update statement, permission type=2 is used for delete statement, ..etc. Each permission may have many conditions or constraints on the database fields which will result in the cell level granularity. The meta tables (from table 5 to table 11) shown below are part of the tables corresponding to the ERD schema of the RBAC model .

Table 5 : User's information table.

USER_ID	USER_NAME	PASSWORD	STATUS
12345	Admin	admin123	1
12346	Ali	Ali123	1
12347	Khaled	Khaled123	1

Table 6 : Permissions table.

PID	PTYPE	PTABLE	PFIELD
1	0	Emp	ename
2	2	Emp	
3	1	Emp	phone

Table7: Roles table.

ROLE_ID	ROLE_NAME
1	Admin
2	Employee

Table 8: Roles permissions.	
PERMISSION_ID	ROLE_ID
1	1
2	2
3	1

Table 9: User Roles.	
USER_ID	ROLE_ID
12345	1
12346	2
12347	2

Table 10: Constraints.	
Constraint_id	predicate
11	DNO=50
12	SALARY<2500
13	DNO=60

Table 11: permission constraints.	
Permission_id	Constraint_id
1	11
2	12
3	13

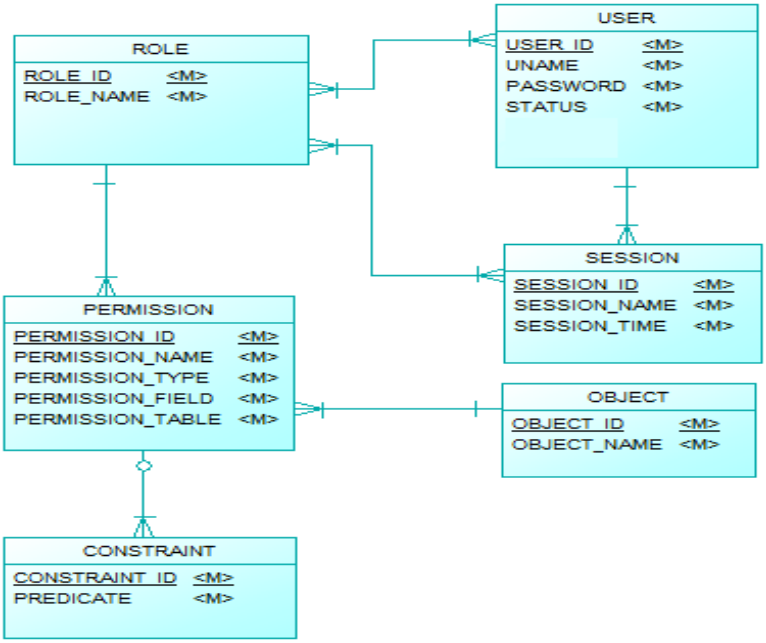

Figure 6 : ERD of RBQM Schema.

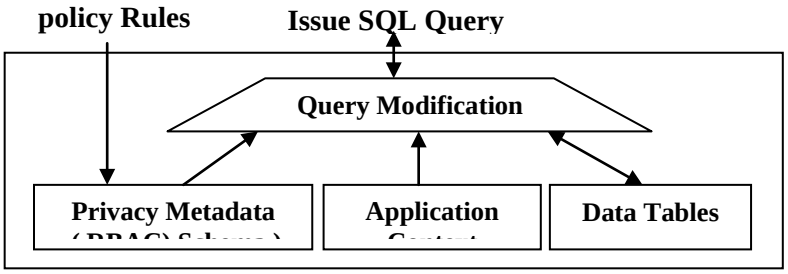

Figure 7 :Architecture Overview of RBQM.

4.2 Modification of DML statements in RBQM: RBQM can modify select, update and delete statements to reflect the policy specified by the RBAC. Once the user is logged on the DB , all of the permissions , predicates ,and role that are assigned to the user are used to modify the DML statements . The flowchart of RBQM is shown in figure 8. RBQM receives the SQL statement then checks it to determine how it will be modified. If the received statement is a Select statement , the select modifier is called which will determine whether an attribute in the select statement will be displayed without modification or it will be replaced by a null value . If an update

statement is received , the update modifier is called which tests the attributes contained in the update statement to determine if it will be rejected or modified by associating predicates to it. If the received statement is a delete statement , the delete modifier is called which modifies it by associating predicates to it or reject it if the delete operation is not allowed by the user roles . In the case of the insert statement , it will be rejected if it is not allowed by the user's roles or executed without any modification if it is allowed.

