Formal Specification and Impementation of RBAC Model with SOD

Su Yu
Shanghai University of Engineering Science/ College of Electronic and Electrical Engineering, Shanghai, China
suyu_sh@hotmail.com

Jon. J. Brewster
Lawrence Technological University/ Department of Mathematics and Computer Science, Southfield, USA
jbrewster@ltu.edu

***Abstract*—RBAC(Role-Based Access Control) is an efficient and safe role_based access control mechanism. Separation Of Duty (SOD) is one of the most expected characteristics of RBAC system and also is one of the main characteristics of secure system. This paper presents SOD's attributes of RBAC model by formal specification and their relations by state graph. This paper also explains a program for implementing SOD constraint in MIS by relationship between tables.**

***Index Terms*—-Role-Based Access Control(RBAC), formal specification, Separation Of Duty (SOD), Security, constraint**

I. INTRODUCTION

The concept of access control, or authorization, is as old as multiuser computers. Authorization allows users to access computer system resources, and as it is usually used in today's information technology it maps the set of users onto the set of permissions, that is,"who can access what". Access control continues to be a fundamental security mechanism. It is a feature of virtually all IT systems from the ubiquitous PC to the various enterprise computing architecture and administrative management. In decade and a half, Role-Based Access Control (RBAC) [1] has come to play a growing role in authorization.

Compared with traditional lattice-based access control policies, such as Discretionary Access Control (DAC) and Mandatory Access Control (MAC) developed primarily for military systems(see Sandhu [2] for a discussion of these), RBAC can more effectively meet the needs of commercial systems[3][4]. A set of roles is introduced in RBAC, which is interposed between user and permission. That is, users are mapped into roles, and roles in turn are mapped into permissions, as shown in Figure 1. Security administration is greatly simplified by the use of roles to organize access privileges, because there are many users that map to a given role, and the permissions for that role need only be defined once for each role. For example, if a user moves to a new functional role within the organization, the user can simply be assigned to the new role and removed from the old one, whereas in the absence of an RBAC model, the user's old permissions would have to be individually revoked, and new permissions would have to be granted. These permissions would typically include access to role-specific files and programs.

Another benefit of the RBAC access control model is its use to implement Saltzer and Schroeder's principle of "least privilege". [5] The principle states that all users should have only the privileges that they need to do their work and no more. Because that refers to roles rather than the identity of users as such, RBAC is far more convenient and logical as an access control mechanism.

A further reduction in the privilege accorded each role is described as "Separation of Duty" and is a familiar technique employed in financial world. In order to reduce the chance of fraud, key transactions, such as disbursements, require the cooperation of two employees to carry out two halves of the task. For example, one employee may be tasked with verifying the receipt of goods, while a second one receives the checked receipt and actually transmits a payment to the vendor. Such SOD systems can be created using the appropriate version of RBAC system.

The RBAC96 model is actually a model composed of four sub-family of models [6][7] (see also [8]), each sub-model are given in the corresponding formal specification. The two main extended fields of RBAC96 are SOD constraints and delegation. [21]

SOD is very important in business system, also is one of the most desired features in RBAC system. Administration constraints may need to be enforced to prevent information misuse and prevent fraudulent activities. A typical authorization constraint, broadly relevant and well recognized, is separation of duties (SOD). Reducing the risk of fraud by not allowing any individual to have sufficient authority within the system to single-handedly perpetrate fraud is the intent of SOD. Such constraints can be easily expressed using an RBAC model through SOD constraints on roles, user-role assignments, and role-permission assignments. Furthermore, using constraints on the activation of user assigned roles, users can sign on with the least privilege set required for any access. In case of inadvertent errors,

doi:10.4304/jsw.7.4.870-877

such least privilege assignments can contain damage. Simon[9] divided SOD into five categories: static SOD, simple dynamic SOD, object-based SOD, operation-based SOD, history-based SOD. Gligor, Gavrila and Ferraiolo [10] gave the formal specification of separation of duties policy and put forward some new SOD variants, such as object-based static SOD, object-based dynamic SOD and so on. Ahn and Sandhu [11] described the separation of duty constraint by RSL99 language, and proposed session-based SOD and user-based SOD.

In previous document, the technique of implementing SOD was introduced. (See also [12].) This paper presents a detailed implementation program, including time and frequency on restrictions, with some innovative and practical value. First section describes the basic model RBAC96, section II describes the properties of SOD constraint with formal specification, Section III presents an execution program to achieve SOD constraint, including time and frequency on restrictions, section IV concludes the paper and propose future research directions.

