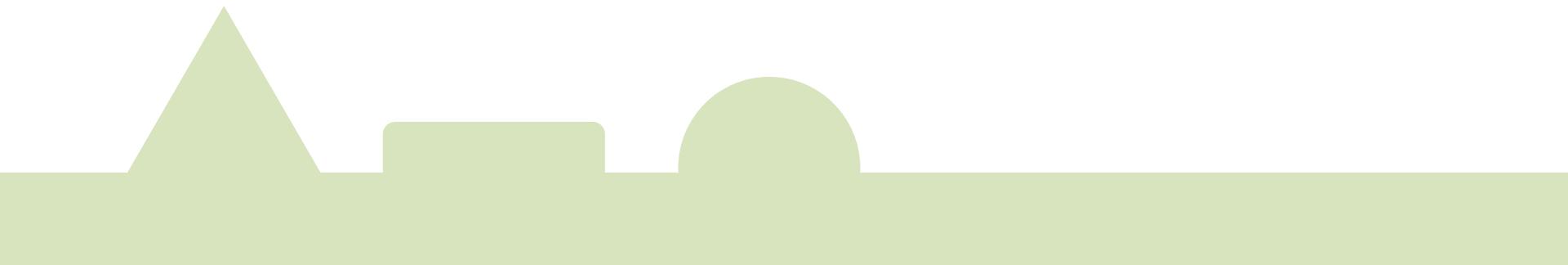


ORACLE®

17. 룰



목차

1. 롤이란
2. 롤의 종류
3. 사용자 롤 정의
4. 롤 회수하기
5. 롤의 장점

1. 롤이란(1)

- ❖ 롤은 사용자에게 보다 효율적으로 권한을 부여할 수 있도록 여러 개의 권한을 묶어 놓은 것이라고 생각하면 된다.
- ❖ 사용자를 생성했으면 그 사용자에게 각종 권한을 부여해야만 생성된 사용자가 데이터베이스를 사용할 수 있다.
- ❖ 데이터베이스의 접속 권한(CREATE SESSION), 테이블 생성 권한(CREATE TABLE), 테이블 수정(UPDATE), 삭제(DELETE), 조회(SELECT) 등과 같은 권한은 사용자에게 기본적으로 필요한 권한들인데 사용자를 생성할 때마다 일일이 이러한 권한을 부여하는 것은 번거롭다.
- ❖ 이 때문에 다수의 사용자에게 공통적으로 필요한 권한들을 룰에 하나의 그룹으로 묶어두고 사용자에게는 특정 룰에 대한 권한 부여를 함으로서 간단하게 권한 부여를 할 수 있도록 한다.

1. 롤이란(2)

- ❖ 또한 여러 사용자에게 부여된 권한을 수정하고 싶을 때에도 일일이 사용자마다 권한을 수정하지 않고 룰만 수정하면 그 룰에 대한 권한 부여를 한 사용자들의 권리가 자동 수정된다. 이 밖에 룰을 활성화 비활성화 함으로서 일시적으로 권리 부여했다 철회할 수 있으므로 사용자 관리를 간편하고 효율적으로 할 수 있다.

2. 롤의 종류[1]

- ❖ 롤은 오라클 데이터베이스를 설치하면 기본적으로 제공되는 사전 정의된 롤과 사용자가 정의한 롤로 구분된다.
- ❖ 사용자가 직접 롤을 정의하는 방법은 복잡하므로 사전에 정의된 롤부터 살펴보자.

2. 롤의 종류[2]

❖ CONNECT 률

- 사용자가 데이터베이스에 접속 가능하도록 하기 위해서 다음과 같이 **가장 기본적인 시스템 권한** 8가지를 묶어 놓았다.

예

**ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK,
CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM,
CREATE TABLE, CREATE VIEW**

❖ RESOURCE 률

- 사용자가 **객체(테이블, 뷰, 인덱스)**를 생성할 수 있도록 하기 위해서 시스템 권한을 묶어 놓았다.

