ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 4, April 2014

Role-Based Access Control Approaches In Mangodb 2.4 and Informix Online Dynamic Server Version 7.2

Abubakar Sulaiman Gezawa[1], Ahmed Aliyu[2], Tong Yujun[3], Saifullahi Aminu Bello[4], Abubakar Ado[5]

System Analyst, MIS Department, Bayero University Kano, Kano, Nigeria [1]

Lecturer, Department of Computer Science, Bauchi State University, Bauchi, Nigeria [2]

Professor, Electrical/Electronics, Liaoning University of Technology, Jinzhou, Liaoning Province, China [3]

Lecturer, Department of Computer science, Kano University of Science and Technology, Kano, Nigeria [4]

Lecturer, Department of Computer Science, Northwest University, Kano, Nigeria [5]

Abstract: This paper compares and contrast role based access control (RBAC) approaches in mangodb 2.4 and Informix Online Dynamic Server Version 7.2. We categorize RBAC features under two major areas. User role assignment and assigning privileges. Many commercially successful access control systems for mainframes implement role for secondary administration for example, an operator role could access all resources but not change access permissions, a security officer role could access all resources but have no access e.t.c. my findings is that these products provide a sound basis for implementing the basic features of RBAC, although there are significant differences. In particular, Informix restricts users to a single active role at any time, while mangodb allow multiple roles to be activated simultaneously as per the user's selection.

Keywords: RBAC; features; privileges; Approaches; Dynamic; Permission

I. INTRODUCTION

In computer systems security, role-based access control (RBAC) [1][2] is an approach to restricting systems access to authorized users. It is used by the majority of enterprises with more than 500 employees [3] and can implement mandatory access control (MAC) or discretionary access control (DAC) RBAC is sometimes referred to as role based security. In RBAC permissions are associated with roles, and users are made members of appropriate roles thereby acquiring the roles permissions. This greatly simplifies management of permissions. Roles are created for the various job functions in an organization and users are assigned roles based on their responsibilities and qualifications.

In DBMS, The RBAC model forms an integral component of the access control mechanism, and hence the RBAC model data is used for enforcing access control on the various resources (database objects) under the control of the DBMS product [4]. An application system developed using a DBMS can contain a large amount of data with highly differentiated access permissions for different users depending upon their function or role within the organization. Hence database management is a key area which needs mechanisms for management of authorization or privileges.

In this paper, we analyze compare and contrast RBAC approaches implemented in mangodb 2.4 and Informix Online Dynamic Server Version 7.2. The RBAC features that are supported have been categorized under two broad areas as follows

- User role assignment
- Assignable privileges

In comparing the features of this complex commercial software packages, it is not always possible to readily obtain the total set of all supported features from product manuals alone. However, it is possible to extract and compare the major differences in features from using the multiple manuals that come with the product this is the approach that has been adopted in this paper.

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 4, April 2014

The paper is organized as follows .section II give some brief discussion on related work done by some scholars about RBAC approaches in some commercial databases while section III and IV describes the roles and privilege assignment of both the two database, then section V presents the results of the discussion and finally section VI presents the conclusion.

II. RELATED WORK

Phillippe Balbiani [6] proposed an access control language in which RBAC, and following extensions namely Delegation, separation of duty and history based access control can be encoded. In contrast with Cassandra, they have not promoted role management mechanism to first-class citizenship, and have based their model on the assumption that access control system could be separated into a dynamic part that evolves according to action performed by users and a static part. Ravi S. Sandhu [7] introduces introduce a family of reference models for role-based access control (RBAC) in which permissions are associated with roles, and users are made members of appropriate roles. This greatly simplifies management of permissions. Roles are closely related to the concept of user groups in access control. However, a role brings together a set of users on one side and a set of permissions on the other, whereas user groups are typically defined as a set of users only [5].

III. INFORMIX ONLINE DYNAMIC SERVER VERSION 7.2

3.1 USER ROLE ASSIGNMENT

A role can be granted to a single user, a role, a list of users or-by using the keyword PUBLIC-to all users [5]. A user can be granted more than one role. Users who have been granted a role with the GRANT OPTION can further grant that role or delete it by using the DROP ROLE command.

A user can have only one role active at any point in time. Initially all users are assigned the role NULL or NONE, by default, when they sign on to a database. The user can enable an authorized role by means of the SET ROLE statement. The SET ROLE statement allows for specifying only one role, so the user can enable only one role at a time. Moreover, if a user executes the SET ROLE statement after a role is already set, the new role replaces the old role. This implies that a user can be active in one and only role at every moment. Informix provides no feature to specify a default active role, different from NULL or NONE, for a user.