II. RBAC96 BASIC MODEL

RBAC96 is a model family, including four sub-models: RBAC0, RBAC1, RBAC2 and RBAC3. RBAC0 is the basis for the other three sub-models. The basic concept of RBAC0 is that users are assigned to roles, permissions are assigned to roles, and users acquire permissions by being members of roles. RBAC0 includes requirements that user-role and permission-role assignment can be many-to-many. Thus the same user can be assigned to many roles and a single role can have many users. For permissions, a single permission can be assigned to many roles and a single role can be assigned to many permissions. RBAC0 defined as follows:

user set: users(U), role set: roles(R), permission set: permissions(p), session set: sessions(S).

$PA \subseteq PR$, permission-role assignment can be many-to-many, thus a single permission can be assigned to many roles and a single role can be assigned to many permissions.

$UA \subseteq U \times R$, user-role assignment can be many-to-many, thus the same user can be assigned to many roles and a single role can have many users.

user: $S \rightarrow U$, a user can be assigned to many sessions but a session can be only assigned to a user.

roles: $S \rightarrow 2R$, he function maps a session to a role set.

$roles(Si) \subseteq \{r | (user(Si), r) \in UA\}$, Session contains the user's active roles.

RBAC1 is known as hierarchical RBAC model. Hierarchical RBAC1 adds requirements for supporting role hierarchies. A hierarchy is mathematically a partial order defining a seniority relation between roles, whereby senior roles acquire the permissions of their juniors, and junior roles acquire the user membership of their seniors. The inheritance is partial order. that is transitive, reflexive and anti-symmetry. RBAC1 increased the following two entities : $RH \subseteq R \times R$, A partial orders. Partial order can be written as $\geq$, Such that $r1 \geq r2$, r1 is r2's superior role. roles: $S \rightarrow 2R roles(Si) \subseteq \{r | (\exists r' \geq r)[(user(Si), r') \in UA]\}$, In the hierarchical model, a session contains a subset of the role set (active role), the junior roles of active roles also included in the session.

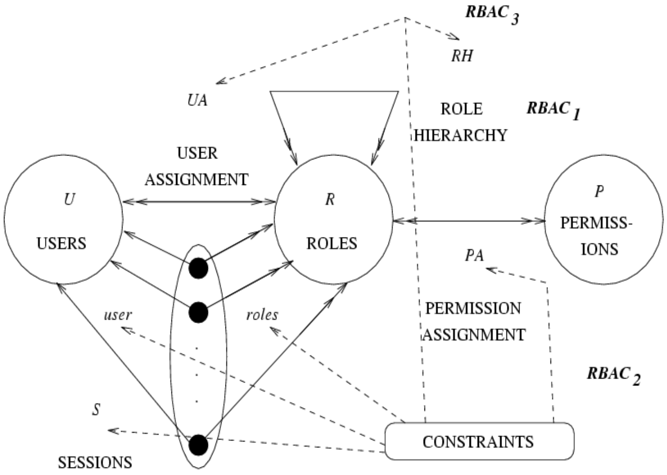

Figure 1 RBAC96 model

RBAC2 is also known as constrained RBAC model, which is introduced based on the RBAC0 constraint (constraint), RBAC constraints is a very important concept, often with the view that the constraints are the main driving force behind one of RBAC [13], Constraints are also often an organization security strategy in the development of a powerful protection mechanism. Constraints can be divided into three categories: separation of duty constraint (SOD), pre-request constraints, cardinality constraints. RBAC3 known as comprehensive RBAC model, which contains RBAC1 and RBAC2, and indirectly includes RBAC0, is a comprehensive role-based access control model. RBAC96 model shown in Figure 1.SOD constraints and relationships described

SOD is one of the characteristics most desired in RBAC system. SOD is very important for an organization, through the implementation of SOD can significantly reduce the incidence of fraud and error. The roles of conflict relationship can not be assigned to a user. The reasons caused all kinds of role conflict related to security policy adopted by system, such as separation of duty or conflict of interest, the principle of least privilege. In NIST Standard RBAC model is based on the principle of SOD, often divided into a static separation of duty SSD (static role conflict) and dynamic separation of duty DSD (dynamic role conflict) by the role conflict time. The former refers to the roles of conflict can not be given a user, this conflict does not depend on the time between changes in change. The latter refers to the roles of conflict in the same session can not be given the same user, that is, conflict relationship will be generated at any given time, and end with the termination of the activities. SSD is also known as the repulsive, the DSD is known as the weak exclusion. As too strict SSD has less chance to implement in real organizations, DSD is relatively more flexible and easy to implementnt.