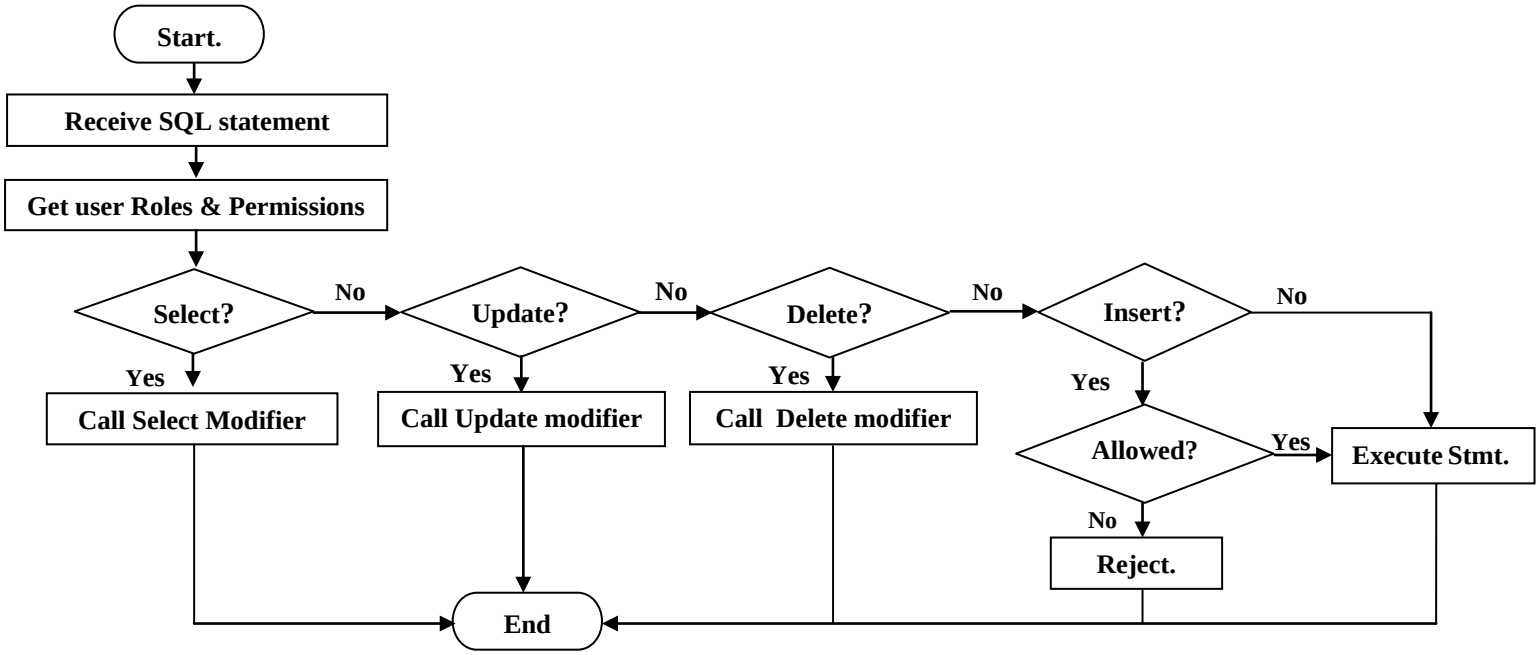

Figure 8: RBQM Flowchart.

4.2.1 Select statement modification: RBQM modifies select statement using the case statement algorithm to another select statement that satisfies privacy meta data of the RBAC schema. A case function is added corresponding to each field in the project list of the incoming select statement. The case function returns either the value contained in the field if this attribute is allowed by the roles assigned to the user or it will return a null value if it isn't allowed to be disclosed by this user. Also a set of row predicates are added to the query which limits the rows displayed to those rows that are allowed by the roles assigned to the user. If all cells in a row are null values then the whole row will not be displayed. The steps of producing the modified query are shown in the select modifier flowchart as shown in figure 9.

