6.4 Authorization

권한

1. instance - read, insert, update, elete
2. schema - index(create, delete), resource(create new relation), alteration, drop

1. SQL 권한

- select
- insert
- update
- delete
- reference -> foreign key 참조하는 사람만 해당 key 참조 가능
- usage - 도메인 쓸 수 있는 권한
- all privillage..

Grant - 권한 부여

```
Grant <privilege list> on <relation name or view name> to <user list> [with grant option]
```

--> [with grant option] 이용시 권한 부여받은 사람이 다시 권한 부여할 수 있음

ex.

```
Grant select on professor to U1, U2, U3;
//U1, U2, U3에게 professor table select 할 수 있는 권한

Grant references (deptname) on department to Lee;
//Lee에게 department의 deptname을 참조할 수 있는 권한 부여
```

Revoke - 권한 취소

```
Revoke <privilege list> on <relation name or view name> from <user list> [restrict|cascade];
```

- privilege list -> all 표현 가능
- 권한 두 번 부여시 하나 취소되도 계속 사용 가능

option

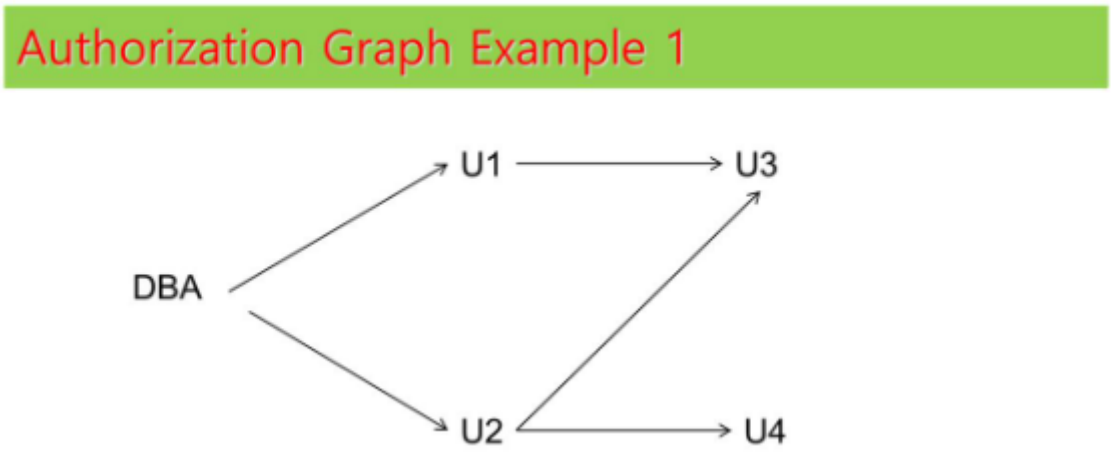
- cascade (default) -> 취소당하는 user가 같은 권한을 다른 user에게 부여했다면 해당 유저도 권한 취소됨 (전파)
- restrict -> 전파되어야 하는 상황에서 권한 취소시 해당 revoke 제한함 (fail) //side effect 방지

revoke list

- public -> 모든 사람에게 권한 한번씩 취소 (권한 중복으로 받은 사람은 여전히 권한 유지)
- grant option 해당 relation에서만 권한 취소

Authorization Graph

권한 부여한 그래프..



- DBA -> U2 없애도 U3는 권한 사용 가능 (cascade default option)

2. Authorizaition on view

- base table의 권한과 독립적으로 움직임
- view에 권한 부여해도 base table 권한과는 무관

view 만드는 권한은 resource 권한 없어도 상관없음 -> 실제로 table을 만드는 게 아니라서

base: table 만든 사람은 모든 권한 가짐 view: 그렇지 못함..

base table에 대한 select 권한만 가지고 view 만들기 --> 이러면 해당 view에 대한 select 권한만 존재함 & 최소 base table의 read(select)권한 가지고있어야 뷰 생성 가능 비록 insert, update ...등등은 불가

3. Roles

사용자를 집합으로 만들어서 해당 집합에 권한 부여

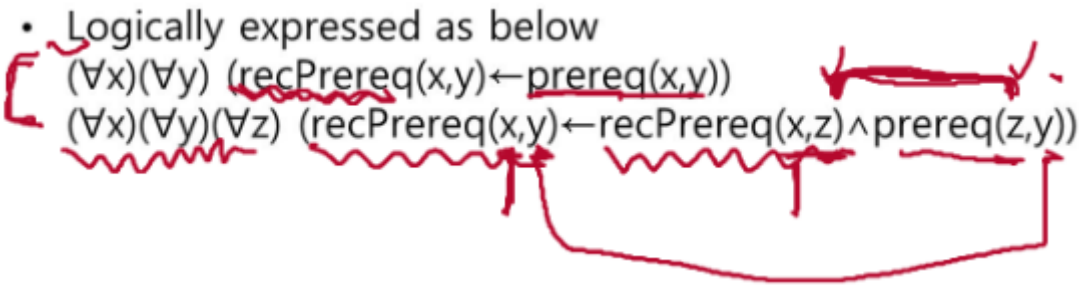
```
create role teller;
grant select on branch to teller;
grant update(balance) on account to teller;
//role 만들어서 권한 부여

create role manager;
grant teller to manager;
//role의 권한 또다른 role에 그대로 부여
```