In 1995, Ferraiolo et al [13] defines three types of SOD: the static separation of duty SSD, , and dynamic separation of duty DSD, separation of duty based on the operation OpSOD. Simon and Zurko [9] increased the variety of types of SOD: obect-based separation of duties ObjSOD and history-based separation of duties HSD. Sandhu et al [13] proposed a session-based SOD and user-based SOD.

A. SOD formal specification

RBAC current research focuses on the description and expressed. One way is by proposing a new language or by using an existing language to express constraints, Such as formal language RSL99 (role-based separation of duty language) [14][15] and its successor languages RCL 2000 (role-based constrains language 2000) [16] for that role-based authorization constraints. Another way is to use a more intuitive graphical to illustrate, Such as the use class diagrams and object diagrams in UML; The researches also use roles to describe conflict and find conflict, and use exclusive collection [17] or Algebra [18] to study the conflict constraints. This paper discusses common conflict constraints using formal specification to describe the role of different constraints on the operation of different models. Formal specification can express these constraints unambiguous, explicit the time of constraints, describe the accuracy of operation to ensure the correct implementation of RBAC. This paper further discusses the practical application of scenarios to verify the system using the formal specification of conflict to the availability of this method. The Formal specification described as follows:

Users, Roles, Ops, Obs, Sessions are finite sets respectively for users, roles, operations, objects and sessions ; The partial order of role is called the dominance relationship. Hierarchical relationships between the roles possess reflexive, transmission and anti-symmetry.

1) Non-strict partial order, or reflexive partial order

Given set S, "≤" is a binary relation on S, if the "≤" is satisfied:

Reflexive: $\forall a \in S$, there is $a \leq a$; Anti-symmetry: $\forall a, b \in S$, $a \leq b$ and $b \leq a$, then $a = b$;

Transitive: $\forall a, b, c \in S$, $a \leq b$ and $b \leq c$, then $a \leq c$;

That, "≤" is Called non-strictly partial order or reflexive partial order on the S.

2) Strict partial order, or anti-reflexive partial order

Given set S, "<" is a binary relation on S, if "<" is satisfied;

Anti-reflexive: $\forall a \in S$, S there is $a \nless a$; Non-symmetry: $\forall a, b \in S$, $a < b \Rightarrow b \nless a$;

Transitive: $\forall a, b, c \in S$, $a < b$ and $b < c$, then $a < c$;

That, "≤" is Called strict partial order or anti-reflexive partial order on S.

Strict partial order and directed acyclic graph (dag) has a direct correspondence. A set of strict partial order on the relationship between the graph is a directed acyclic graph. The transitive closure is its own.

3) Partial order

Easy to prove the following conclusions:

Given set S on a (non-rigorous, reflexive) partial order "≤", S can be naturally induced on a (strict, anti-reflexive) partial order "<", simply defined: $< = \leq \setminus \{(a, a) \mid a \in S\}$;

Given set S on a (strict, anti-reflexive) partial order "<", S can be naturally induced on a (non-rigorous, reflexive) partial order "≤", simply defined: $\leq = < \cup \{(a, a) \mid a \in S\}$;

Given set S on a (non-rigorous, reflexive) partial order "≤", the inverse relation "≥" S is a (non-rigorous, reflexive) partial order on the S;

Given set S on a (non-rigorous, reflexive) partial order "<"and its inverse relationship ">"is also an (non-rigorous, reflexive) partial order on the S.

From the above we can see, as long as the definition one of the "≤ ", "<", " ≥ ", ">", remaining three relations can be defined out naturally. These four relationships can actually be seen as one.

Therefore not strictly distinguish between the cases, only one can be defined (usually "≤"), called partial order on set S. ("partial order" is usually used to refer to non-strict partial order.)

4) UA, PA and others

Role Hierarchy set ⊆ Roles; Op（Permission: Permissions)-> {op ⊆Ops}, after mapping of permissions to operate, there'll be back operations set related to Permission; UA ⊆Users x Roles is a many to many set of distribution relationship between roles and users; PA ⊆ Permission x Role is a many to many set of distribution relationship between permissions and roles; Session_users is a set of user configuration for session.

5) Role-based static separation of duty SSD

Two conflicted roles can not be assigned same role at the same time. Expressed as a formal specification:

$\forall ri, rj \in Roles, i \neq j, \forall u \in Users, (ri, rj) \in SSD$

$\Longrightarrow (u, ri) \notin UA$ or $(u, rj) \notin UA$

6) Role-based dynamic separation of duty DSD

Session_permission is a set of permission for session; Session_UA is a many to many set of distribution relationship between users and roles for session;Session_PA is a many to many set of permission.

