

# 제 5장 무결성과 보안

- 무결성 제약
- 데이터베이스 보안
- 오라클에서의 무결성과 보안

# 무결성 제약

---

- ▶ 무결성 제약(integrity constraint) 또는 무결성 규칙(integrity rule)
  - ▶ 데이터베이스에 저장된 데이터가 실제 세계에 존재하는 정보들을 모순 없이 반영하는 성질
  - ▶ 데이터베이스에 저장된 데이터가 갖추어야할 제약 조건을 항상 만족하도록 보장하는 성질
- ▶ 무결성 제약의 예
  - ▶ 학생은 하나의 학과에 소속된다.
  - ▶ 하나의 강좌는 한 명의 담당교수가 배정된다.
  - ▶ 하나의 교과목은 각 학기마다 두 강좌 이하만 개설할 수 있다.
  - ▶ 학생은 한 학기에 20학점 이상 수강할 수 없다.

# 무결성 제약을 위반한 예

stu_id	resident_id	name	year	address	dept_id	dept_id	dept_name	office
1292001	900424-1825409	김광식	3	서울	920	920	컴퓨터공학과	201호
1292002	900305-1730021	김정현	3	서울	920	923	산업공학과	207호
1292003	891021-2308302	김현정	4	대전	923	925	전자공학과	308호
1292301	890902-2704012	김현정	2	대구	900			

학과번호가 '900'인 학과는 없음

(a) student 테이블과 department 테이블

class_id	course_id	year	semester	division	prof_id	classroom	enroll
C101-01	C101	2012	1	A	92301	301호	40
C102-01	C102	2012	1	A	92001	209호	30
C103-01	C103	2012	1	A	92501	208호	30
C103-02	C103	2012	1	B	92301	301호	30
C103-03	C103	2012	1	C	92001	209호	40

동일 교과목에 대해서 같은 학기에 3개 이상의 강좌가 개설됨

# 무결성 제약

---

- ▶ 무결성 제약 조건을 충족하기 위한 방법
  - ▶ 데이터를 삽입, 삭제, 수정할 때 마다 수동으로 주어진 무결성 제약의 만족 여부를 검증
    - ▶ 단순한 방법
    - ▶ 데이터의 양이 방대하거나 데이터를 변경하는 빈도가 높을 경우 거의 불가능
  - ▶ 데이터베이스 설계자가 데이터베이스가 지켜야 할 무결성 제약들을 정의
    - ▶ DBMS가 이들을 자동으로 보장해 줄 수 있는 장치

# 무결성 제약의 유형

분류	의미	무결성 제약의 종류 및 방식
기본적 무결성 제약	관계형 데이터 모델에서 정의한 무결성 제약	기본키 무결성 제약
		참조 무결성 제약
테이블의 무결성 제약	테이블을 정의하거나 변경 과정에서 설정 가능한 무결성 제약	NOT NULL
		UNIQUE
		CHECK
		DEFAULT
기타 무결성 제약	위의 방법으로 정의할 수 없는 무결성 제약	주장(assertion)
		트리거(trigger)
		데이터베이스 프로그래밍을 이용한 무결성 제약 설정

# 기본키 무결성 제약

---

- ▶ primary key integrity constraint
- ▶ 테이블에서 레코드들이 반드시 유일하게 식별될 수 있어야 한다는 조건

## **정의: 기본키 무결성 제약**

기본키는 널 값을 가질 수 없으며 기본키의 값이 동일한 레코드가 하나의 테이블에 동시에 두 개 이상 존재할 수 없다.

# 기본키 무결성 제약

## ▶ 설정 방법

### ▶ 형식

**constraint** <제약식명> **primary key** (<필드리스트>)

#### ▶ <제약식명>

- 기본키를 정의하는 제약식에 주어진 이름이고

#### ▶ <필드리스트>

- 기본키로 정의할 필드들의 리스트

## ▶ 예

### (질의 1)

```
create table student
(
    stu_id                varchar2(10),
    resident_id           varchar2(14),
    name                  varchar2(10),
    year                  int,
    address               varchar2(10),
    dept_id               varchar2(10),
    constraint            pk_student primary key (stu_id)
)
```

# 기본키 무결성 제약

- ▶ 사용자가 기본키를 정의하여 테이블을 생성하면 기본키 무결성 제약은 DBMS에서 자동적으로 검증
  - ▶ 이러한 제약을 위반하는 레코드가 입력될 때 DBMS에서 자동적으로 거부
- ▶ 다음과 같이 간단한 표현할 수도 있음

(질의 2)

```
create table student
```

```
(
```

```
    stu_id
```

```
    varchar2(10)
```

```
    primary key,
```

```
    resident_id
```

```
    varchar2(14),
```

```
    name
```

```
    varchar2(10),
```

```
    year
```

```
    int,
```

```
    address
```

```
    varchar2(10),
```

```
    dept_id
```

```
    varchar2(10)
```

```
)
```

- ▶ 기본키가 하나의 필드로만 구성될 때만 가능
- ▶ 제약식에 이름이 부여되지 않아 추후에 기본키를 취소하거나 변경할 경우 현재의 기본키를 지정하지 못하는 문제가 발생



# 기본키 무결성 제약

- ▶ 테이블 생성 당시에 기본키를 설정하지 않았다면?
  - ▶ 나중에 **alter table**문을 이용하여 기본키를 별도로 설정
  - ▶ 예)

(질의 4)

```
alter table      student
add constraint   pk_student      primary key (stu_id)
```

- ▶ 기본키 삭제
  - ▶ 예)

(질의 5)

```
alter table      student
drop constraint   pk_student
```

# 참조 무결성 제약

- ▶ 한 테이블의 레코드가 다른 테이블을 참조
  - ▶ 참조되는 테이블에 해당 레코드가 반드시 존재하거나 널 값을 가짐
  - ▶ 이 조건이 지켜지지 않는다면, 참조하는 레코드는 실제로 존재하지 않는 레코드를 참조하게 되는 오류가 발생
  - ▶ 외래키의 조건과 일치
- ▶ 실제 존재하지 않는 잘못된 값이 저장되지 않도록 보장하는 수단

## ▶ 형식

**constraint  
references**

<제약식명>

<테이블이름> (<필드리스트2>)

**foreign key** (<필드리스트1>)

- ▶ <제약식명>: 외래키를 정의하는 제약식에 주어진 이름
- ▶ <필드리스트1> : 외래키로 정의하는 필드들의 리스트
- ▶ <테이블이름> : 참조 대상인 테이블의 이름
- ▶ <필드리스트2> : <테이블이름>의 기본키

# 참조 무결성 제약

## ▶ 예)

(질의 6)

```
create table student
(
    stu_id          varchar2(10),
    resident_id     varchar2(14),
    name            varchar2(10),
    year            int,
    address          varchar2(10),
    dept_id          varchar2(10),
    constraint      fk_dept foreign key (dept_id)
                    references      department (dept_id)
)
```

- ▶ 참조 무결성 제약은 외래키 정의에 의해 DBMS에서 자동적으로 검증
  - ▶ DBMS는 이 조건을 위반하게 되는 연산의 실행을 거부

# 참조 무결성 제약

- ▶ 외래키가 하나의 필드로만 구성되거나 제약식에 이름을 부여하지 않을 때 다음과 같이 정의 가능

(질의 7)

```
create table student
```

```
(
```

```
    stu_id
```

```
varchar2(10),
```

```
    resident_id
```

```
varchar2(14),
```

```
    name
```

```
varchar2(10),
```

```
    year
```

```
int,
```

```
    address
```

```
varchar2(10),
```

```
    dept_id
```

```
varchar2(10)
```

```
foreign key
```

```
references
```

```
department (dept_id)
```

```
)
```

# 참조 무결성 제약

- ▶ 테이블 생성 당시에 외래키를 설정하지 않았다면?