예

**CREATE CLUSTER, CREATE PROCEDURE, CREATE SEQUENCE,
CREATE TABLE, CREATE TRIGGER**

❖ DBA 률

- 사용자들이 소유한 데이터베이스 객체를 관리하고 사용자들을 작성하고 변경하고 제거할 수 있도록 하는 모든 권한을 가진다. 즉, 시스템 자원을 무제한적으로 사용하며 시스템 관리에 필요한 모든 권한을 부여할 수 있는 강력한 권한을 보유한 률이다.

2.2 롤 관련 데이터 딕셔너리

- ❖ 다음은 사용자에게 부여된 를을 확인해 보겠다. 를을 확인하기 위한 데이터 딕셔너리는 무수히 많다.

형식

SELECT * FROM DICT WHERE TABLE_NAME LIKE '%ROLE%';

- ❖ 위 조회 결과로 얻어진 데이터 딕셔너리를 통해서 부여된 권한에 대한 정보를 확인할 수 있다. 다음은 **롤 관련 데이터 딕셔너리**를 정리한 표이다.

딕셔너리 명	설 명
ORLE_SYS_PRIVS	롤에 부여된 시스템 권한 정보
ROLE_TAB_PRIVS	롤에 부여된 테이블 관련 권한 정보
USER_ROLE_PRIVS	접근 가능한 를 정보
USER_TAB_PRIVS_MADE	해당 사용자 소유의 오브젝트에 대한 오브젝트 권한 정보
USER_TAB_PRIVS_REC'D	사용자에게 부여된 오브젝트 권한 정보
USER_COL_PRIVS_MADE	사용자 소유의 오브젝트 중 칼럼에 부여된 오브젝트 권한 정보
USER_COL_PRIVS_REC'D	사용자에게 부여된 특정 칼럼에 대한 오브젝트 권한 정보

2.3 룰 확인하기

- ❖ 룰 관련 데이터 딕셔너리 중에서 현재 사용자에게 부여된 룰을 확인하기 위한 데이터 딕셔너리는 USER_ROLE_PRIVS 이다. USER04로 로그인 하였으므로 다음과 같이 입력하면 사용자 USER04에 부여된 룰에 대한 정보를 확인할 수 있다.

예

```
CONN USER04/TIGER  
SELECT * FROM USER_ROLE_PRIVS;
```

USERNAME	GRANTED_ROLE	ADMIN_OPTION	DEFAULT_ROLE	OS_GRANTED
USER04	CONNECT	NO	YES	NO
USER04	RESOURCE	NO	YES	NO

3. 사용자 롤 정의(1)

- ❖ CONNECT, RESOURCE 룰과 같은 기본적으로 제공되는 사전 정의된 를을 사용자에게 부여해 보았다.
- ❖ 이번에는 사용자가 정의해서 사용하는 률에 대해 살펴보겠다. 사용자는 CREATE ROLE 명령어로 다음 형식에 따라 륄을 생성해야한다.

형식

CREATE ROLE role_name;
GRANT privilege_name TO role_name;

3. 사용자 룰 정의(2)

- ❖ 이번에는 사용자 USER04에 객체 권한을 직접 부여하지 않고 룰을 이용해 보도록 하겠다.
- ❖ 다음과 같은 순서로 작업을 진행할 것이다.

