사용자-역할 할당을 위한 URA99 모델의 구현

박동규, 황유동°, 안현수
순천향대학교 정보기술공학부

An Implementation of the URA99 Model for User-Role Assignment

Dong-Gue Park, Yu-Dong Hwang°, Hyun-Su An
Dept. of Information and Technology Engineering, Soonchunhyang University

요 약

역할기반 접근제어(RBAC)는 역할(Role)과 역할계층(Role hierarchy)을 통해 사용자 및 접근권한 관리를 효율적으로 수행할 수 있도록 해준다. 그러나 시스템에 수많은 사용자, 역할, 권한이 존재하는 경우 한사람의 보안 관리자가 이들을 모두 관리하는 것은 불가능하므로 역할을 관리하는 관리역할을 두어 시스템을 효율적으로 관리할 수 있는 방법(ARBAC)이 제안되었다. ARBAC는 URA(User Role Assignment), PRA(Permission Role Assignment), RRA(Role Role Assignment)로 구성되어있다. 본 논문에서는 URA99 모델을 기반으로 사용자-역할 관리를 위하여 관리도구를 구현한다. 구현된 관리도구는 오라클의 저장 프로시저를 사용하고 자바를 기반으로 한 EJB 컴포넌트로 구현한다.

1. 서론

역할기반 접근통제(RBAC, Role-Based Access Control)의 개념은 접근통제의 전통적인 강제적 접근통제(MAC, Mandatory Access Control) 및 임의적 접근통제(DAC, Discretionary Access Control)의 대안으로서 많은 관심을 집중시키고 있다.[1,2,3,4]

역할기반 접근통제는 역할(role), 역할과 관련된 행동을 나타내는 허가(permission), 사용자(user)의 관계로 표현된다.

역할기반 접근통제는 관리자에게 편리한 관리 능력을 제공하여 관리업무의 효율성을 꾀할 수 있고, 역할, 역할계층(Role hierarchy), 관계(relationship), 제약(constraint)의 정립을 통하여 사용자의 행동을 정적 또는 동적으로 규제할 수 있으므로 시스템 관리자에게 객체단위가 아닌 추상적인 개념으로 접근을 통제할 수 있어서 실제 환경에 자연스럽게 접목, 구현될 수 있는 장점과 분산환경에서 사용되는 경우 역할기반 접근통제 관리자의 책임을 중앙과 국지 보호 영역으로 구분할 수 있어서 관리 책임을 분명히 할 수 있다는 장점이 있다.

그러나, 시스템에 수많은 사용자, 역할, 권한이 존재하는 경우 한 사람의 보안 관리자가 이들을 모두 관리하는 것은 불가능하므로 Ravi S. Sandhu가 역할을 관리하는 관리역할(Administrative Role)을 두어 시스템을 효율적으로 관리할 수 있는 ARBAC(Administrative Role-Based Access Control)를 제안하였다.[1,2,3,4]

본 논문에서는 ARBAC97의 URA97과 ARBAC99의 URA99에 대해 살펴보고, URA99를 기반으로 사용자-역할 관리를 위하여 관리도구를 구현한다. 구현된 관리도구는 오라클의 저장 프로시저(stored procedure)를 사용하고 자바를 기반으로 한 EJB 컴포넌트로 구현한다.

본 연구는 정보통신부의 ITRC 사업에 의해 수행된 것 임

2. URA97 Administrative Model

URA97 모델은 사용자에게 역할을 허가하는 모델과 사용자의 자격을 취소하는 모델로 이루어져 있다. 아래의 그림 1은 역할계층과 관리역할계층의 일부를 예로 보여준다. 그림 1에서 a)는 개발 부서의 역할계층을 예로 들었으며, b)는 그 역할을 관리하는 관리 역

할계층이다. 그림 1, a)에서 모든 사원은 최하위 역할인 E에 속하며, 하위 역할인 ED부터 상위 역할인 DIR사이의 역할이 개발 부서의 역할이다. b)의 관리 역할계층은 최상위에 보안관리자(SSO)역할을 두고 두 개의 프로젝트 보안관리자(PSO1, PSO2)역할이 있으며, 두 역할사이에 부서 보안관리자(DSO)역할이 있다. URA97에서는 can-assign 관계를 이용하여 사용자-역할 할당을 제어하고 can-revoke 관계를 이용하여 사용자-역할 취소(revocation)를 한다.[1,4]