- ▶ **alter table**문을 이용하여 외래키를 별도로 설정

- ▶ 예)

(질의 8)

```
alter table  
add constraint
```

```
student  
fk_dept  
references
```

```
foreign key (dept_id)  
department (dept_id)
```

- ▶ 외래키 삭제

- ▶ 예)

(질의 9)

```
alter table  
drop constraint
```

```
student  
fk_dept
```

# 테이블의 무결성 제약

---

- ▶ 테이블 스키마를 정의할 때 지켜야 하는 무결성 제약
  - ▶ **not null**
  - ▶ **unique**
  - ▶ **check**
  - ▶ **default**

# NOT NULL

- ▶ 특정 필드에 대해서 널 값의 입력을 허용하지 않아야 되는 경우
- ▶ 예)

(질의 10)

```
create table student
(
    stu_id                varchar2(10),
    resident_id           varchar2(14)    not null,
    name                  varchar2(10),
    year                  int,
    address               varchar2(10),
    dept_id               varchar2(10)
)
```

- ▶ 기본키로 정의된 필드에 대해서는 명시적으로 **not null** 조건을 설정하지 않아도 됨

# UNIQUE

- ▶ 해당 필드가 테이블 내에서 중복된 값을 갖지 않고 유일하게 식별되도록 하는 제약 조건
- ▶ UNIQUE은 후보키의 자격이 있음
  - ▶ 단, **not null**로 지정되지 않는 한 널 값의 입력을 허용
- ▶ 형식

**constraint** <제약식명> **unique** (<필드리스트>)

  - ▶ <제약식명> : 제약식의 이름
  - ▶ <필드리스트> : **unique**을 설정할 필드들의 리스트



# UNIQUE

---

▶ 예)

(질의 11)

```
create table student
(
    stu_id          varchar2(10),
    resident_id     varchar2(14),
    name            varchar2(10),
    year            int,
    address          varchar2(10),
    dept_id         varchar2(10),
    constraint uc_rid unique (resident_id)
)
```

# UNIQUE

- ▶ 예) 두 개 이상의 필드에 동시에 설정 가능

(질의 12)

```
create table student
```

```
(
```

```
    stu_id                varchar2(10),
```

```
    resident_id           varchar2(14),
```

```
    family_name            varchar2(10),
```

```
    given_name             varchar2(10),
```

```
    Year                   int,
```

```
    Address                varchar2(10),
```

```
    dept_id                varchar2(10)
```

```
    constraint            uc_name                unique (family_name, first_name)
```

```
)
```

- ▶ 두 필드 각각에 대해서 유일해야 한다는 조건이 아니고 이름과 성이 동시에 같은 경우에 대해 중복을 허용하지 않음

# UNIQUE

- ▶ 예) 제약식에 이름을 부여하지 않거나 하나의 필드에 대해서만 설정할 경우 다음과 같은 표현이 가능

(질의 13)

```
create table student
(
    stu_id          varchar2(10),
    resident_id     varchar2(14)    unique,
    name            varchar2(10),
    year            int,
    address          varchar2(10),
    dept_id         varchar2(10)
)
```

# UNIQUE

---

- ▶ unique을 별도로 지정

(질의 14)

```
alter table      student
add constraint   uc_rid      unique (resident_id)
```

- ▶ unique 설정 해제

(질의 15)

```
alter table      student
drop constraint   uc_rid
```

# CHECK

---

- ▶ 도메인 제약(domain constraint)
  - ▶ 각 필드의 값은 정의된 도메인에 속한 값만 허용하는 성질
  - ▶ CHECK
    - ▶ 필드를 정의할 때 주어진 데이터 타입 이외에도 좀 더 세부적으로 허용할 수 있는 값의 범위를 지정
- ▶ 형식

**constraint** <제약식명> **check** (<조건식>)

- ▶ <조건식> 만족해야할 필드들의 조건

# CHECK

## ▶ 예)

(질의 16)

```
create table student
```

```
(
```

```
    stu_id          varchar2(10),
```

```
    resident_id     varchar2(14),
```

```
    name            varchar2(10),
```

```
    year            int,
```

```
    address          varchar2(10),
```

```
    dept_id          varchar2(10),
```

```
    constraint      chk_year check (year >= 1 and year <= 4)
```

```
)
```

# CHECK

▶ 예)

(질의 17)

**create table** student

(

stu\_id

**varchar2**(10),

resident\_id

**varchar2**(14),

Name

**varchar2**(10),

Year

**int**,

Address

**varchar2**(10),

dept\_id

**varchar2**(10) **foreign key**

**references** department (dept\_id),

**constraint**

chk1

**check**

(year >= 1 **and**  
year <= 4 **and**  
address **in** ('서울', '부산'))

)

# CHECK

- ▶ 제약식에 이름을 부여하지 않거나, 각 필드에 대한 조건식을 분리하여 명시 가능

(질의 18)

```
create table student
```

```
(
```

```
    stu_id                varchar2(10),
```

```
    resident_id           varchar2(14),
```

```
    name                  varchar2(10),
```

```
    year                  int
```

```
    address                varchar2(10)
```

```
    dept_id               varchar2(10)
```

```
)
```

```
check (year >= 1 and year <= 4),  
check (address in ('서울', '부산')),
```



# CHECK

---

- ▶ check를 별도로 지정

(질의 19)

```
alter table      student
add constraint   chk_year check (year >= 1 and year <= 4)
```

- ▶ check를 해제

(질의 20)

```
alter table      student
drop constraint   chk_year
```

# DEFAULT

- ▶ 레코드를 삽입할 때, 필드에 대한 값이 정해지지 않았을 경우 사전에 정해놓은 값으로 입력하도록 지정
  - ▶ 널 값 대신에 지정된 값이 자동적으로 입력

▶ 예)

(질의 21)

```
create table student
(
    stu_id          varchar2(10),
    resident_id     varchar2(14),
    name            varchar2(10),
    year            int          default 1,
    address         varchar2(10),
    dept_id         varchar2(10)
)
```

# DEFAULT

---

- ▶ default를 별도로 설정

(질의 22)

```
alter table student  
alter column year set default 1
```

- ▶ default 해제

(질의 23)

```
alter table student  
alter column year drop default
```

## 참고(DEFAULT)

- ▶ 오라클은 default에 대한 별도의 설정 및 해제에 SQL을 표준을 따르지 않음
- ▶ 오라클에서는 (질의 22)와 (질의 23)대신에 다음과 같은 형식을 사용해야 함

<b>alter table</b> <b>modify</b>	student (year <b>int</b> <b>default</b> 1)
-------------------------------------	---

<b>alter table</b> <b>modify</b>	student (year <b>int</b> <b>default</b> null)
-------------------------------------	--

# 무결성 제약 설정의 유의점

---

- ▶ 지나치게 많은 제약조건들이 존재할 경우 예외적인 데이터가 발생할 때 문제가 발생할 수 있음
  - ▶ 예) 외국인의 경우 주민등록번호가 없으므로 student 테이블의 resident\_id에 값을 입력할 수 없음
  - ▶ 그러나 이 필드에는 **not null** 조건이 있으므로 삽입이 불가능
- ▶ 각 레코드에 대한 삽입, 삭제, 수정 연산을 할 때 정의된 무결성 제약들이 모두 만족하는지 검증함
  - ▶ 무결성 제약의 만족 여부를 검사하는 부담이 가중되어 전체적인 성능이 떨어지는 문제가 발생할 수 있음

# 기타 무결성 제약

---

- ▶ 지금까지의 방법만으로 모든 무결성 제약들을 표현할 수 없음
- ▶ 현실 세계의 제약들은 다양하고 복잡함
- ▶ 기타 무결성 제약 방법
  - ▶ 주장(Assertion)
  - ▶ 트리거(Trigger)
  - ▶ 응용 프로그램을 이용한 제약

# 주장(Assertion)

## ▶ 형식

```
create assertion <주장이름> check <조건식>
```

- ▶ 예) 교수는 한 학기에 3 강좌 이상을 강의할 수 없다.