Two conflicted roles can not be activated in the same session, expressed as a formal specification:

$\forall ri, rj \in Roles, i \neq j, \forall u \in Users, (ri, rj) \in DSD,$

$(u, ri) \in Session\_UA, (u, rj) \in Session\_UA,$

$\Longrightarrow (u, ri) \notin Session\_UA$ or $(u, rj) \notin Session\_UA$

The set intersected between SSD and DSD is empty. Expressed as a formal specification:

$\forall ri, rj \in Roles, i \neq j, (ri, rj) \in SSD \Longrightarrow (ri, rj) \notin DSD$

7) history-based separation of duties HSD

Two conflicted roles in the history can not be assigned same role at the same time. Expressed as a formal specification:

$\forall ri, rj \in Roles, i \neq j, (ri, rj) \in hsd,$

$\forall u \in Users,$

$ri \in User\_Role\_History,$

$\Longrightarrow (u, rj) \notin ua$

8) Inherited

Inherited the role is still to maintain conflicting relationships, expressed as a formal specification:

$\forall ri, rj, rm, rn \in Roles$, $i \neq j \neq m \neq n$,

$(ri, rj) \in SSD$ $rm \geq rj$, $rn \geq rj$

$\Longrightarrow (rm, rn) \notin SSD$

Conflicting roles can not have a common ancestor roles, expressed as a formal specification:

$\forall ri, rj \in Roles$, $i \neq j$, $(ri, rj) \in SSD$, $rk \geq ri$, $rk \geq rj$ $\Longrightarrow rk \notin Roles$

Conflicting roles can not be inherited each other, expressed as a formal specification:

$\forall ri, rj \in Roles$, $i \neq j$, $(ri, rj) \in SSD$, $rk \not\succ= ri$, $rk \not\succ= rj$

B. SOD constraint relationship and its description

Based on RBAC96 model , the design in this paper will be integrated into HSD, SSD, DSD, event and frequency limit of role and so on.

Separation of duty relations are used to enforce conflict of interest policies. Conflict of interest in a role-based system may arise as a result of a user gaining authorization for permissions associated with conflicting roles. One means of preventing this form of conflict of interest is though static separation of duty(SSD), which enforce constraints on the assignment of users to roles. An example of such a static constraint is the requirement that two roles be mutually exclusive; for example 1, if one role requests expenditures and another approves them, the organization may prohibit the same user from being assigned to both roles. For example 2, a user is assigned the role of accounting, then he may no longer be assigned the role of cashier. Because the accounting and cashier are mutually exclusive. The SSD policy can be centrally specified and then uniformly imposed on specific roles. Because of the potential for inconsistencies with respect to static separation of duty relations and inheritance relations of a role hierarchy, we define SSD requirements both in the presence and absence of role hierarchies.

Static Separation of Duty relations reduce the number of potential permissions that can be made available to a user by placing constraints on the users that can be assigned to a set of roles. Dynamic separation of duty (DSD) relations, like SSD relations, are intended to limit the permissions that are available to a user. However DSD relations differ from SSD relations by the context in which these limitations are imposed. DSD requirements limit the availability of the permissions by placing constraints on the roles that can be activated within or across a user's sessions. DSD allows a user to be authorized for two or more roles that do not create a conflict of interest when acted on independently, but produce policy concerns when activated simultaneously. For example, a user is assigned role A who receives money from customers and role B who supervises how much money to be received in role A's open cash drawer. If the individual acting in the role A attempted to switch to the role B at the same event, DSD would require user to drop the role A before assuming the role B. As long as the same user is not allowed to assume both of these roles at the same time, a conflict of interest situation will not arise.

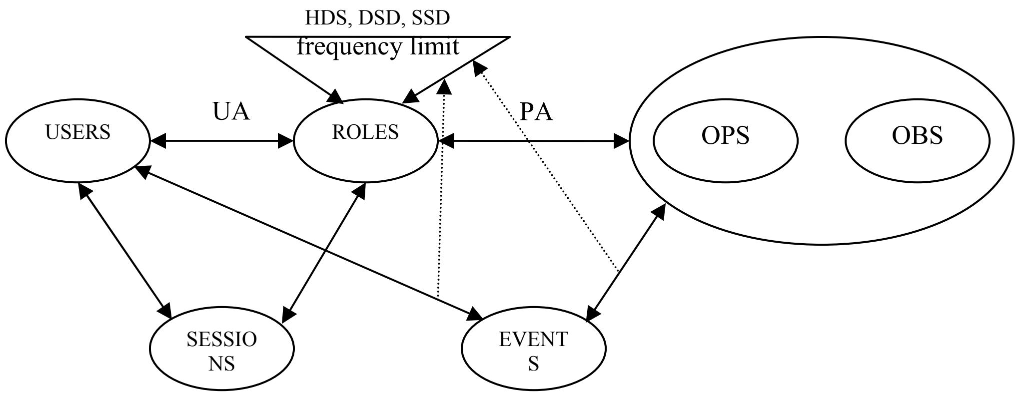

Figure 2 RBAC extended model

SSD provides the capability to address potential conflict of interest issues at the time a user is assigned to a role. DSD allows a user to be authorized for roles that do not cause a conflict of interest when acted on independently, but which produce policy concerns when activated simultaneously. Although this separation of duty requirement could be achieved through the establishment of a static separation of duty relationship, DSD relationships generally provide the enterprise with greater efficiency and operational flexibility.