① 룰을 생성한다.(데이터베이스 관리자)

② 룰에 권한 부여한다.
(데이터베이스 관리자 혹은 특정 사용자)

③ 사용자에 룰을 부여한다.(데이터베이스 관리자)

3. 사용자 룰 정의(3)

- ❖ ① 룰을 생성하기 위한 작업은 DBA에서 이루어진다.
 - `CREATE ROLE ROLE_NAME;`
- ❖ ② 룰에 부여할 권한의 종류에 따라서 DBA에서 부여할 수도 있고, 객체를 소유한 사용자로 접속한 후 부여해야 한다.
- ❖ 다음과 같이 시스템 권한 일 경우에는 DBA에서 이루어진다.
 - `GRANT CREATE SESSION, CREATE TABLE, CREATE VIEW TO ROLE_NAME;`
- ❖ 다음과 같이 객체 권한일 경우에는 특정 객체로 접근해서 부여해야 한다
 - `GRANT OBJECT_PRIV TO ROLE_NAME;`
- ❖ ③ 사용자에 룰을 부여하는 작업 역시 DBA에서 이루어진다.
 - `GRANT ROLE_NAME TO USER_NAME;`

3.1 룰을 생성하여 시스템 권한 할당하기

- ❖ 대략적인 순서는 위와 같다. 룰을 생성하여 할당하는 본격적인 실습을 하도록 하자.
- ❖ 1. 룰을 생성할 수 있는 사용자는 반드시 DBA 권한이 있는 사용자여야만 하기에 룰을 생성하기 앞서서 반드시 DBA 권한을 가진 사용자로 접속해야만 한다.

예

```
CONN SYSTEM/PASSWORD  
CREATE ROLE MROLE;
```

- ❖ 2. 생성된 룰에게 권한을 부여한다.

예

```
GRANT CREATE SESSION, CREATE TABLE, CREATE VIEW TO  
MROLE;
```

- ❖ 3. 사용자를 생성하여 룰을 부여한다.

예

```
CREATE USER USER05 IDENTIFIED BY TIGER;  
GRANT MROLE TO USER05;
```

3.1 룰을 생성하여 객체 권한 할당하기(1)

- ❖ 룰을 생성하여 할당하는 객체 권한을 할당해보자.
- ❖ 1. 룰을 생성할 수 있는 사용자는 반드시 DBA 권한이 있는 사용자여야만 하기에 룰을 생성하기 앞서서 반드시 DBA 권한을 가진 사용자로 접속해야만 합니다. CREATE ROLE 명령문을 사용하여 MROLE02 를 생성한다.

예

```
CONN system/password  
CREATE ROLE MROLE02;
```

- ❖ 2. 생성한 룰에 권한 부여를 하기 위해서 EMP 테이블 객체를 소유하고 있는 scott 사용자로 로그인 한다. MROLE02 에게 EMP 테이블 객체를 조회할 수 있도록 SELECT 권한을 준다.

예

```
CONN scott/tiger  
GRANT SELECT ON EMP TO MROLE02;
```

3.1 룰을 생성하여 객체 권한 할당하기(2)

- 3. MRLOE02 에 EMP 테이블 객체를 조회할 수 있도록 SELECT 권한을 주었다. 생성된 룰을 사용자에게 부여하기 위해서 다시 데이터베이스 관리자로 로그인하여 사용자 USER05에게 룰에 대한 권한 부여한다.

예

```
CONN system/password  
GRANT MROLE02 TO USER05;
```

- 4. 사용자 USER05에게 룰에 대한 권한 부여를 마쳤으면 사용자 USER05로 로그인하여 룰에 대한 권한이 부여되었는지 확인해보자.

예

```
CONN USER05/TIGER  
SELECT * FROM USER_ROLE_PRIVS;
```

USERNAME	GRANTED_ROLE	ADMIN_OPTION	DEFAULT_ROLE	OS_GRANTED
USER05	MROLE	NO	YES	NO
USER05	MROLE02	NO	YES	NO

4. 룰 회수하기[1]

- ❖ 룰 역시 권한처럼 사용하지 않게 되었을 경우 이를 회수할 수 있습니다. 다음은 **룰을 회수하기** 위한 REVOKE 명령어의 형식이다.