(질의 24)

```
create assertion          assert1  check  
(not exists ( select          count(*)  
              from            class  
              group by        year, semester, prof_id  
              ) >= 3)         )
```

- ▶ 주장 명령문을 실행하면 데이터베이스는 현재 저장된 데이터들에 대해 해당 조건이 만족되는지 검증
- ▶ 이후에 발생하는 데이터베이스의 변경에 대해서도 각 주장에서 명시된 조건이 만족되는지 감시

# 응용 프로그램에서의 무결성 제약

---

- ▶ 데이터베이스 사용자가 응용 프로그램에서 직접 무결성을 보장하는 코드를 삽입
- ▶ 예) **check**를 사용하지 않는다면?
  - ▶ 새로운 레코드를 삽입하는 프로그램에서 **insert**나 **update**문을 실행하기 전에 year 필드의 값이 1에서 4까지의 정수인지를 판단
  - ▶ 만족하지 못할 경우 에러처리를 하는 코드를 삽입
- ▶ 번거롭고 어렵지만, DBMS가 제공하는 기능으로 해결하지 못할 경우 반드시 응용 프로그램에서 무결성을 만족시켜야 함
- ▶ DBMS가 일일이 검증해야하는 부담(성능 저하)을 줄일 수 있는 장점도 있음



# 데이터베이스 보안

---

- ▶ 데이터베이스 관리자(DBA)는 외부의 권한을 갖지 못한 사용자로부터 데이터를 안전하게 관리할 의무가 있음
- ▶ DBMS에서 제공하는 보안
  - ▶ 허가받지 않은 사용자, 즉, 권한이 없는 사용자로부터 데이터의 접근을 사전에 차단

# 데이터베이스에서의 사용권한

---

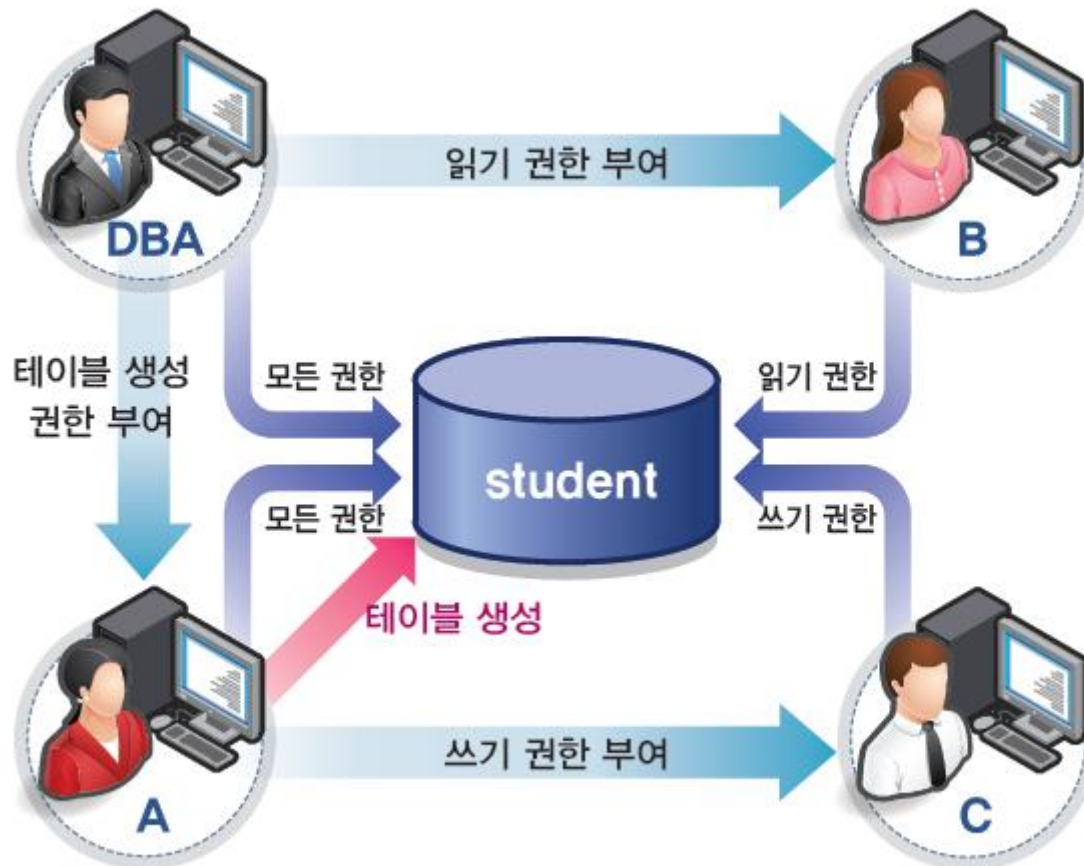
- ▶ 사용자가 특정 객체에 대해 특정 연산을 실행할 수 있는 권리
- ▶ 특정 객체
  - ▶ 테이블, 뷰, 필드 등 데이터베이스의 구성 요소
- ▶ 권한 제어가 가능한 연산의 종류
  - ▶ 데이터 접근관련 연산(DML)
    - SQL의 **select, insert, delete, update** 등
  - ▶ 스키마 관련 연산(DDL)
    - 스키마를 수정하는 연산
    - **create table, alter table, drop table, create index** 등

# 권한에 따른 사용자 분류

---

- ▶ 데이터베이스 관리자(DBA)
  - ▶ DBMS내의 모든 객체에 대해 모든 권한
  - ▶ 객체에 대한 연산, 객체의 제거와 변경을 포함
  - ▶ 다른 사용자에게 해당 객체에 대한 권한을 부여하거나 회수
- ▶ 객체 소유자(owner)
  - ▶ 특정 사용자가 객체를 생성한 사용자
  - ▶ 생성한 객체에 대해 모든 권한(해당 객체에 대한 권한의 부여나 회수도 포함)
- ▶ 기타 사용자
  - ▶ 기본적으로 객체에 대한 일체의 사용권한이 없음
  - ▶ 다만 데이터베이스 관리자나 객체 소유자로부터 일부 또는 모든 권한을 별도로 부여받을 수 있음

# 권한 부여의 예



# SQL에서의 권한제어 - GRANT

## ▶ GRANT

- ▶ 권한을 부여하는 명령
- ▶ 형식

**grant** <권한리스트> **on** <객체명> **to** <사용자리스트>

- ▶ <권한리스트> : 권한의 종류에 대한 리스트
    - select, insert, delete, update, references 중 한 개 이상
  - ▶ <객체명> : 대상이 되는 객체
  - ▶ <사용자리스트> : 권한을 부여받는 사용자들의 리스트
  - ▶ <사용자리스트>에게 <객체명>에 대한 <권한리스트>를 실행할 권리를 부여한다는 의미
- ▶ 예) ID **kim**인 사용자에게 student 테이블에 대해 **select** 연산을 수행할 수 있는 권한을 부여

(질의 25)

**grant select on student to kim**

# SQL에서의 권한제어 - GRANT

---

- ▶ 예) **kim**에게 **select**와 **delete** 권한을 부여

(질의 26)

```
grant select, delete on student to kim
```

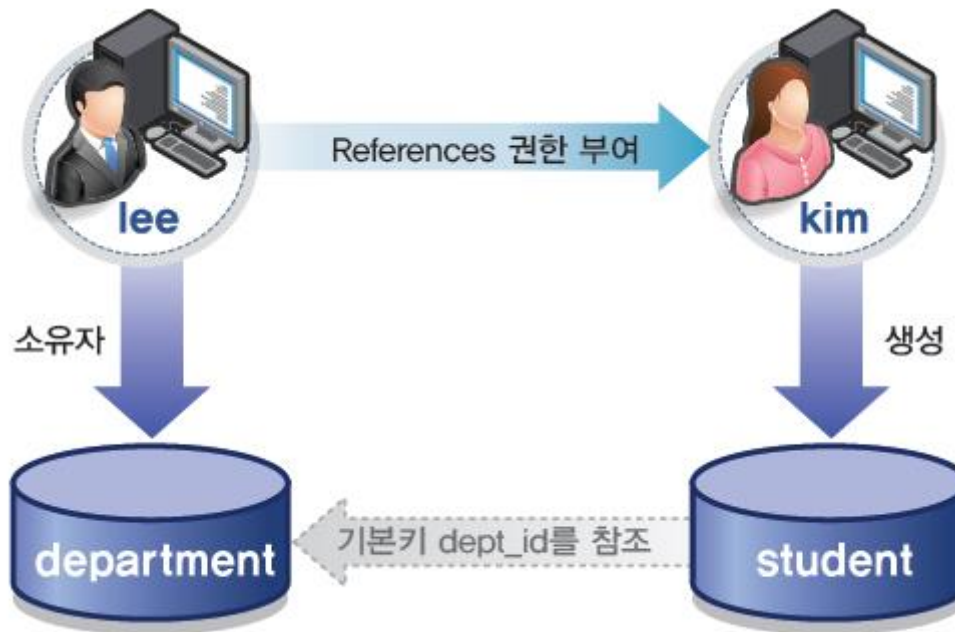
- ▶ <권한리스트>에서 **select**와 **update**는 특별히 테이블내의 특정 필드에 대해서만 연산을 허용할 수 있음
- ▶ 예) student 테이블에서 stu\_id에 대해서만 **select** 연산을 허용

(질의 27)

```
grant select (stu_id) on student to kim
```

# SQL에서의 권한제어 - GRANT

- ▶ references 권한이란?