The conflict could be from user who was assigned a role that is mutually exclusive with the role he is assigned. It is HSD(Historical Separation of Duty). For example, A user who works in the consulting company is assigned a role A that analyzes company A in sale market. He was assigned a role B that analyzed company B in sale market some years ago. But now there is a competitive relationship between company A and company B. So, the user shouldn't be assigned a role A because of HSD. A user can't have a role conflicted with roles in history. Event, that is, the user-role-privilege system is running under events. Such a work, by a representative name as event name, the event have an event_ID. All users associated with the event will work under this event_ID. The events are marked by their respective departments. Only division is responsible for the initiation of events. Events could be very good to enhance operability and flexibility of the whole system. Events are very important to enhance ability of practical application of system.

Frequency limit of role is pre-set time limit and frequency of use for selected role.

The model Included HSD. SSD, DSD and other elements is described in Figure 2.

TABLE 1 SOD constraints attribute

Operation / Object	Static State（Authorizing)		Dynamic State（running)	
	Authorized to an Operation	Authorized Operation<Number of Operation Set	Performed an Operation	Performed Operation<Number of Operation Set
A Object	HSD SSD	HSD SSD	DSD	DSD
Many Objects	HSD SSD	HSD SSD	DSD	DSD

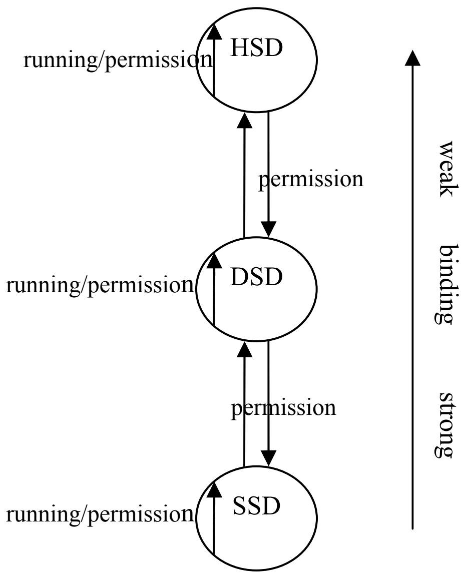

Figure 3 SOD constraint relationship

According to three key factors: operation, object and static /dynamic state in separation of duties constraints, attribute table of separation of duties is listed to describe the relationship between properties of separation of duties, as shown in Table 1.

The relationship between SOD constraints is described in Figure 3.

The strongest constraint is in the state diagram at the bottom, with the arrow up, followed by reduced binding. While implementing the strategy of SOD, the weak constraints of SOD is more flexible and more easy to implement separation of duties, the strong constraints is more difficult to achieve due to too strict.

DSD constraint is mainly carried out in the event. For example, an usre need enter an event, such as the nnumber of roles that user is assigned is greater than or equal two, the DSD constraint will be drived, if the role of user is in DSD, the role is a constraint role and then user can't be enter

When user is assigned a role, SSD constraint'll be judged. In case of its own role and selected role in the SSD, the role is a constraint role and role should be assigned selected role.

III. A PROGRAM FOR IMPLEMENTING SOD CONSTRAINT RELATIONSHIP

The system's main functions are the following:

- Make users associated with the roles and roles associated with permissions.
- Make restriction constrains of Static Separation of Duty (SSD) and Historical Separation of Duty (HSD) while establishing the relationship between users and roles.
- Make restriction constrains of Time and Frequency while establishing the relationship between users and roles.
- Make restriction constraints of Dynamic Separation of Duty(DSD) after user is assigned roles and while the event is entered.
- Set users, roles and permissions.
- Configure binary relation to SSD, HSD and DSD.
- Maintain system data.

The design is divided into two parts: the system administrator management and user management. The system administrator management includes how to manage information set, user assignment, role assignment, permission assignment, and how to configure binary relation for SSD and HSD, how to maintain system data. User management mainly starts configuring binary relation for DSD when event happened. The case diagram is shown in Figure 4.