4. Limitations of SQL Authorization

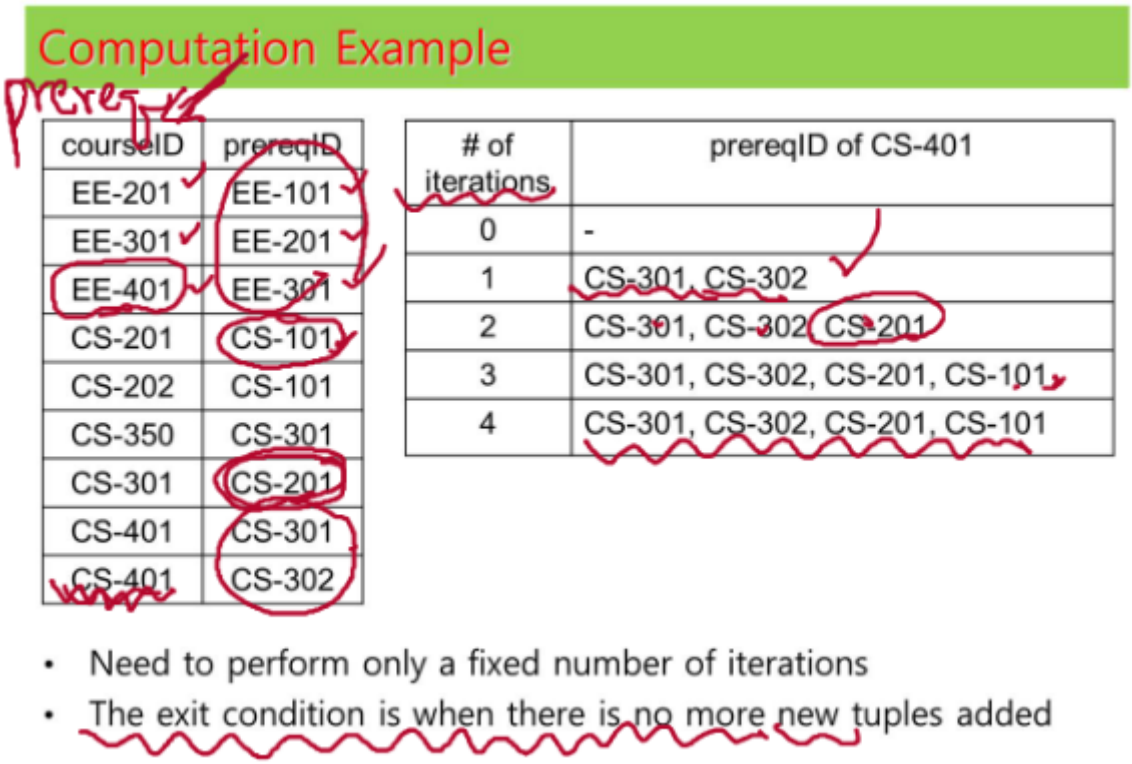
- tuple 단위로 권한 부여 불가
- 하지만 Web 단위에서 Application이 권한 관리한다면... tuple단위로 제어 가능

6.5 Recursive Queries

자기 자신의 view로 또다른 view 만들기 선수과목 구하기(선수 1, 후수 1) 후수 1을 듣기위해 또 들어야 할 선수과목 있을수도..



-> x가 반드시 앞에 나와야하고 recursive의 결과로 y



CS-401을 듣기 위해

1. CS-302, 301
2. CS-301을 듣기 위해 CS-201 ... 이렇게 계속 추가해나가기 추가 안될때까지 재귀

Transitive Closure

이러한 과정을 이렇게 말함

- final result = fixed point임
- Recursive views are required to be monotonic --> 새로운 tuple이 없으면 stop

```
With recursive recPrereq (courseID, prereqID) as
(( select courseID, prereqID
from prereq)
union
(select recPrereq.courseID, prereq.prereqID
from recPrereq, prereq
where recPrereq.prereqID))
select *
from reqPrereq;

//x,y <- (x,z) (z,y) 형태
```