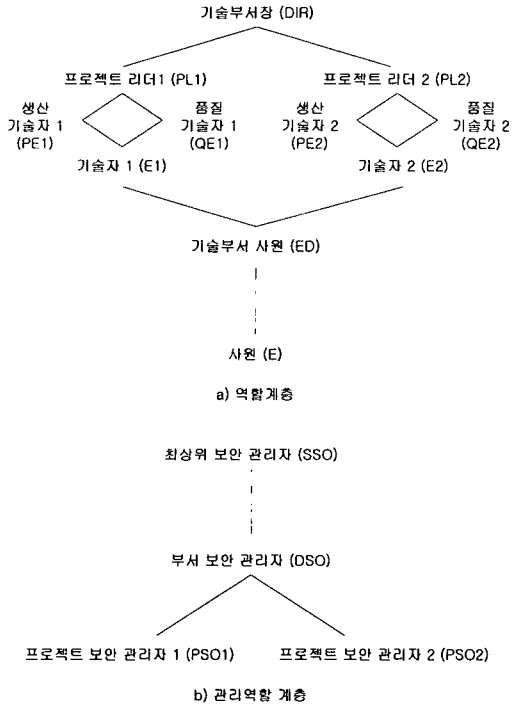

그림 2 역할계층과 관리역할계층의 예

URA97에서 사용자-역할 취소는 약한취소(weak revocation)와 강한취소(strong revocation)가 있다. 약한취소는 E1, PE1, PL1의 역할을 가진 사용자 u에서 관리자가 E1의 역할을 취소해도 사용자는 계속 PE1과 PL1의 역할을 가지는 것을 의미한다. 강한 취소는 E1의 역할을 취소할 경우 E1을 포함하고 있는 사용자의 모든 역할이 취소되는 것을 의미한다. URA97에서 implict membership은 사용자에게 상위 역할이 할당되면 할당된 역할보다 하위역할의 권한을 가질 수 있음을 의미한다. explicit membership은 할당된 역할의 권한을 가질 수 있음을 의미한다.

3. URA99 Administrative Model

URA99에서는 역할을 이동자격(mobile membership)과 부동자격(immobile membership)으로 분류한다. 역할에 대해 이동자격이 주어진 사용자는 관리역할이 다른 역할을 사용자에게 매핑 시킬 때 다른 역할로 이동시킬 수 있음을 의미하고, 역할에 대해 부동 자격이 주어진 사용자는 관리 역할이 다른 역할을 사용자에게 매핑 시킬 때 다른 역할로 이동시킬 수 없음을 의미한다. URA99에서 can-assign 관계를 이용하여 사용자-역할 할당의 예를 들면 다음 표 1, 2와 같다.

표 1 can-assign-M의 예

관리 역할	전제 역할	역할 범위
PSO1	ED	[E1, PL1)
PSO2	ED	[E2, PL2)
DSO	ED ∧ $\overline{PL2}$	[PL1, PL1]
DSO	ED ∧ $\overline{PL1}$	[PL2, PL2]
SSO	ED	(ED, DIR]
SSO	E	[ED, ED]

표 2 can-assign-IM의 예

관리 역할	전제 역할	역할 범위
PSO1	ED	[E1, PL1)
PSO2	ED	[E2, PL2)
DSO	ED ∧ $\overline{PL2}$	[PL1, PL1]
DSO	ED ∧ $\overline{PL1}$	[PL2, PL2]
SSO	ED	(ED, DIR]
SSO	E	[ED, ED]
DSO	E	[ED, ED]

표2의 여섯 번째 줄까지는 표 1과 같다. 여기서 이동자격이나 부동자격은 관리자가 선택함을 알 수 있다. 표2의 마지막 줄 DSO는 사원(E)를 ED의 부동자격으로 등록하고 이동자격으로 등록할 수 없다. URA99에서 can-revoke 관계를 이용하면 can-revoke-M, can-revoke-IM도 위 그림 1, 2와 같은 형식의 사용자-역할 취소 표를 구할 수 있다.

4. URA99 구현

URA99를 구현하기 위하여 표 1, 2를 이용하여 아래 그림 2와 같은 ER 다이어그램을 작성하였다. 그림 2는 can-assign-M과 can-assign-IM의 경우이며 can- revoke-M과 can-revoke-IM도 그림 2와 같은 형식의 ER 다이어그램을 작성할 수 있다.

다음의 표 3, 4, 5, 6, 7은 그림 2의 ER 다이어그램을

이용하여 테이블을 작성하고 작성된 테이블에 관리역할 PSO1에 대한 데이터를 입력한 예이다.