Taking into account role-based access control, system adopts the following tables associated with the control competence. The relationship of tables is shown in Figure 5.

Visitors are divided into two normal users and administrator. They first enter their ID. If ID is correct and is normal users, they obtain the corresponding user roles and then get operational permission of various functional modules, and enter the corresponding subsystems operate; If ID is correct and is administrator, they obtain the authority to manage user assignment, role assignment, permission assignment and so on. If ID is not correct, they are refused.

The following shows SQL statement of the main database. They are UA, PA, SSD, DSD, HSD, Session and event.

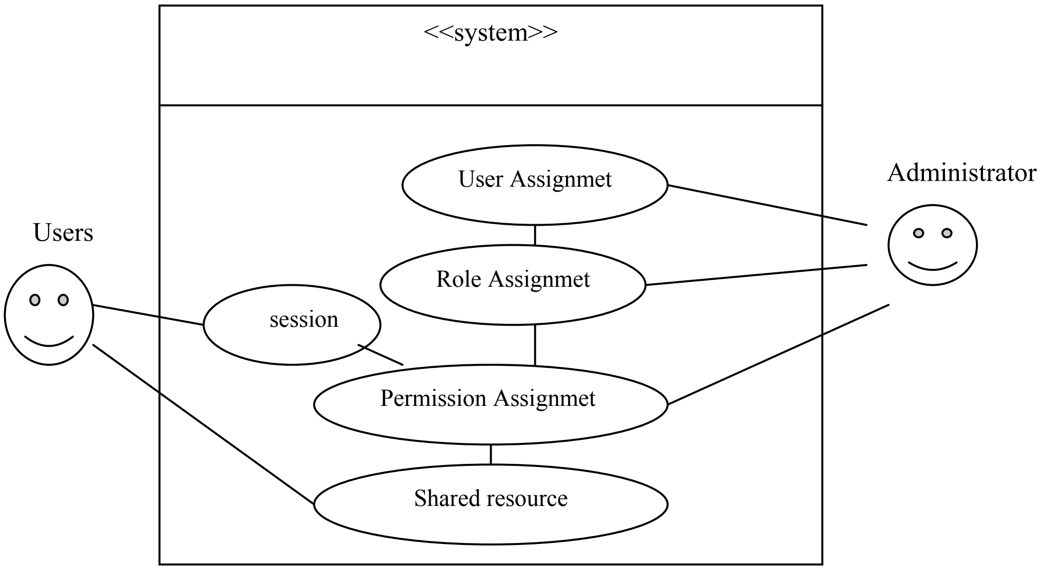

Figure 4 Case diagram

```
CREATE TABLE `ua` (
  `User_ID` varchar(50) NOT NULL,
  `Role_ID` varchar(50) NOT NULL,
  `Description` varchar(50) default NULL,
  `Role_Time` varchar(50) default '-',
  `Role_Frequency_Limit` varchar(50) default '-',
  `Role_Frequency_Now` varchar(50) default '-',
  PRIMARY KEY  (`User_ID`, `Role_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `pa` (
  `Role_ID` varchar(50) NOT NULL,
  `Permission_ID` varchar(50) NOT NULL,
  `Description` varchar(120) default NULL,
  PRIMARY KEY  (`Role_ID`, `Permission_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `ssd` (
  `Role_ID1` varchar(50) NOT NULL default '',
  `Role_ID2` varchar(50) NOT NULL default '',
  `Description` varchar(120) default NULL,
  PRIMARY KEY  (`Role_ID1`, `Role_ID2`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;


CREATE TABLE `dsd` (
  `Role_ID1` varchar(50) NOT NULL default '',
  `Role_ID2` varchar(50) NOT NULL default '',
  `Description` varchar(120) default NULL,
  PRIMARY KEY  (`Role_ID1`, `Role_ID2`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `hsd` (
  `Role_ID1` varchar(50) NOT NULL default '',
  `Role_ID2` varchar(50) NOT NULL default '',
  `Description` varchar(120) default NULL,
  PRIMARY KEY  (`Role_ID1`, `Role_ID2`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `sessions` (
  `Session_Total_ID` int(10) NOT NULL default '0',
  `Session_Event_ID` varchar(20) default '0',
  `Session_Users_ID` varchar(50) default NULL,
  `Session_Roles_ID` varchar(50) default NULL,
  `Session_Permission_ID` varchar(50) default NULL,
  `Session_Time` varchar(50) default NULL,
  PRIMARY KEY  (`Session_Total_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `event` (
  `Event_ID` varchar(50) NOT NULL,
  `Event_Bool` varchar(20) default NULL,
  `Event_User_ID` varchar(50) default NULL,
  `Department_ID` varchar(50) default NULL,
  PRIMARY KEY  (`Event_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

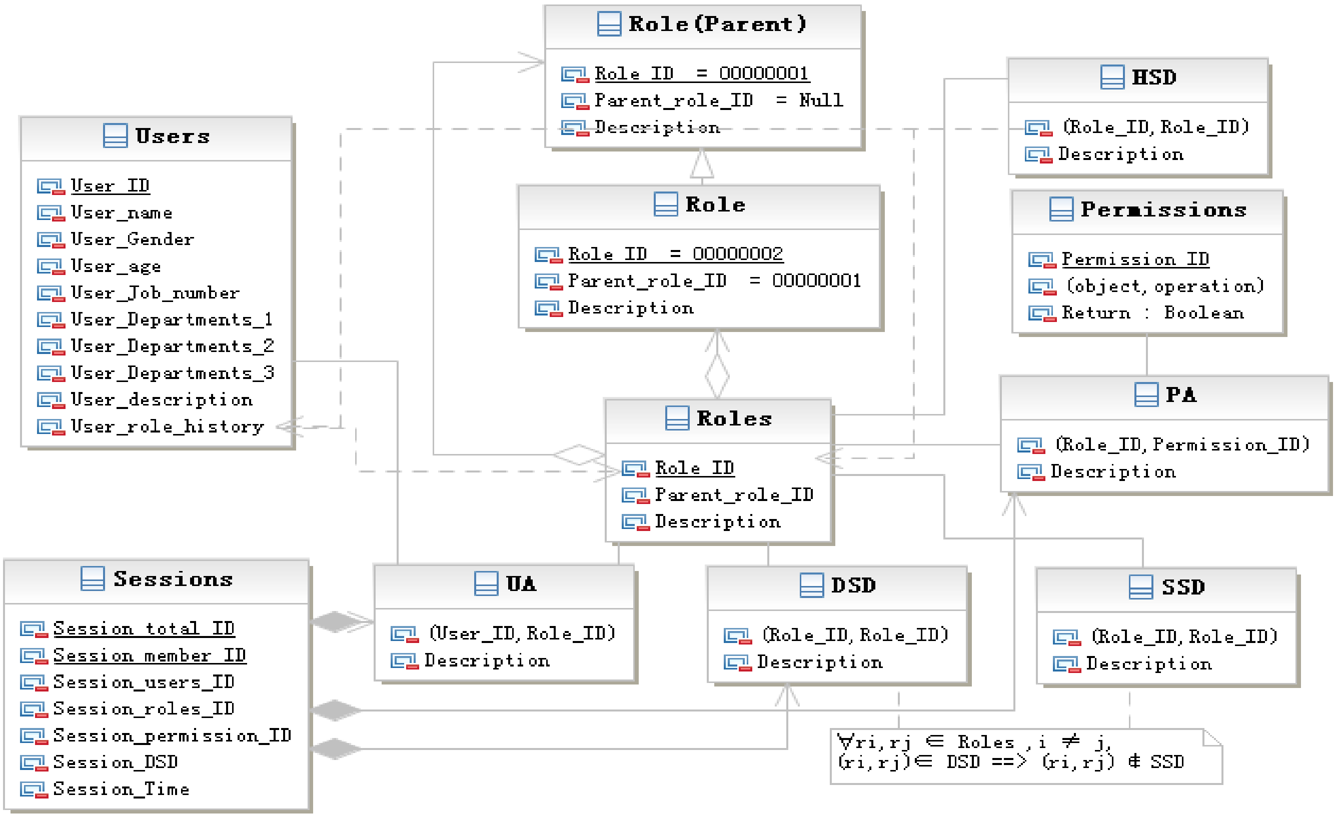

Figure 5 The relationship between tables

IV. Conclusion