For example consider the select statement : **SELECT ename , salary FROM emp**. is issued by the Admin (user_id=12345). It will be modified to the following query:

```
SELECT ename, salary  FROM  ( Select
case  when exists ( select *  from  permission where  pfield ='ename' and ptable='emp' and ptype=0  and pid in ( select pid from
role_permissions where rid in (select rid from user_role where u_id=12345) ) ) then ename  else null end as ename  ,
 case when exists ( select *  from  permission where  pfield ='salary' and ptable='emp'and ptype=0 and pid in ( select pid from
role_permissions where rid in (select rid from user_role where u_id=12345) ) ) then salary  else null end as salary
from emp  where DNO=50 )
where  ename is not null or salary is not null
```

As shown in the previous meta tables of the RBAC schema , this user is assigned the Admin role which contains permission1 and permission 3. Permission 1 allows the user to disclose the ename of employees with DNO= 50 according to the constraints table , while permission 3 allows the user to update the phones of employees with DNO=60. So the result of this query after modification will display the ename values of

employees with DNO=50 while it will display null values for the salary filed as shown in Table 13 shown below.

Table 12 : Select query result.

ename	salary
ali	

4.2.2 Update statement modification: In RBQM, the update statement can be modified to another update statement that satisfy the constraints allowed by the user's roles. This can be accomplished by adding a predicate to the update statement that determines the rows in the database that can be updated by the roles assigned to the user. The steps required to produce the modified update statement are shown in the update modifier shown in figure 10. For example assume that the user Ali issued the following update statement : **update emp set salary= salary+2500** then it will be modified to the following statement .

```
Update emp
set salary= salary+2500
where exists (select * from permission where pfield ='salary' and ptable='emp' and ptype=1 and pid in ( select pid from role_permissions where rid in (select rid from user_role where u_id=12346) ) )
```

As shown in the previous meta tables of the RBAC schema , this user is assigned the employee role . This role has a delete permission on the emp table for rows with salary < 2500 , so this query will be rejected and the user is notified through an error message that he isn't allowed to execute this statement.

4.2.3 Delete statement modification: Delete statements also can be modified to an equivalent statement that satisfies the constraints of the permissions of roles assigned to the user . This modification takes place by adding a predicate that reflects the constraints to the delete statement. The flowchart of the delete modifier is shown in figure 11. For example if the user Khaled writes the following delete statement: **Delete from emp.** then it can be modified to the following delete statement:

```
Delete from emp
where exists (select * from permission where ptable='emp' and ptype=2 and pid in ( select pid from role_permissions
where rid in (select rid from user_role where u_id=12347) ) )
and salary < 2500.
```

From the previous meta tables of the RBAC schema , this user is assigned the employee role that only has a permission that allows its owner to delete all employees with salary < 2500 , so the statement will delete all employees with salary < 2500 from the emp table.