- ▶ 권한이 없으면 kim이 생성한 student는 department를 외래키로 참조 못함

# SQL에서의 권한제어 - GRANT

---

- ▶ references 권한 부여 예)

(질의 28)

**grant references** (dept\_id) **on** department **to** kim

- ▶ dept\_id는 테이블 department의 기본키
- ▶ **kim**은 department 테이블을 참조하는 외래키를 포함하는 테이블의 생성이 가능



# SQL에서의 권한제어 - GRANT

---

- ▶ 모든 사용자에게 권한을 부여
  - ▶ <사용자리스트>에 **public**을 사용
- ▶ 예) student에 대해 모든 사용자들에게 select 권한을 부여

(질의 29)

```
grant select on student to public
```

- ▶ 모든 종류의 권한을 하나의 명령으로 부여하는 방법
  - ▶ all privileges라는 키워드를 사용
- ▶ 예) student 테이블에 대한 모든 권한을 lee에게 부여

(질의 30)

```
grant all privileges on student to lee
```

# SQL에서의 권한제어 - GRANT

---

- ▶ SQL 표준에서는 주로 읽기, 삽입, 삭제, 수정 등 데이터 접근에 관련된 권한만을 정의
- ▶ 오라클을 비롯한 일부 DBMS들은 create table, alter table, drop table과 같은 스키마 관련 연산에 대한 권한도 grant문으로 부여할 수 있는 기능을 제공

# WITH GRANT OPTION

- ▶ SQL에서는 부여받은 권한을 다른 사용자에게 전파할 수 있는 특별한 옵션

▶ 예)

(질의 31)

**grant select on student to kim with grant option**

- ▶ student 테이블에 대한 select 권한을 kim에게 부여함과 동시에 이 권한을 다른 사용자에게 다시 전파할 수 있는 자격까지 부여
- ▶ 예) kim이 다음 명령 실행 가능

(질의 32)

**grant select on student to chang**

- ▶ 단, chang은 더 이상 다른 사용자에게 권한의 전파가 불가능

- ▶ 권한을 전파할 수 있는 막강한 수단
  - ▶ 하지만 이 옵션이 남용될 경우 데이터 보안에 심각한 문제를 야기할 수 있음

# REVOKE

- ▶ 다른 사용자에게 부여한 권한을 회수하기 위한 명령

- ▶ 형식

**revoke** <권한리스트> **on** <객체명> **from** <사용자리스트>

- ▶ <사용자리스트>로 부터 <객체명>에 대한 <권한리스트>연산에 대한 권한을 회수

- ▶ 예) kim에게 부여되었던 student 테이블에 대한 select 권한을 회수

(질의 33)

**revoke select on student from kim**

# REVOKE

---

- ▶ 연쇄적인 회수가 되는 상황
- ▶ 예)

(질의 34) - lee가 실행

**grant select on student to kim with grant option**

(질의 35) - kim이 실행

**grant select on student to chang**

- ▶ (질의 33)의 revoke문은 kim 뿐만 아니라 kim이 권한을 부여한 chang으로 부터도 자동적으로 권한을 회수

# 롤(ROLE)

---

- ▶ 특정 테이블에 대한 권한을 부여할 사용자가 많다면, 반복적으로 grant문이나 revoke을 실행해야 하는 문제가 발생
- ▶ 단순 반복되는 작업을 줄이기 위해서 권한별로 사용자 그룹을 만들어 그룹에 권한을 부여하는 방법이 필요
- ▶ 예) 회사의 구성원들이 사원과 임원으로 구분
  - ▶ 각각의 보안등급이 다르면 사원 그룹과 임원 그룹으로 나누고 각 그룹에 해당 권한을 부여
- ▶ 롤(role)
  - ▶ 권한에 따른 사용자 그룹
  - ▶ 롤은 데이터베이스 관리자만이 생성 가능

# 롤의 생성

---

## ▶ 롤 생성 형식

```
create role <롤이름>
```

## ▶ 예) 사원(employee)과 임원(manager)에 대한 롤을 생성

(질의 36)

```
create role employee
```

(질의 37)

```
create role manager
```

# 룰에 사용자 배정

## ▶ 룰 배정 형식

**grant** <롤리스트> **to** <사용자리스트>

- ▶ <롤리스트>: 룰 이름의 리스트
- ▶ <사용자리스트>: <롤리스트>에 포함될 사용자 계정의 리스트
- ▶ 예) 사용자 lee와 kim은 사원, chang과 choi이 임원일 경우

(질의 38)

**grant** employee **to** lee, kim

(질의 39)

**grant** manager **to** chang, choi

- ▶ 예) 사용자 **park**을 employee와 manager에 모두 배정

(질의 40)

**grant** employee, manager **to** park



# 롤에 권한을 부여

---

- ▶ **grant**문의 형식과 동일
- ▶ 예) employee에게는 student 테이블에 대해 **select** 연산을 허용하고, manager에게는 **select**와 **insert** 연산을 허용

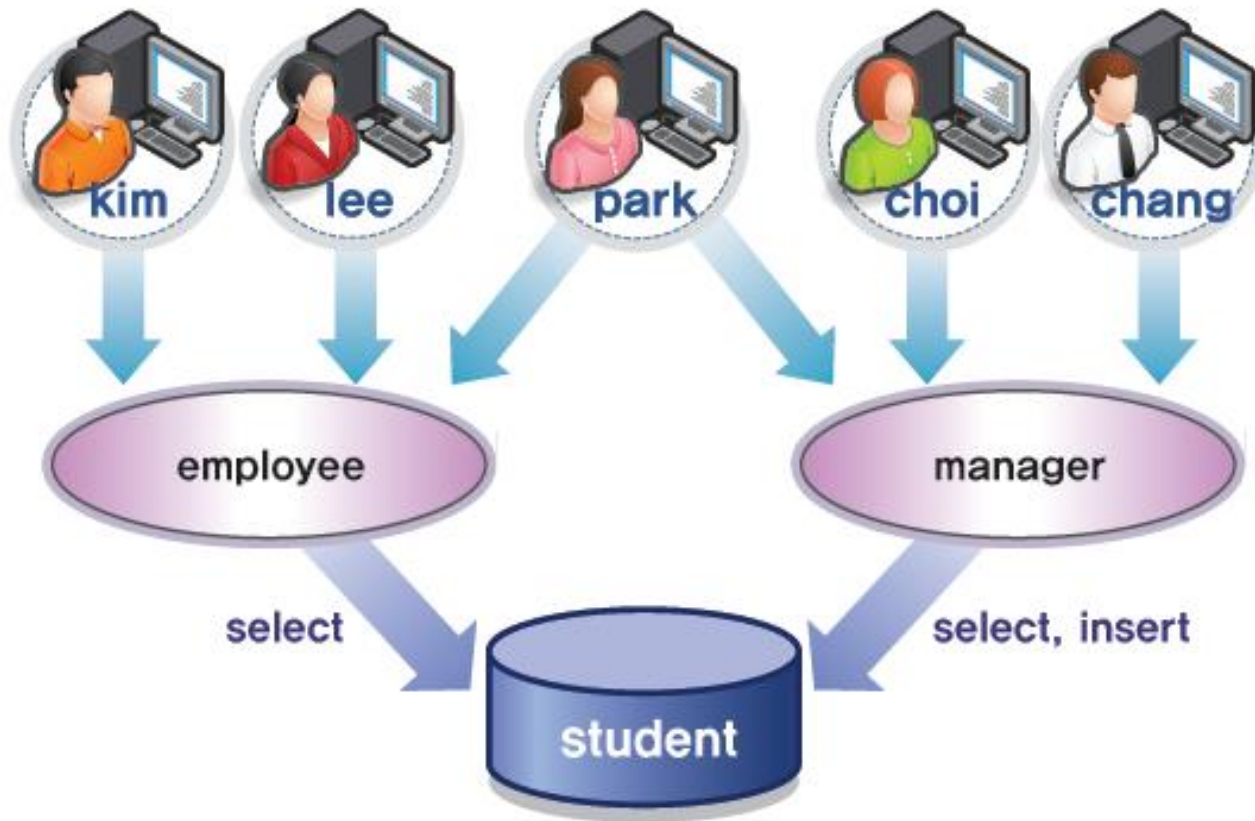
(질의 41)

```
grant select on student to employee
```

(질의 42)

```
grant select, insert on student to manager
```

# 롤의 생성과 권한 부여



# 롤에 부여된 권한을 회수

---

- ▶ **revoke**문과 동일
- ▶ 예) manager에게 부여된 **insert** 연산을 회수

(질의 43)

```
revoke insert on student from manager
```

# 배정된 롤에서 사용자를 배제

---

## ▶ 형식

**revoke** <롤리스트> **from** <사용자리스트>

## ▶ 예) 사용자 **choi**를 manager로부터 배제

(질의 44)