Role-based access control is a flexible, easy-to-manage, low-cost access control method. It is a good reflection of function and structure of actual organization. RBAC is used widely as a access control method.

SOD is very important in business system, also is one of the most desired features in RBAC system. The system can achieve the desired security policy and security objectives. This paper describes basic model RBAC96 and properties of SOD constraint with formal specification, presents an execution program to achieve SOD constraint with relationship of tables, including time and frequency on

restrictions. Some SQL statements are given to explain what's relationship between SOD constraints.

This paper analyzes the properties of various separation of duty constraint, but it does not consider the objects and other details, which need to research in the future.

Acknowledgment

Here special thanks to the two foundations for funding the project. Research and Innovation Projects Of Shanghai Municipal Education Commission, project number is 11YZ212 (The writer is the head of the project); Technology Development Fund of Shanghai University of Engineering of Science , project number is 2008xy20 (The writer is the head of the project).

References

[1] David Farraiolo and Richard Kuhn, "Role-Based Access Control, " 15th NIST-NCSC National Computer Security Conference, 1992.

[2] Ravi Sandhu, "Lattice Based Access Control Models, " IEEE Computer, 26: 11, November 1993.

[3] David D. Clark and David R. Wilson, "A Comparison of Commercial and Military Computer Security Policies, " Proceedings of the 1987 IEEE Symposium on Research in Security and Privacy (SP'87), May 1987

[4] Michael J. Nash and Keith R. Poland, "Some Conundrums Concerning Separation of Duty", IEEE Symposium on Research in Security and Privacy, May 1990.

[5] Jerome H. Saltzer and Michael D. Schroeder, "The Protection of Information in Computer Systems, " Communications of the ACM, 17: 7, 1975.

[6] Ravi Sandhu et al. "Role Based Access-Control Models", IEEE Computer, February 1996.

[7] Ravi Sandhu, "Rationale for the RBAC96 Family of Access Control Models." Proceedings of the first ACM Workshop on Role-based access control", February 1996.

[8] David Ferraiolo et al., "Proposed NIST Standard for Role-Based Access Control, " ACM Transactions on Information and System Security, 4: 3, August 2001.

[9] Simon, R. and Zurko, M. E. 1997. Separation of duty in role based access control environments [C]. In Proceedings of the 10th IEEE Workshop on Computer Security Foundations (Rockport, MA, June 10-12). IEEE Computer Society Press, Los Alamitos, CA, 183–194

[10] Virgil D. Gligor, Serban I. Gavrila, and David Ferraiolo. On the formal definition of separation-of-duty policies and their composition [C]. In Proceedings of IEEE Symposium