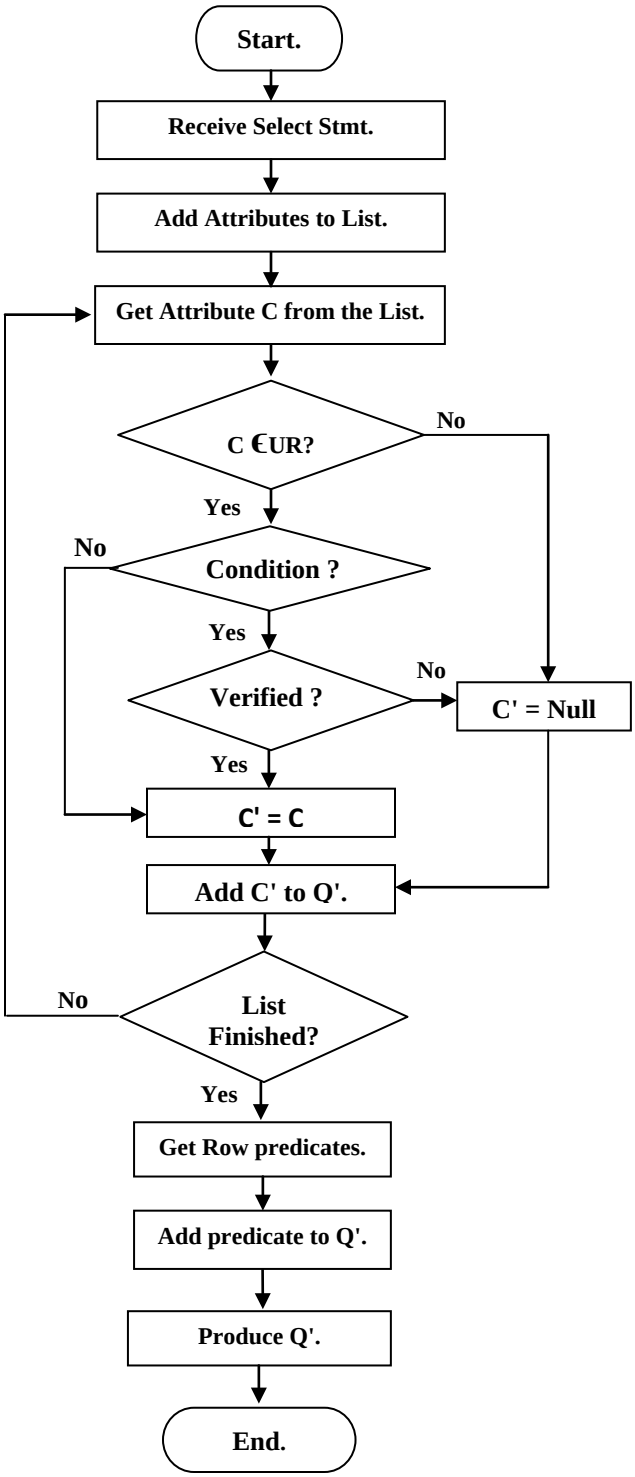

Figure 9: Select Modifier

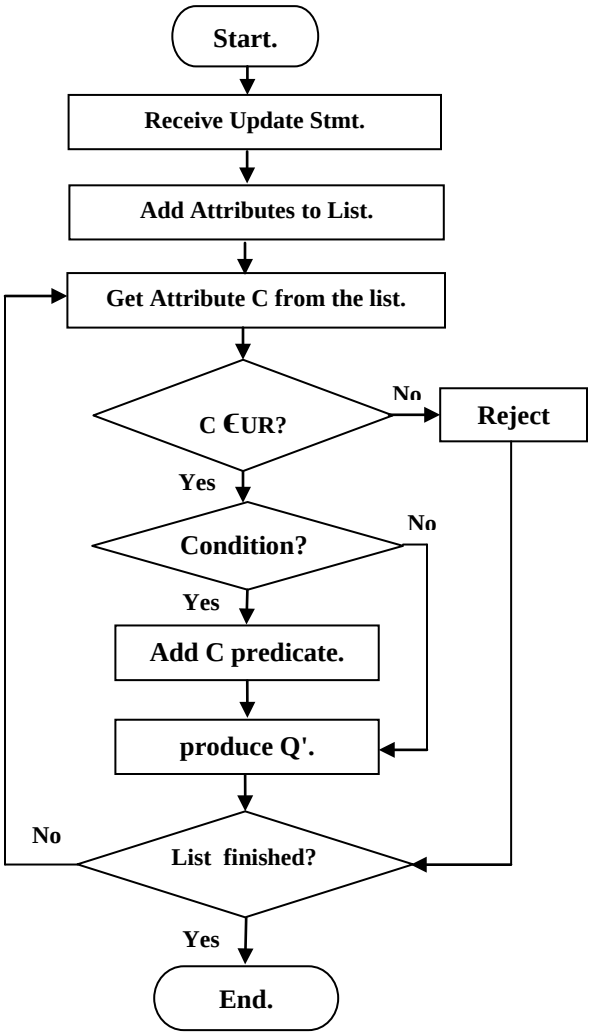

Figure 10:Update Modifier

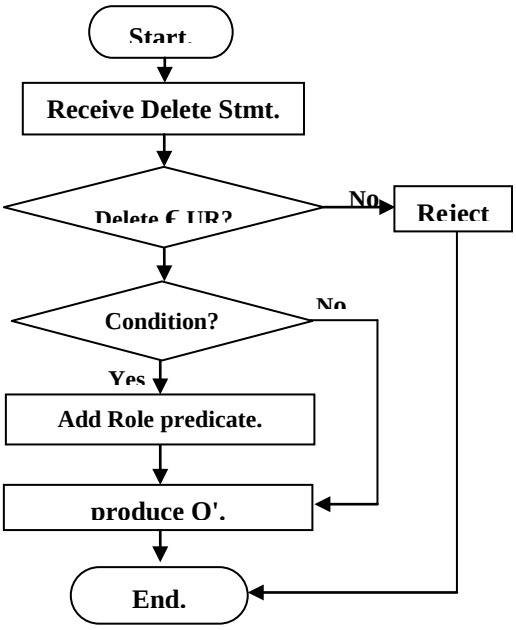

Figure 11 :Delete Modifier

5. Performance Evaluation: The performance of RBQM against Case modification as used in Hippocratic database and Outer join query modification is described. In all experiments the cost of executing queries (execution time) at different database sizes and different number of attributes are measured. This include :
(1)-impact of varying the number of records (data base size) on a given select query.
(2)-impact of varying the number of attributes on a given select query.