형식

REVOKE *role_name* FROM *user_name*;

- ❖ 룰을 삭제하기 위한 DROP ROLE 명령어의 형식

형식

DROP ROLE *role_name*;

4. 룰 회수하기[2]

- ❖ 이번에는 REVOKE 문을 사용하여 룰을 회수해 보도록 하자.
- ❖ 1. 현재 사용자에게 부여된 룰 권한을 확인하기 위해서 다음과 같이 명령 문을 수행해 보도록 한다.

예

CONN USER05/TIGER
SELECT * FROM USER_ROLE_PRIVS;

USERNAME	GRANTED_ROLE	ADMIN_OPTION	DEFAULT_ROLE	OS_GRANTED
USER05	MROLE	NO	YES	NO
USER05	MROLE02	NO	YES	NO

4. 룰 회수하기(3)

- ❖ 2. 데이터베이스 관리자로 접속한 후에 룰을 회수한다.

예

```
CONN system/password  
REVOKE MROLE02 FROM USER05;
```

- ❖ 3. 다시 USER05로 접속하여 USER05에 부여된 룰을 확인해보면 MROLE02 룰이 회수된 것을 확인할 수 있다.

예

```
CONN USER05/TIGER  
SELECT * FROM USER_ROLE_PRIVS;
```

USERNAME	GRANTED_ROLE	ADMIN_OPTION	DEFAULT_ROLE	OS_GRANTED
USER05	MROLE	NO	YES	NO

4.1 롤 제거하기[1]

- ❖ 1. 사용자 USER05에게 부여되었던 룰에 대한 권한만을 회수할 뿐, 룰 MROLE02은 아직 존재한다.
- ❖ SYSTEM 계정에서 룰을 생성하였으므로 SYSTEM 계정으로 접속하여 데이터 딕셔너리 USER_ROLE_PRIVS의 내용을 출력하여 MROLE02이 존재를 확인하다.

예

```
CONN system/password  
SELECT * FROM USER_ROLE_PRIVS;
```

USERNAME	GRANTED_ROLE	ADMIN_OPTION	DEFAULT_ROLE	OS_GRANTED
SYSTEM	AQ ADMINISTRATOR ROLE	YES	YES	NO
SYSTEM	DBA	YES	YES	NO
SYSTEM	MROLE	YES	YES	NO
SYSTEM	MROLE02	YES	YES	NO

4.1 롤 제거하기[2]

- 2. MROLE02을 제거해보자. MROLE02을 제거한 후 USER_ROLE_PRIVS 데 이터 딕셔너리를 살펴보면 MROLE02이 나타나지 않는다.