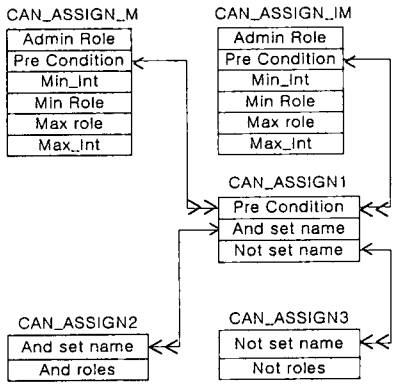

그림 3 can-assign-M, can-assign-IM 관계의 ER 다이어그램

표 3 can_assign_M 테이블의 예

AR	PC	Min_Int	Min_R	Max_R	Max_Int
PSO1	C1	[	E1	E1	]
PSO1	C2	[	PE1	PE1	]
PSO1	C3	[	QE1	QE1	]
...	...	...	...	...	...

표 4 can_assign_IM 테이블의 예

AR	PC	Min_Int	Min_R	Max_R	Max_Int
PSO1	C1	[	E1	E1	]
PSO1	C2	[	PE1	PE1	]
PSO1	C3	[	QE1	QE1	]
...	...	...	...	...	...

표 5 can_assign1 테이블의 예

PC	and_set_name	not_set_name
C1	ASET1	null
C2	ASET2	NSET2
C3	ASET3	NSET2
...	...	...

표 6 can_assign2 테이블의 예

and_set_name	and_roles
ASET1	ED
ASET2	ED
ASET3	ED
...	...

표7 can_assign3 테이블의 예

not_set_name	not_roles
NSET1	QE1
NSET2	PE1
...	...

본 논문에서는 URA99를 구현하기 위해 위의 역할관계 테이블을 이용하고, Oracle8i에서 SQL을 확장한 PL/SQL을 사용하여 사용자-역할 할당을 위한 관리도구를 저장 프로시저로 작성하였다. 작성한 저장 프로시저는 다음과 같다.

- ASSIGN(user, trole, arole, mobile)
- WEAK_REVOKE(user, trole, arole, mobile)
- STRONG_REVOKE(user, trole, arole, mobile)

저장 프로시저는 사용자에게 역할을 할당하고 역할을 취소하는 기능을 한다.

모든 프로시저에 공통으로 입력되는 파라미터 user와 역할 trole(target role)은 user에게 역할 trole을 할당하거나 user의 역할 trole을 취소한다. arole(administrative role) 파라미터는 사용자-역할 할당과 취소를 관리하기 위해 적용하는 관리역할이다. mobile 파라미터는 trole 파라미터에 입력되는 역할이 이동자격을 가지는지 부동자격을 가지는지를 표시한다. mobile 파라미터에 입력된 값이 이동자격을 표시하면 저장 프로시저는 can_assign_M 테이블에서 관리역할과 역할의 범위를 검색하고 부동자격을 표시하면 저장 프로시저는 can-assign_IM 테이블에서 관리역할과 역할의 범위를 검색한다.

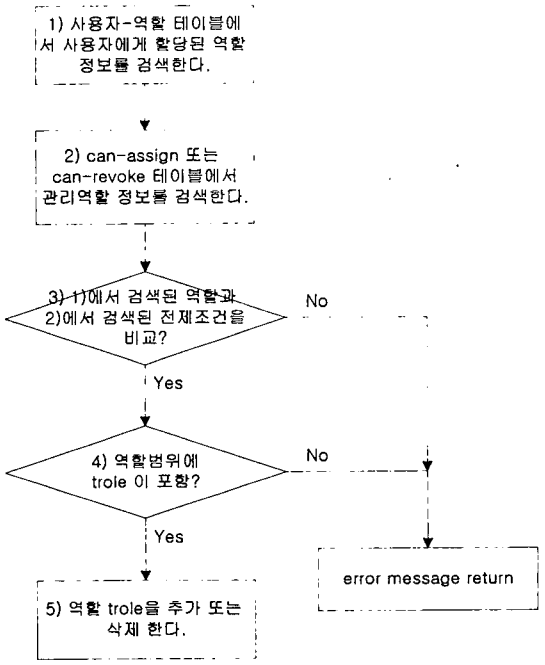

그림 4 저장 프로시저 동작 단계

저장 프로시저가 동작되는 단계를 설명하면 위 그림 3과 같다. 다음 그림 4는 ASSIGN 프로시저에서 역할

의 범위에 할당하고자 하는 역할이 포함되는지 확인하는 저장 프로시저 구문의 일부이다.

```
CREATE PROCEDURE ASSIGN ( USER IN CHAR(20), TROLE IN CHAR(10),
                          AROLE IN CHAR(10), MOBILE IN CHAR(1), RET OUT CHAR(50) ) IS
BEGIN
                                    ...
    IF (TROLE_CURSOR.TINT_MIN = '(') AND (TROLE > TROLE_CURSOR.TROLE_MIN) AND
       (TROLE_CURSOR.TINT_MAX = ')') AND (TROLE < TROLE_CURSOR.TROLE_MIN)) THEN
       RET := 'assign';
    END IF;
                                    ...
END ASSIGN;
```

그림 5 저장 프로시저 ASSIGN의 예

프로시저에서는 테이블 내에서 관리역할이 중복되어 존재하므로 프로시저 내에서 커서와 FOR LOOP을 사용하여 처리하였다.