5.1 Experiment setup: All experiments were run using ORACLE DB 10G. The operating system was Microsoft Windows XP, Service Pack 3. The hardware consisted of single core 2.8GHz Intel machine with 1 GB of memory and 160 GB IDE Western Digital hard drive. The cost of executing queries is measured using the Toad utility. Each query was run 5 to 10 times. The results in 5.2 give the average performance numbers for each query.

5.2 Experimental Results and Analysis:

5.2.1 Impact of varying the number of records: In this experiment, for the RBQM ,Case query modification and Outer join query modification algorithms , The execution time of the select query is measured at different number of records in the tables considering the presence of join operation in the query. The database size in terms of the number of records were varied from 0.5 to 3 million record while the number of attributes in the query were kept fixed (only 3 attributes were used in the query) . The results of execution are shown in figure 12 and figure 13 for select and join query respectively. From these figures , the increase in the database size has a slight effect on the execution time of both RBQM and Case statement algorithms while it has a bad effect on the execution time of the outer join algorithm. Also the better performance RBQM is due to the reduction in the database size which is gained as a result of using RBAC schema to store the privacy meta data.

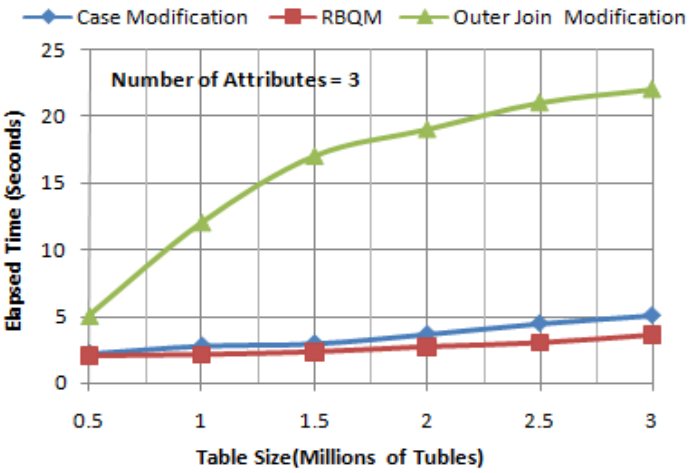

Figure12 : Impact of varying the number of records on the execution time of a select query.

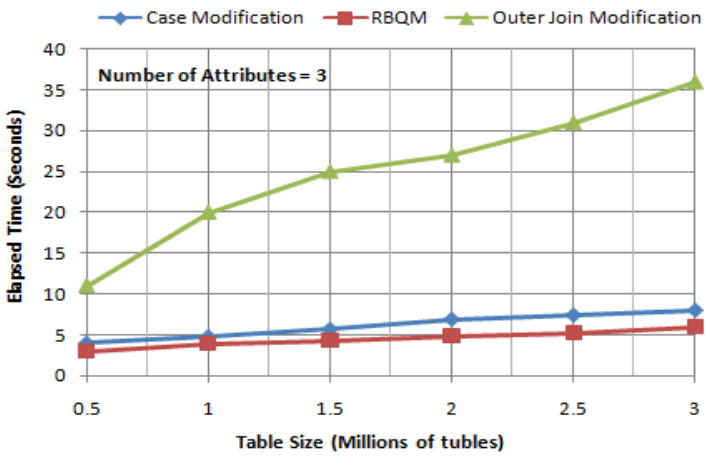

Figure13 : Impact of varying the number of records on the execution time of join query.

5.2.2 Impact of varying the number of attributes: In this experiment , the execution time of both select and join query is measured for RBQM, Case and outer join algorithms at different numbers of attributes in the query while the database size (number of records) are kept constant at 2 million records and using 2,3,4,5,6

attributes in the query. The results of execution are shown in figure 14 , figure 15 for both select and join query respectively. From these figures , the number of attributes has been significantly increased the execution time of the Outer algorithm while it has a slight effect on execution time of both RBQM and Case query modification. Also the better performance of RBQM is due to the reduction in database size.