**revoke** manager **from** choi

# 롤 삭제

---

- ▶ 형식

```
drop role <롤이름>
```

- ▶ 예) manager를 데이터베이스에서 삭제

(질의 45)

```
drop role manager
```

# 뷰를 이용한 권한 제어

- ▶ 강력한 보안수단 제공
- ▶ 특정 테이블에서 일부 필드 혹은 일부 레코드에 대해서만 접근을 허용할 경우 이 부분들을 뷰로 정의
  - ▶ 정의된 뷰에 대해 접근 권한을 부여하고 실제 테이블에 대한 접근을 차단
  - ▶ 사용자에게 감추고 싶은 부분에 대한 보안이 자연스럽게 구현
  - ▶ **필드뿐만 아니라 레코드 일부에 대해서도 접근제어가 가능**
- ▶ 예)

(질의 46)

```
create view junior
as select stu_id, name, year, dept_id
from student
where year = 3
```

(질의 47)

```
grant select on junior to kim
```

- ▶ **kim**은 junior에 대한 접근은 허용되지만 실제 student 테이블에 대해서는 권한이 없으므로 접근이 차단

# 오라클에서의 무결성 제약

---

- ▶ 예) 다음의 순서로 데이터 준비

(질의 48)

```
create table department
(
    dept_id          varchar2(10),
    dept_name        varchar2(20) not null,
    office           varchar2(20),
    constraint      pk_dpt primary key (dept_id)
)
```

# 오라클에서의 무결성 제약

(질의 49)

```
create table student
```

```
(
```

```
    stu_id                varchar2(10),
```

```
    resident_id           varchar2(14) not null,
```

```
    name                  varchar2(10),
```

```
    year                  int default 1,
```

```
    address               varchar2(10),
```

```
    dept_id               varchar2(10),
```

```
    constraint            pk_student      primary key (stu_id),
```

```
    constraint            fk_dept         foreign key (dept_id)
```

```
    references            department (dept_id),
```

```
    constraint            uc_rid          unique (resident_id),
```

```
    constraint            chk_year        check (year >= 1 and year <= 4)
```

```
)
```



# 오라클에서의 무결성 제약

(질의 50)

**insert into** department **values** ('920', '컴퓨터공학과', '201호')

**insert into** department **values** ('923', '산업공학과', '207호')

**insert into** department **values** ('925', '전자공학과', '308호')

**insert into** student **values**  
('1292001', '900424-1825409', '김광식', 3, '서울', '920')

**insert into** student **values**  
('1292301', '890902-2704012', '김현정', 2, '대구', '923')

**insert into** student **values**  
('1292303', '910715-1524390', '박광수', 3, '광주', '923')

**insert into** student **values**  
('1292502', '911011-1809003', '백태성', 3, '서울', '925')

# 기본적 무결성 제약의 보장

---

- ▶ 기본키 무결성 제약
- ▶ 참조 무결성 제약

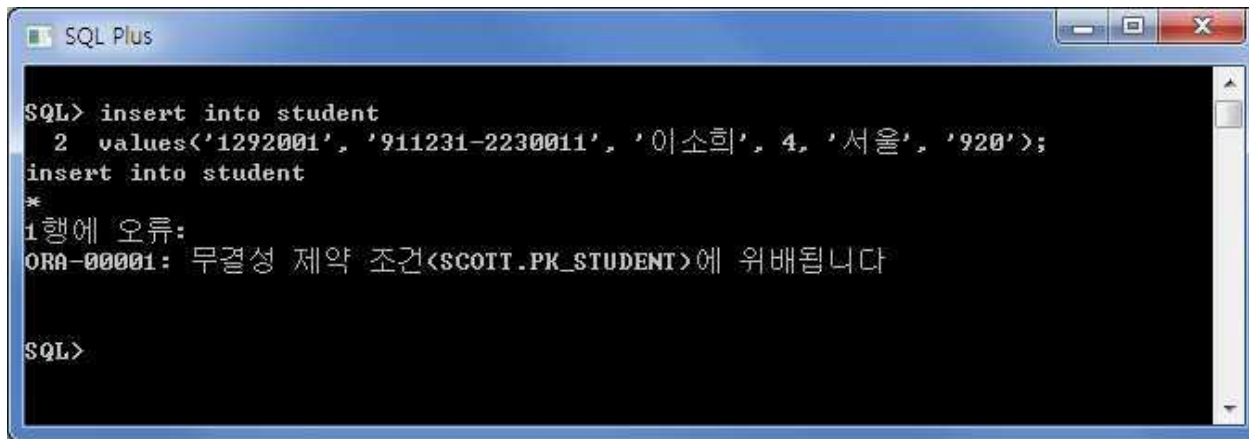
# 기본키 무결성 제약

- ▶ 다음의 레코드 삽입하면 오류

(질의 51)

**insert into student values**

('1292001', '911231-2230011', '이소희', 4, '서울', '920')



The screenshot shows a SQL Plus window with a black background and white text. The text displays the execution of an SQL insert statement and the resulting error. The error message is in Korean, indicating a violation of the primary key constraint for the SCOTT.PK\_STUDENT table.

```
SQL> insert into student
  2  values('1292001', '911231-2230011', '이소희', 4, '서울', '920');
insert into student
*
1행에 오류:
ORA-00001: 무결성 제약 조건(SCOTT.PK_STUDENT)에 위배됩니다

SQL>
```

- ▶ **<SCOTT.PK\_STUDENT>**

- scott : 사용자 ID
- pk\_student: 테이블을 생성할 때 부여된 무결성 제약의 이름

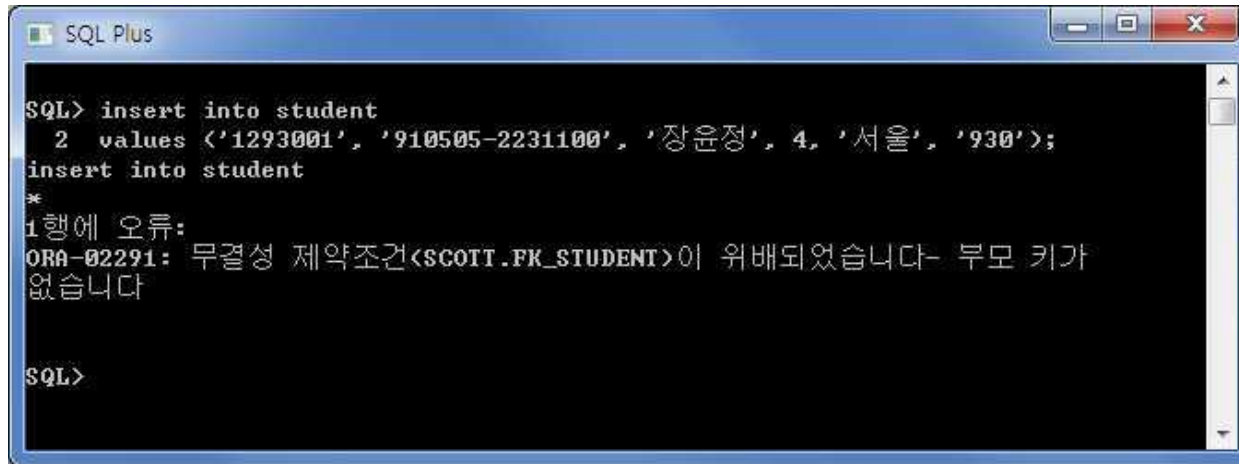
# 참조 무결성 제약

- ▶ 다음의 레코드 삽입하면 오류

(질의 52)