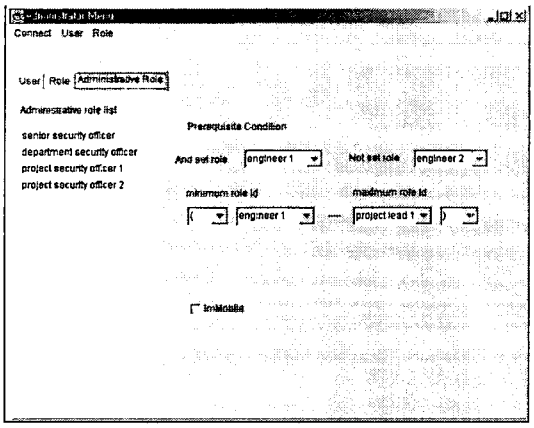

그림 6 관리도구에서 관리역할을 정의하는 화면

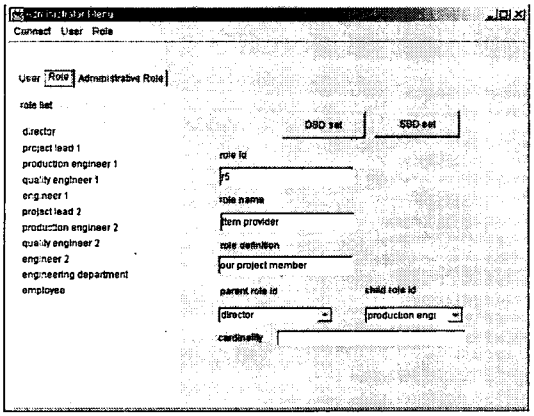

그림 7 관리도구에서 역할을 정의하는 화면

위 그림 5, 6과 7은 저장 프로시저를 호출해서 사용할 수 있는 관리도구의 GUI이다. 관리도구는 인프라이스사의 JBuilder4와 IAS4.5를 이용하여 EJB 컴포넌트로 구현하였으며 데이터베이스는 Oracle8i를 사용하였다.

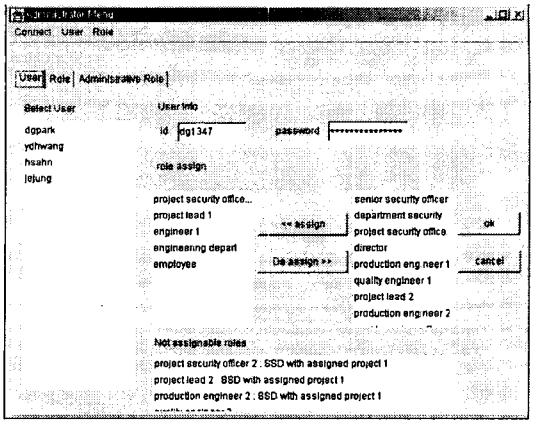

그림 8 관리도구에서 사용자-역할 할당을 관리하는 화면

5. 결론

시스템에 수많은 사용자, 역할, 권한이 존재하는 경우 한사람의 보안 관리자가 이들을 모두 관리하는 것은 불가능하므로 역할을 관리하는 관리역할을 두어 시스템을 효율적으로 관리할 수 있는 방법(ARBAC)이 필요하다.

본 논문에서는 ARBAC99에서 URA99를 기반으로 사용자-역할 관리를 위하여 관리도구를 구현하였다. 구현된 관리도구는 오라클의 저장 프로시저를 사용하고 자바를 기반으로 한 EJB 컴포넌트로 구현한다. 구현된 URA99를 기업환경의 시스템에 적용해 보고 향후 PRA99와 RRA99를 추가 구현하여 ARBAC99를 기업환경의 시스템에 적용해 보고자 한다.

[참고문헌]

[1] Ravi Sandhu, Venkata Bhamidipati, "Role-Based Administration of User-Role Assignment : The URA97 Model and its Oracle Implementation", Journal of Computer Security, Volume 7,1999

[2] Ravi Sandhu , Qamar Munawer, "The ARBAC99 Model for Administration of Roles", ACSAC, 1999

[3] Ravi Sandhu , Venkata Bhamidipadi, "An Oracle Implementation of the PRA97 Model for Permission-Role Assignment ", ACM RBAC, 1998

[4] Ravi Sandhu, Joon S. Park, "Decentralized User-Role Assignment for Web-based Intranets ", ACM RBAC, 1998