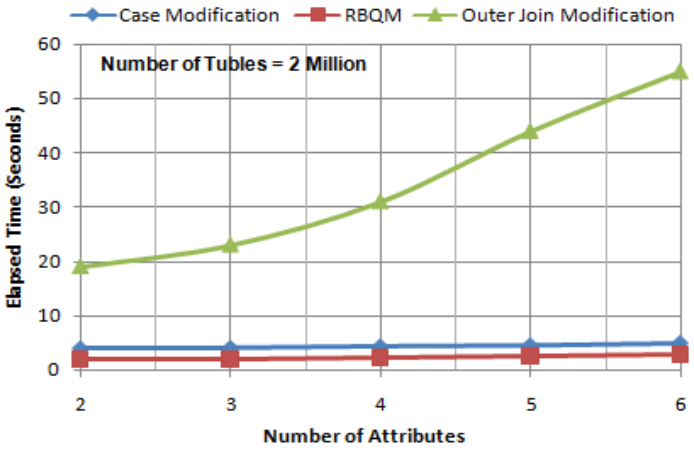

Figure14 : Impact of varying the number of attributes on the execution time of the select query.

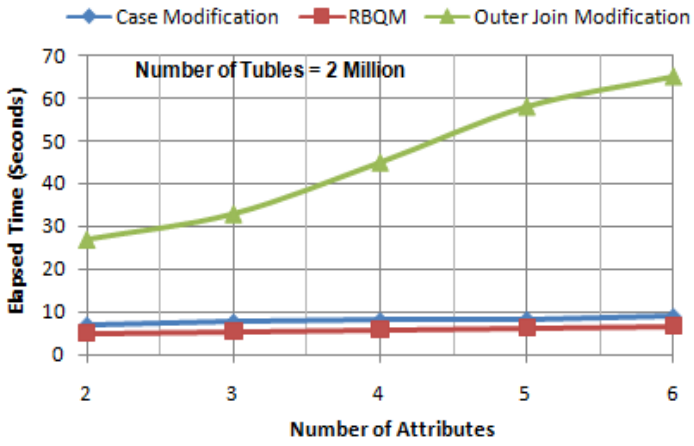

Figure15 : Impact of varying the number of attributes on the execution time of the join query.

6. Conclusion : In this paper , a role based access control model based on the Case statement query modification is proposed . Also the performance of the proposed model was evaluated by measuring the execution time while varying the number of records and attributes in both select and join queries. Using the RBAC schema to store the privacy meta data instead of using a disclosure choice table for each table in the database will reduce the database size which in turn improves the performance by reducing the retrieval time of a given query.

References:

[1] M. Stonebraker and E. Wong." Access control in a relational data base management system by query modification" In ACM/CSC-ER, 1974.

[2] B.W. Lampson." Protection ". 5th Princeton Symposium on Information Science and Systems, pages 437–443, 1971. Reprinted in ACM Operating Systems Review 8(1):18–24, 1974.

[3] D.E. Bell and L.J. LaPadula. "Secure computer system ": Unified exposition and multics interpretation. Technical Report ESD-TR-278, vol. 4, The Mitre Corp., Bedford, MA, 1973.

[4] David F. Ferraiolo and D. Richard Kuhn" Role-Based Access Controls", 15th National Computer Security Conference ,1992 . pages 554 - 563.

[5] Jay Jarman, James A. McCart, Donald Berndt, Jay Ligatti " A Dynamic query-rewriting mechanism for role-based access control in databases " 14th Americas Conference on Information Systems, AMCIS 2008, Toronto, Ontario, Canada, August 14-17, 2008

[6] "Oracle Virtual Private Database" An Oracle Database 10g Release 2 White Paper June 2005 .

[7] Tyrone Grandison, Christopher Johnsonand Jerry Kiernan " Hippocratic Databases: Current Capabilities and Future Trends " Handbook of Database Security - Applications and Trends. Springer 2008, pp 409-429. ISBN 978-0-387-48532-4

[8] K. LeFevre, R. Agrawal, R. Ercegovac, R. Ramakrishnan, Y. Xu, D. DeWitt, “Limiting Disclosure in Hippocratic Databases,” In Proceedings of the 30th International Conference on Very Large Databases, Toronto, Canada, August 2004.

[9] Markus Kirchberg, Sebastian Link " Hippocratic Databases: Extending Current Transaction Processing Approaches to Satisfy the Limited Retention Principle " Proceedings of the 43rd Hawaii International Conference on System Sciences - 2010

[10] Ravi S. Sandhu, Edward J. Coynek, Hal L. Feinstein and Charles E. Youmank “Role-Based Access Control Models“ IEEE Computer, February 1996, Volume 29, Number 2, pages 38-47.