**insert into student values**

('1293001', '910505-2231100', '장윤정', 4, '서울', '930')



```
SQL> insert into student
  2  values (<'1293001', '910505-2231100', '장윤정', 4, '서울', '930');
insert into student
*
1행에 오류:
ORA-02291: 무결성 제약조건(SCOTT.FK_STUDENT)이 위반되었습니다- 부모 키가
없습니다

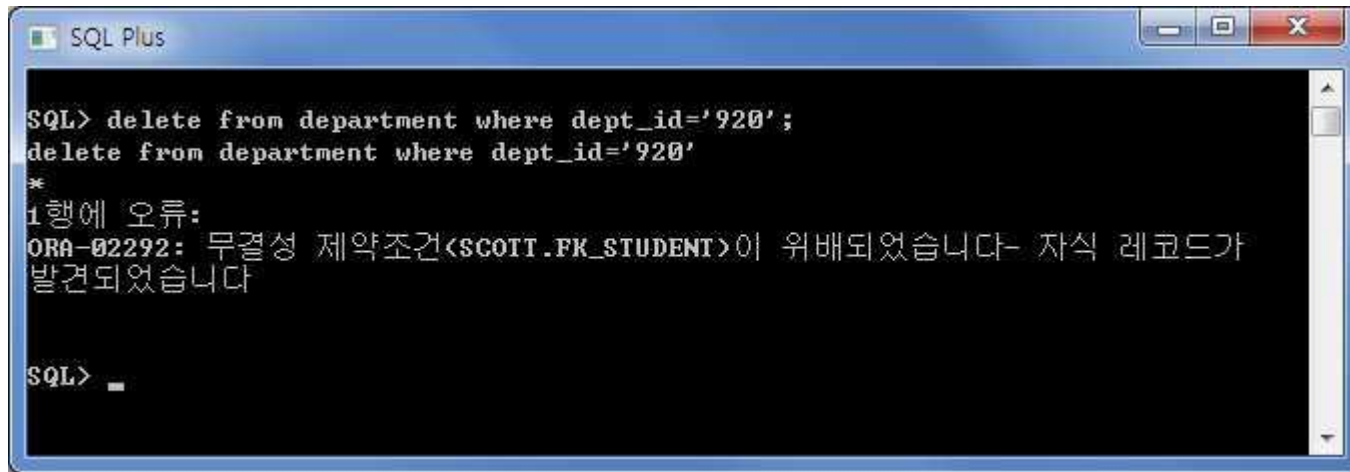
SQL>
```

# 참조 무결성 제약

- ▶ 다음의 레코드 삭제하면 오류

(질의 53)

**delete from** department **where** dept\_id='920'



The screenshot shows a SQL Plus window with a black background and white text. The command entered is 'delete from department where dept\_id='920';'. The output shows an error: 'ORA-02292: 무결성 제약조건(SCOTT.FK\_STUDENT)이 위배되었습니다. 자식 레코드가 발견되었습니다'.

```
SQL> delete from department where dept_id='920';
delete from department where dept_id='920'
*
1행에 오류:
ORA-02292: 무결성 제약조건(SCOTT.FK_STUDENT)이 위배되었습니다. 자식 레코드가
발견되었습니다

SQL> _
```

- ▶ 테이블을 삭제할 때도 참조 무결성 제약이 적용
  - ▶ department 테이블을 **drop table**문으로 삭제된다면, student 테이블의 dept\_id는 더 이상 department 테이블을 참조하지 못함

# 테이블 무결성 제약의 보장

---

- ▶ NOT NULL
- ▶ UNIQUE
- ▶ CHECK
- ▶ DEFAULT

# NOT NULL

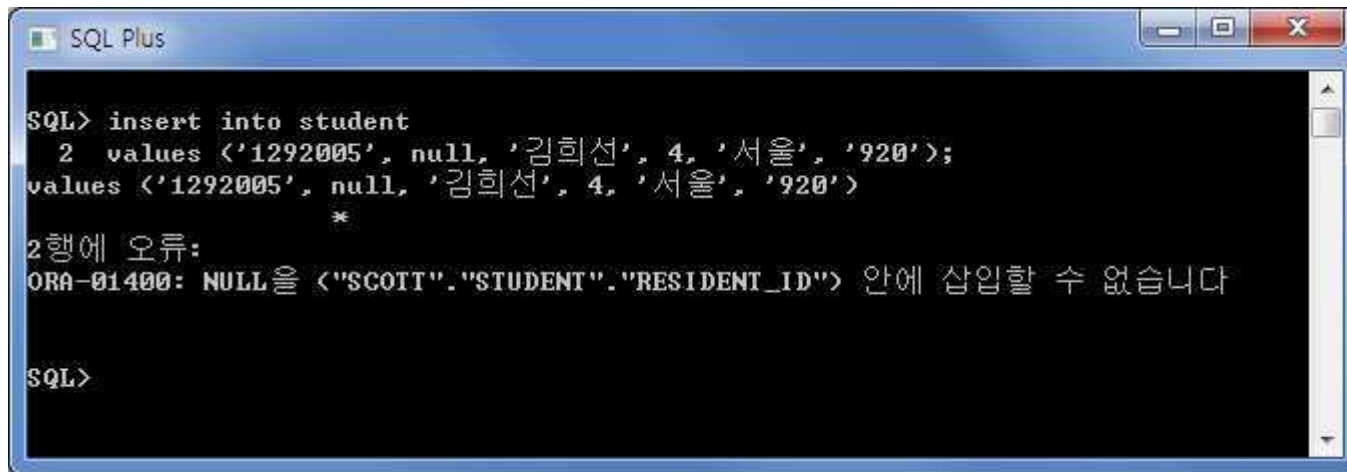
- ▶ 다음의 레코드 삽입하면 오류

(질의 54)

**insert into** student

**values**

('1292005', null, '김희선', 4, '서울', '920')

A screenshot of a SQL Plus window titled "SQL Plus". The window has a black background with white text. The text shows a successful insert statement followed by a duplicate insert statement that causes an error. The error message is in Korean and indicates that a NULL value cannot be inserted into the RESIDENT\_ID column of the SCOTT.STUDENT table.

```
SQL> insert into student
  2  values ('1292005', null, '김희선', 4, '서울', '920');
values ('1292005', null, '김희선', 4, '서울', '920')
      *
```

2행에 오류:  
ORA-01400: NULL을 <"SCOTT"."STUDENT"."RESIDENT\_ID"> 안에 삽입할 수 없습니다

SQL>

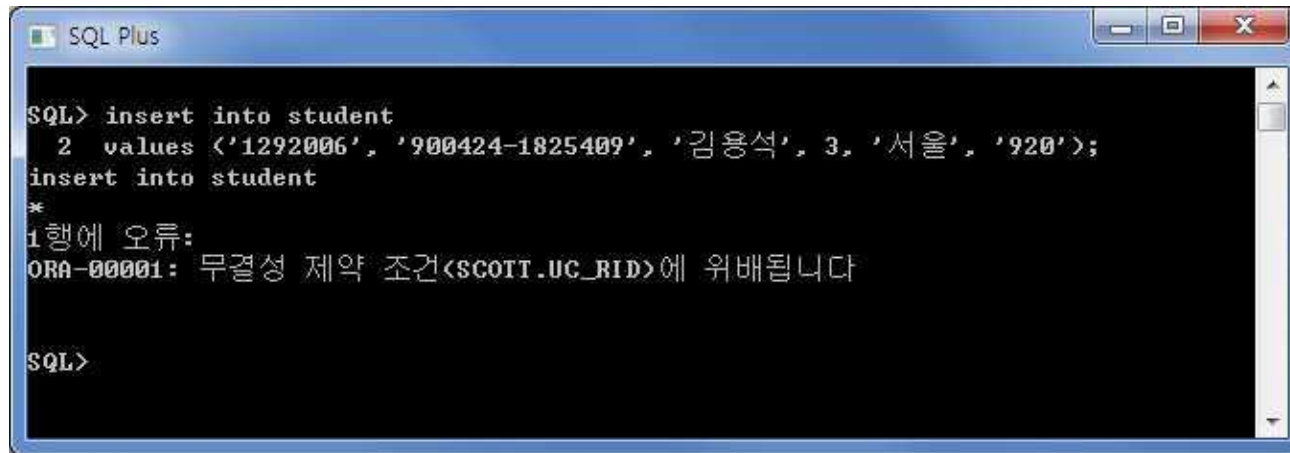
# UNIQUE

- ▶ 다음의 레코드 삽입하면 오류(resident\_id 중복)

(질의 55)

**insert into student values**

('1292006', '900424-1825409', '김용석', 3, '서울', '920')

A screenshot of a SQL Plus window. The window has a title bar that says "SQL Plus" and standard Windows window controls (minimize, maximize, close). The main area is a black terminal with white text. The text shows a successful insert command followed by a duplicate insert command that fails with an error.

```
SQL> insert into student
  2  values ('1292006', '900424-1825409', '김용석', 3, '서울', '920');
insert into student
*
1행에 오류:
ORA-00001: 무결성 제약 조건(SCOTT.UC_RID)에 위배됩니다

SQL>
```