3.2 Support for role relationships and constraints

As already stated, users who have been granted a role with GRANT OPTION as well as DBAs can grant a role to another role. This feature enables building nested roles, so it is possible to build a role hierarchy.

Informix has no features to specify mutually exclusive roles that are sets of roles which cannot be granted to the same user. Hence it does not support static separation of duty. There is also no support for cardinality constraint to restrict the maximum or minimum number of users that can be authorized for a role. Informix does in a sense support dynamic separation of duties that is specification of roles that cannot be simultaneously activated. However, this is more a side effect of the fact that only role can be activated at a time rather than an independent feature in its own right.

3.3 Assignable privileges

Informix divides the universe of all privileges that can be assigned into three categories: database-level privileges, table-level privileges and execute privilege.

Database-level privileges refer to privileges needed to connect to a database, add new objects and perform administrative functions like security management (including transfer of object ownerships) and space management etc. They include the CONNECT privilege (ability to establish the user context to a database schema so that the user can query and modify the objects in the schema depending upon the permissions and ownerships), RESOURCE privilege (ability to create new objects in a database schema like tables, indexes and procedures) and DBA privilege (grant privileges to another user or role, create new objects under a designated ownership the default owner of a database object is the one who created it, update rows of system catalog tables and control the growth of physical spaces by altering extent sizes etc)[6].

Table-level privileges refer to privileges that can be granted on a base table [6]. They include INSERT, DELETE and ALTER that are applicable for the table as a whole, SELECT and UPDATE privileges that can be selectively applied on one or more columns of a table, as well as REFERENCES (ability to reference one or more columns in referential constraints) and INDEX (ability to create permanent indexes). Privileges that can be granted on a view are SELECT, INSERT, DELETE and UPDATE. The last three privileges are only applicable if the view meets all the requirements for updating (updatable view). ALTER, REFERENCES and INDEX privileges cannot be granted on a view.

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 4, April 2014

The EXECUTE privilege is applicable only for database stored procedures. It is a single privilege representing the ability to execute the stored procedure. Informix allows only the Table-level privileges and the EXECUTE privilege to be granted to roles. Database-level privileges cannot be granted to roles.

The DBA and the owner of a database object can grant privileges to a role and can revoke that privilege later on. Informix has a AS GRANTOR clause in the statement that grants privileges to roles. Using this it is possible to designate someone else as the grantor of the specified privilege to a role. However the person who originally executed the grant statement with AS GRANTOR option can no longer revoke that privilege from the role. It is also interesting to note that the user who has been granted a role WITH GRANT OPTION can also revoke privileges from a role.

IV. MANGODB 2.4

4.1 USER ROLE ASSIGNMENT

Roles: Roles in Mongodb provide users with a set of specific privileges, on specific logical databases. Users may have multiple roles and may have different roles on different logical database. Mongodb provides built-in roles, each with a dedicated purpose for a common use case.

The major roles in Mangodb are [5]:

- read
- readWrite
- dbAdmin
- userAdmin
- clusterAdmin
- readAnyDatabase
- readWriteAnyDatabase
- userAdminAnyDatabase
- dbAdminAnyDatabase

4.2 **Role Assignment to Users**

User administrators create the users that access the system's databases. Through "*user management commands*" let administrators create users and assign them roles. The first role assigned in a database should be either userAdmin or userAdminAnyDatabase. This user can then create all other users in the system. Table 1 below shows User Management Commands according to [5].

Name	Description
createUser	Creates a new user.
updateUser	Updates a user's data.
dropUser	Removes a single user.
dropAllUsersFromDatabase	Deletes all users associated with a database.
grantRolesToUser	Grants a role and its privileges to a user.
revokeRolesFromUser	Removes a role from a user.
usersInfo	Returns information about the specified users.

Table 1: User Management Commands according to [5].

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 4, April 2014

Table 2 below shows Role Management Commands according to [5].

Name	Description
createRole	Creates a role and specifies its privileges.
updateRole	Updates a user-defined role.
dropRole	Deletes the user-defined role.
dropAllRolesFromDatabase	Deletes all user-defined roles from a database.
grantPrivilegesToRole	Assigns privileges to a user-defined role.
revokePrivilegesFromRole	Removes the specified privileges from a user-defined role.
grantRolesToRole	Specifies roles from which a user-defined role inherits privileges.
revokeRolesFromRole	Removes specified inherited roles from a user-defined role.
rolesInfo	Returns information for the specified role or roles.