on Research in Security and Privacy, pages 172-183, Oakland, CA, May 1998.
[11] Gail-Joon Ahn & Ravi Sandhu, "Role-based authorization constraints specification, " J. ACM Transactions on Information and System Security., 2000
[12] Chunyang Yuan et al., "A Verifiable Formal Specification for RBAC, " Lecture Notes in Computer Science, 2006, Volume 4318/2006,
[13] Ferraiolo, D., Cugini, J., Kuhn, D. R. "Role-Based Access Control (RBAC): Features and Motivations" [C]. Proc. 1995 Computer Security Applications Conference, 241-248, December 1995.
[14] Ahn, G. -J. AND Sandhu, R. 1999. The RSL99 language for role-based separation of duty constraints [C]. In Proceedings of 4th ACM Workshop on Role-Based Access Control (RBAC '99, Fairfax, VA, Oct. 28-29). ACM, New York, NY, 43–54.
[15] G.J. Ahn, R. Sandhu. The RSL99 language for role-based separation of duty constraints. ACM Workshop on Role-Based Acces Control, Fairfax, Virginia, USA, 1999
[16] G.J. Ahn, R. Sandhu. Role-based authorization constraints specification. ACM Trans on Information and System Security, 2000, 3(4) : 207-226
[17] D. R. Kuhn. Mutual exclusion of roles as a means of implementing separation of duty in role-based access control system. The 2nd ACM Workshop on Role-Based Access Control, Fairfax, VA, 1977
[18] N H Li, Q H Wang, M V Tripumitara. Beyond separation of duty: An algebra for specifying high-level security policies. Purdue University, CERIAS, Tech Rep: 2005-75, 2005
[19] Joon S.Park et al., "Role-based access control on the web s, " J. ACM Transactions on Information and System Security, April 2001.
[20] Jean Bacon et al., "A model of OASIS role-based access control and its support for active security , ", J. ACM Transactions on Information and System Security., April 2002
[21] Jason Crampton, "Delegation in role-based access control , " J. International Journal of Information Security, 2007.
[22] Zhang Zhiyong, "Collaboration Access Control Model for MAS Based on Role and Agent Cooperative Scenarios, " J. IEEE International Conference on Mechatronics and Automation, 2006

Su Yu, Female , was born in China, graduated from Lawrence Technological University in 2007 for master's degree in Computer Science. Lawrence Technological University is located at 21000 West Ten Mile Road, Southfield, MI 48075-1058, USA.

She works in Engineering Training Center, Shanghai University of Engineering Science as a deputy director and associate professor. Shanghai University of Engineering Science is located at Long Teng Lu 333, Shanghai 201620, China. She engages in teaching and research in computer security, computer networks and database applications.

Ms. Yu is a member of committee of computer textbook of undergraduate. She is responsible for a project "Research and Application of information system security in RBAC" from foundation of Shanghai Municipal Education Commission now.