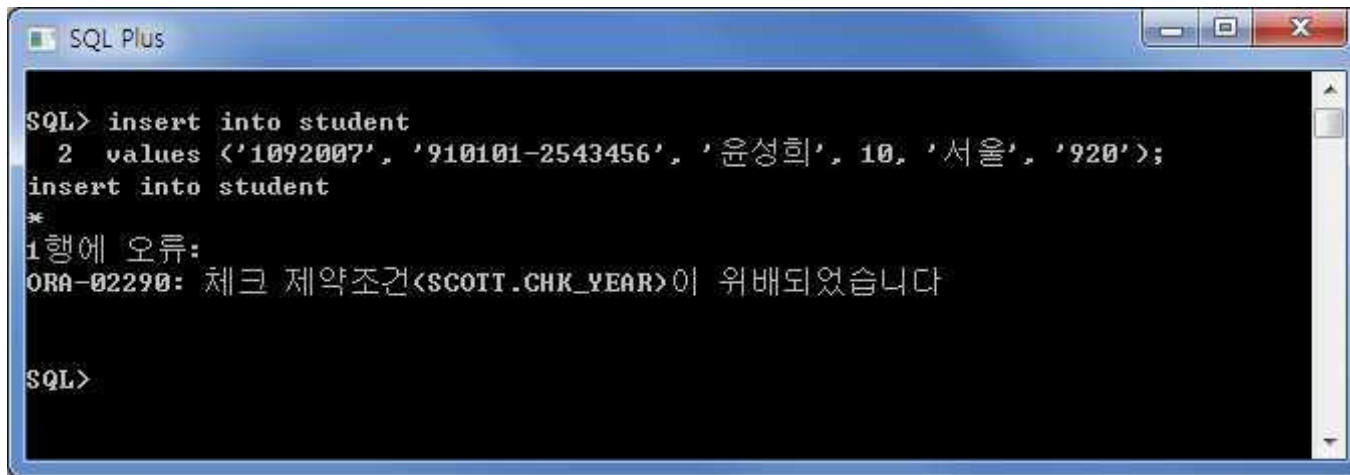
# CHECK

- ▶ 다음의 레코드 삽입하면 오류

(질의 56)

**insert into student values**

('1092007', '910101-2543456', '윤성희', 10, '서울', '920')



```
SQL> insert into student
  2 values ('1092007', '910101-2543456', '윤성희', 10, '서울', '920');
insert into student
*
1행에 오류:
ORA-02290: 체크 제약조건(SCOTT.CHK_YEAR)이 위배되었습니다

SQL>
```

# DEFAULT

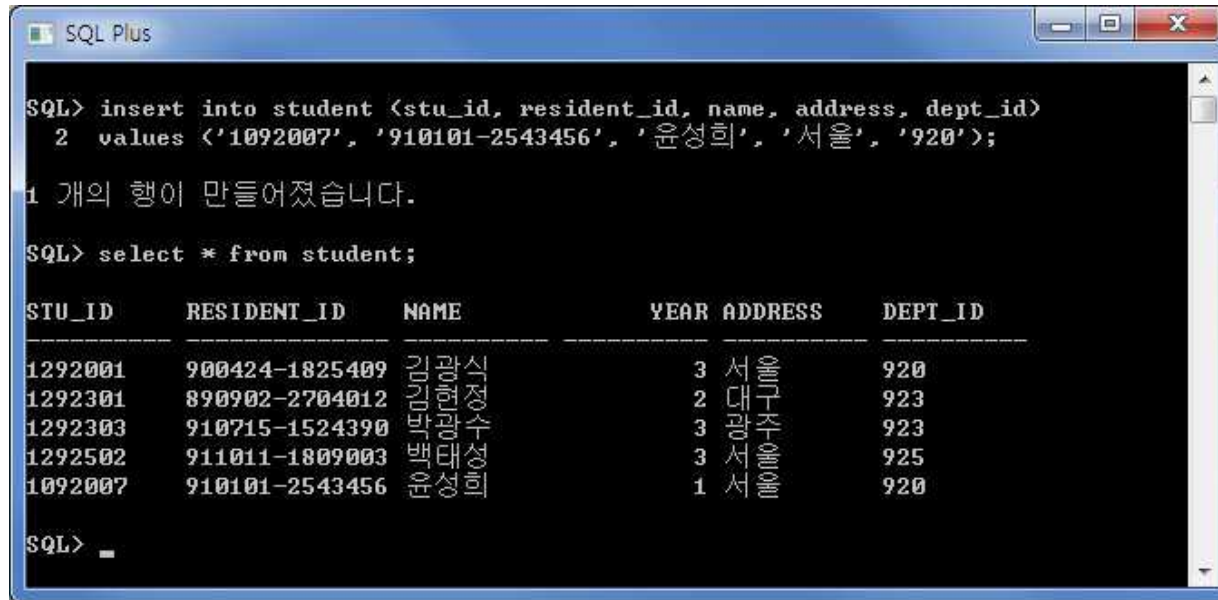
- ▶ 다음의 레코드 삽입하면 year에 1이 자동적으로 삽입

(질의 57)

**insert into**

student (stu\_id, resident\_id, name, address, dept\_id)

**values** ('1092007', '910101-2543456', '윤성희', '서울', '920')



```
SQL> insert into student (stu_id, resident_id, name, address, dept_id)
  2 values ('1092007', '910101-2543456', '윤성희', '서울', '920');
```

1 개의 행이 만들어졌습니다.

```
SQL> select * from student;
```

STU_ID	RESIDENT_ID	NAME	YEAR	ADDRESS	DEPT_ID
1292001	900424-1825409	김광식	3	서울	920
1292301	890902-2704012	김현정	2	대구	923
1292303	910715-1524390	박광수	3	광주	923
1292502	911011-1809003	백태성	3	서울	925
1092007	910101-2543456	윤성희	1	서울	920

```
SQL> _
```

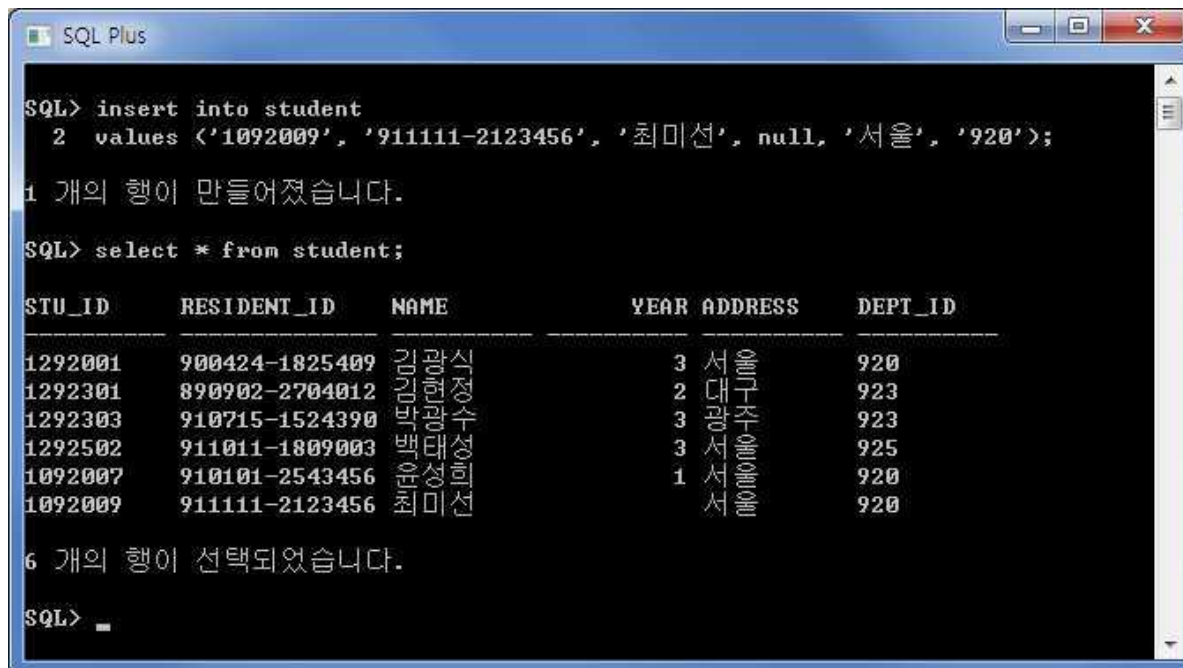
# DEFAULT

- ▶ 의도적인 null 삽입에는 적용되지 않음

(질의 58)

**insert into student values**

('1092009', '911111-2123456', '최미선', null, '서울', '920')



```
SQL> insert into student
  2  values (<'1092009', '911111-2123456', '최미선', null, '서울', '920'>);

1 개의 행이 만들어졌습니다.

SQL> select * from student;
```