Table 2: Role Management Commands according to [5].

The following example creates the user recordsUserAdmin on the records database [5]:

```
db.createUser(
  {
    user: "recordsUserAdmin",
    pwd: "password",
    roles:
    [
      {
        role: "userAdmin",
        db: "records"
      }
    ]
  }
)
```

4.3 Assignable privileges

A privilege consists of a specified resource and the actions permitted on the resource. A privilege *resource* is a database, collection, set of collections, or the cluster. If the cluster, the affiliated actions affect the state of the system rather than a specific database or collection.

An *action* is a command or method the user is allowed to perform on the resource. A resource can have multiple allowed actions For example; a privilege that includes the update action allows a user to modify existing documents on the resource. To additionally grant the user permission to create documents on the resource, the administrator would add the insert action to the privilege. A role can include one or more existing roles in its definition, in which case the role inherits all the privileges of the included roles. A role can inherit privileges from other roles in its database. A role created on the admin database can inherit privileges from roles in any database.

The following is a sample document for a user-defined role appUser defined for the myApp database [5]:

```
{
  _id: "myApp.appUser",
  role: "appUser",
  db: "myApp",
  privileges: [
      { resource: { db: "myApp" , collection: "" },
        actions: [ "find", "createCollection", "dbStats", "collStats" ] },
```

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 4, April 2014

```
        { resource: { db: "myApp", collection: "logs" },
          actions: [ "insert" ] },
        { resource: { db: "myApp", collection: "data" },
          actions: [ "insert", "update", "remove", "compact" ] },
        { resource: { db: "myApp", collection: "system.indexes" },
          actions: [ "find" ] },
        { resource: { db: "myApp", collection: "system.namespaces" },
          actions: [ "find" ] },
    ],
    roles: []
}
```

The privileges array lists the five privileges that the appUser role specifies:

- The first privilege permits its actions ("find", "createCollection", "dbStats", "collStats") on all the collections in the myApp database *excluding* its system collections.
- The next two privileges permit *additional* actions on specific collections, logs and data, in the myApp database.
- The last two privileges permit actions on two *system collections* in the myApp database. While the first privilege gives database-wide permission for the find action, the action does not apply to myApp‘s system collections. To give access to a system collection, a privilege must explicitly specify the collection.

As indicated by the empty roles array, appUser inherits no additional privileges from other roles.

V. Result Discussion

A summary of role based access control features that are supported or not supported in the two DBMS products studied in this paper is given in table 3 below.

Item	Feature	Informix	Mangodb
1	Ability for a role grantee to grant that role to other users	Yes	Yes
2	Build a role hierarchy	Yes	Yes
3	Multiple active roles for a user session	No	Yes
4	Grant DBMS System Privileges to a Role	No	Yes

Table 3: A summary of role based access control features that are supported or not supported in the two DBMS

Features 1 and 3 pertain to user role assignment

Features 2 pertain to support for role relationships and constraints

Features 4 pertain to assignable privileges

TABLE 3: SUMMARY

OBJECT PRIVILEGES ALLOW USERS TO PERFORM A PARTICULAR ACTION ON A SPECIFIC TABLE, VIEW, SEQUENCE OR STORED PROCEDURE. THEY INCLUDE THE SELECT, UPDATE, INSERT, DELETE OPERATIONS ON TABLES AND VIEWS, ALTER, CREATE INDEX OPERATIONS ON TABLES ALONE, AND EXECUTE OPERATION ON PROCEDURES AND FUNCTIONS

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 4, April 2014

Both categories of privileges can be granted to roles. System Privileges can only be granted by the DBA or by a user who has been granted that privilege with the ADMIN OPTION. Object Privileges can only be granted to roles either by owner of the object or by a user who has been granted that privilege with the GRANT OPTION.

VI. CONCLUSION

In the area of user role assignment, we found that in both mangodb 2.4 and Informix Online Dynamic Server Version 7.2. The task of assigning roles to users can be implemented as a discretionary access control mechanism by enabling the role grantee to grant that role to other users. While Mangodb provide for multiple roles to be activated in a user session, Informix has provision for only one role to be active. Since Informix does support role hierarchies it can be argued that by suitable definition of roles and by use of role-to-role assignments, this limitation can be overcome. However, this would require anticipation of role combinations that users would like to activate and definition of a senior role in the hierarchy which combines these together in one.