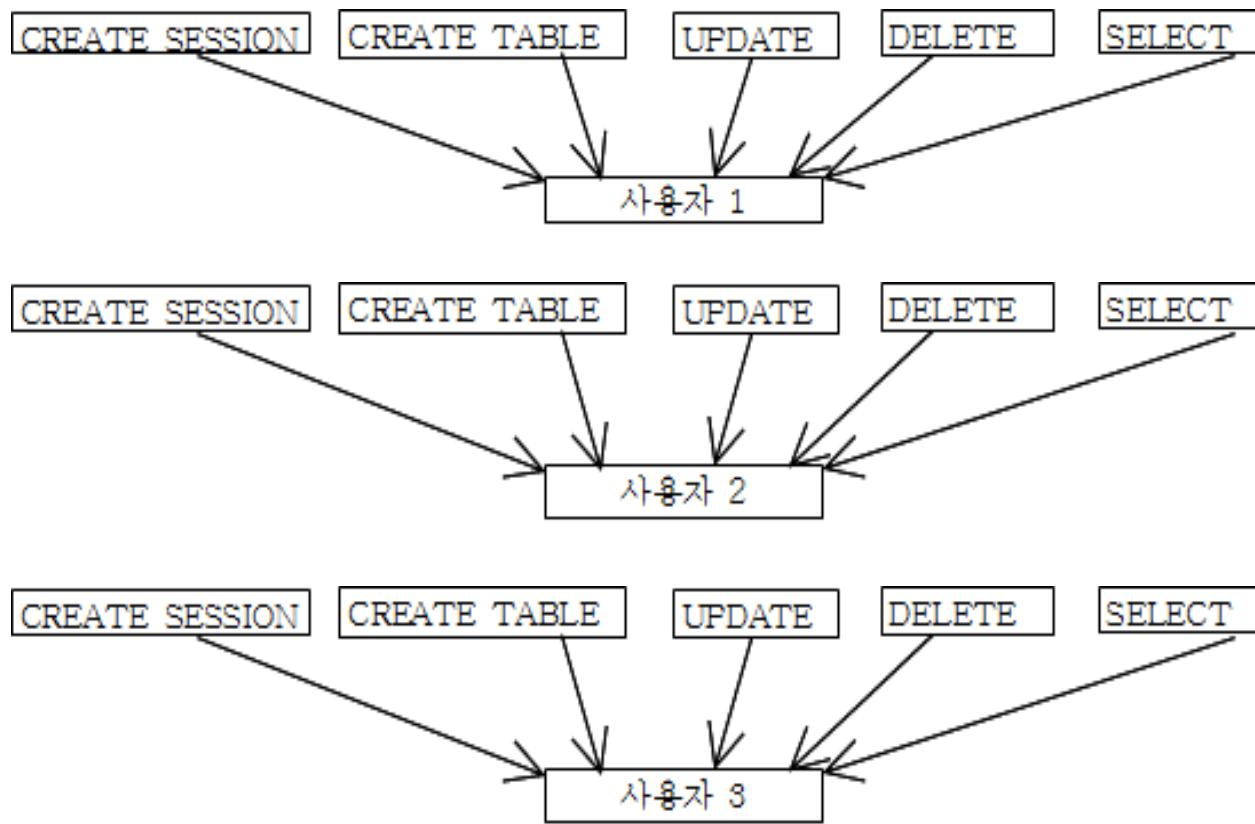
예

```
DROP ROLE MROLE02;  
SELECT * FROM USER_ROLE_PRIVS;
```

USERNAME	GRANTED_ROLE	ADMIN_OPTION	DEFAULT_ROLE	OS_GRANTED
SYSTEM	AQ ADMINISTRATOR ROLE	YES	YES	NO
SYSTEM	DBA	YES	YES	NO
SYSTEM	MROLE	YES	YES	NO

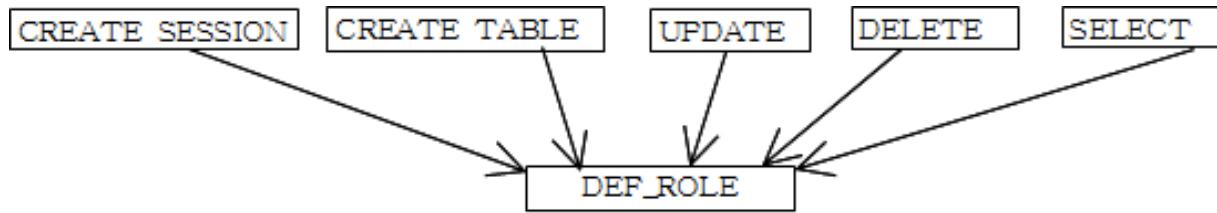
5. 룰의 장점(1)

- ❖ 이번에는 룰을 사용하여 권한 부여함으로서 생기는 장점에 대해서 살펴보겠다. 시스템권한이나 객체 권한을 사용자마다 일일이 부여하게 되면 번거롭다.



5. 롤의 장점[2]

- ❖ 3명의 사용자에게 일일이 권한 부여하는 명령어를 부여하려면 굉장히 번거롭다.
- ❖ 이러한 단점을 롤로 보안할 수 있다, 우선 롤에 시스템 권한과 객체 권한을 부여한다.



- ❖ 그런 후에 를을 사용자에 대해 권한 부여함으로서 작업을 간소화할 수 있다.