In summary we found that Mangodb provide more features than Informix in the areas of user role assignment and assignable privileges. Overall our conclusion is that these products provide a sound basis for implementing the basic features of RBAC, although there are significant differences.

REFERENCES

[1] David Ferraiolo, Janet Cugini, and Richard Kuhn. Role-based access control (RBAC): Features and motivations. In Proceedings of 11th Annual Computer Security Application Conference, pages 241-48, New Orleans, LA, December 11-15 1995.

[2] David Ferraiolo and Richard Kuhn. Role-based access controls. In Proceedings of 15th NIST-NCSC National Computer Security Conference, pages 554{563, Baltimore, MD, October 13-16 1992.

[3] Luigi Guiri. A new model for role-based access control. In Proceedings of 11th Annual Computer Security Application Conference, pages 249{255, New Orleans, LA, December 11-15 1995.

[4] Informix. INFORMIX - Online Dynamic Server - Administrator's Guide - Version 7.2, Vol 1 and 2, 1997.

[5] http://www.mongodb.org

[6] Informix. INFORMIX Guide to SQL : Tutorial, 1997.

[7] Bhamidipati, V. and Sandhu, R.. Push architectures for user-role assignment. Proceedings 23th National Information Systems Security Conference (NISSC), Baltimore, pp. 89-100, 2000.

[8] Cheng, E. C. An object-oriented organizational model to support dynamic role-based access control in electronic commerce applications. Systems Sciences, HICSS-32, 1999, http://www.ieeexplore.ieee.org/iel5/6293/16788/00773053.pdf

[9] Epstein, P. and Sandhu, R. Towards a UML based approach to role engineering. Proceedings of the Fourth ACM Workshop on RBAC, pp. 135-143, October 1999.

[10] Ferraiolo, D., Gilbert, D. M. and Lynch, N. An examination of federal and commercial access control policy needs. Proceedings of NIST-NCSC National Computer Security Conference, pp. 107-116, Baltimore, MD, September 20-23 1993.

[11] Gutzmann, K. Access control and session management in the HTTP environment. IEEE Internet Computing , Vol. 5, No. 1, pp. 26-35, Jan.-Feb. 2001.

[12] S. H. von Solms and Isak van der Merwe. The management of computer security profiles using a role-oriented approach. Computers & Security, 13(8):673-680, 1994.

[13] M.-Y. Hu, S.A. Demurjian, and T.C. Ting. User-role based security in the ADAM object-oriented design and analyses environment. In J. Biskup, M. Morgernstern, and C. Landwehr, editors, Database Security VIII: Status and Prospects. North- Holland, 1995.

BIOGRAPHY

Abubakar Sulaiman Gezawa: is a system analyst at the department of Management Information System (MIS), Bayero University, Kano. He is currently a master student of Computer science and Technology, Liaoning University of Technology, Jinzhou, Liaoning Province, China. His area of interest is knowledge discovery and Intelligent decision support system. He has published a paper titled "A B-S Model for Online Integrated Information system for Bayero University Kano, Nigeria".

PHTOGRAPH

Ahmed Aliyu is working with the department of computer Science at Bauchi State University, Bauchi-Nigeria. He Obtained his B. Tech Computer Science from Abubakar Tafawa Balewa, University,Bauchi-Nigeria. He is currently a master's student at Liaoning University of technology, Jinzhou-China. His area of interests is Computer network security and information system. He has published a paper titled "An Integrated Framework for Detecting and prevention of Trojarn Horse (BINGHE) in a Client-Server Network".

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

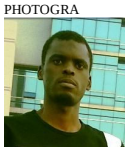
(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 4, April 2014

Tong Yujun: is lecturer at School of electrical/Electronic Engineering Liaoning University of Technology.He is currently an Associate Professor, his Research area include: Web Data Mining, Distributed Database and Software Engineering.

Saifullahi Aminu Bello: is working with department of computer science, Kano University of Science and Technology, Kano-Nigeria. He obtained his B.sc computer science from the same university. And currently studies M.sc Computer Science and Technology at Liaoning University of Technology, Jinzhou-China. His area of interests is Data mining and distributed databases. He received an award as the best graduating student in B.sc Computer Science 2008/2009 by Kano University of Science and Technology, Nigeria.

PHOTOGRA



Abubakar Ado: is working with department of Computer Science, Northwest University, Kano-Nigeria. He Obtained his B. Tech Computer Science from Abubakar Tafawa Balewa,] Bauchi-Nigeria. His area of interests is Database and information system. He has published a Paper titled “Building a Diabetes Data Warehouse to Support Decision making in healthcare industry”.