STU_ID	RESIDENT_ID	NAME	YEAR	ADDRESS	DEPT_ID
1292001	900424-1825409	김광식	3	서울	920
1292301	890902-2704012	김현정	2	대구	923
1292303	910715-1524390	박광수	3	광주	923
1292502	911011-1809003	백태성	3	서울	925
1092007	910101-2543456	윤성희	1	서울	920
1092009	911111-2123456	최미선		서울	920

```
6 개의 행이 선택되었습니다.

SQL> _
```

# 오라클에서의 사용자 권한제어

---

- ▶ 오라클의 권한 종류
  - ▶ 시스템 권한(system privileges)
    - ▶ 사용자의 생성, 테이블이나 테이블 스페이스의 생성 등과 같이 주로 시스템의 자원을 관리할 수 있는 권한
    - ▶ 데이터베이스 관리자 계정(sys, system)은 모든 시스템 권한을 가짐
  - ▶ 객체 권한(object privileges)
    - ▶ 해당 객체에 대해 select, insert, delete, update 등과 같은 DML을 실행할 수 있는 권한
    - ▶ 객체
      - 테이블이나 뷰 또는 사용자 정의 함수 등

# 오라클의 대표적인 시스템 권한

대상	시스템 권한	내용
사용자	create user	사용자 계정을 생성할 수 있는 권한
	drop user	사용자 계정을 삭제하는 권한
세션	create session	데이터베이스의 접속(로그인) 권한
테이블	create table	테이블을 생성할 수 있는 권한
	drop any table	임의의 테이블을 삭제하는 권한
	select any table	임의의 테이블에 대한 읽기 권한
뷰	create view	뷰를 생성할 수 있는 권한
	drop any view	임의의 뷰에 대한 삭제 권한
테이블 스페이스	create tablespace	테이블 스페이스를 생성하는 권한
	drop tablespace	테이블 스페이스에 대한 삭제 권한
롤	create role	롤을 생성할 수 있는 권한
	drop any role	임의의 롤을 삭제하는 권한

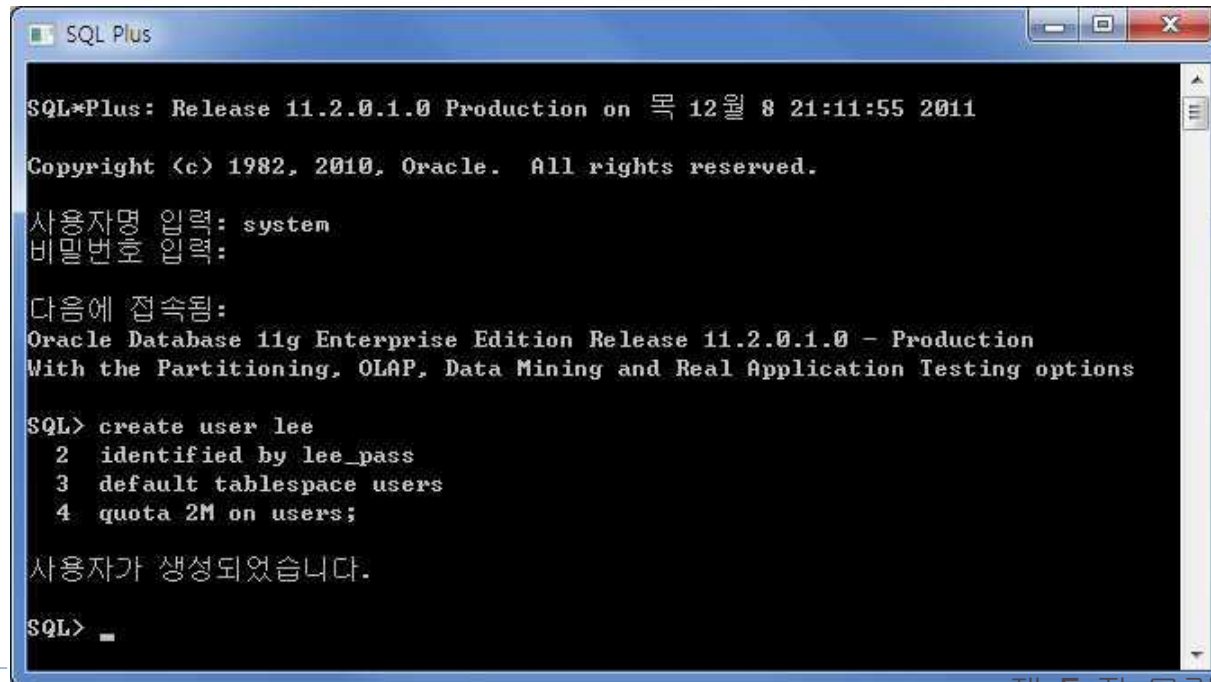
# 사용자 계정 생성과 GRANT

## ▶ 사용자 계정 생성 예

- ▶ system 계정에서 실행
- ▶ lee : 사용자 ID
- ▶ lee\_pass : 비밀번호
- ▶ users : default tablespace, users는 오라클을 설치시 기본적으로 생성

(질의 59)

```
create user    lee
identified by  lee_pass
default tablespace    users
quota 2M        on users
```



```
SQL*Plus: Release 11.2.0.1.0 Production on 목 12월 8 21:11:55 2011
Copyright (c) 1982, 2010, Oracle. All rights reserved.

사용자명 입력: system
비밀번호 입력:

다음에 접속됨:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

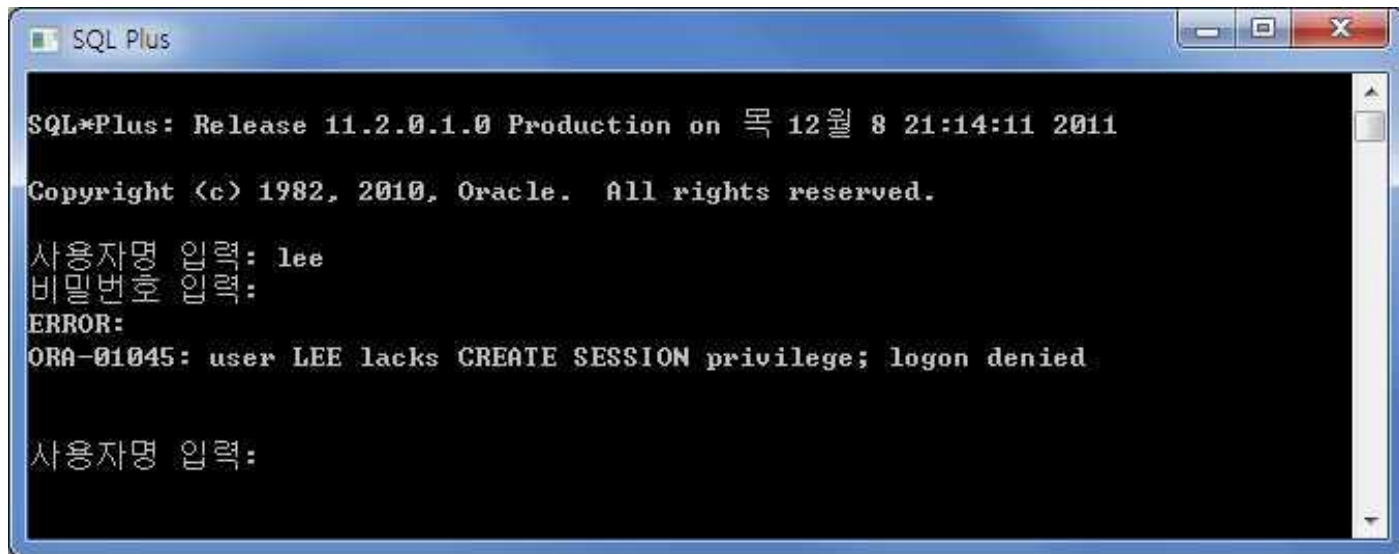
SQL> create user lee
      2 identified by lee_pass
      3 default tablespace users
      4 quota 2M on users;

사용자가 생성되었습니다.

SQL> _
```

# 사용자 계정 생성과 GRANT

- ▶ 사용자 생성 후 권한을 부여해야 함.
- ▶ 권한 없이 로그인할 경우의 예



```
SQL*Plus
SQL*Plus: Release 11.2.0.1.0 Production on 목 12월 8 21:14:11 2011
Copyright (c) 1982, 2010, Oracle. All rights reserved.
사용자명 입력: lee
비밀번호 입력:
ERROR:
ORA-01045: user LEE lacks CREATE SESSION privilege; logon denied
사용자명 입력:
```

# 사용자 계정 생성과 GRANT

- ▶ lee에게 create session과 create table 권한을 부여(system 계정에서)

(질의 60)

```
grant create session to lee  
grant create table to lee
```

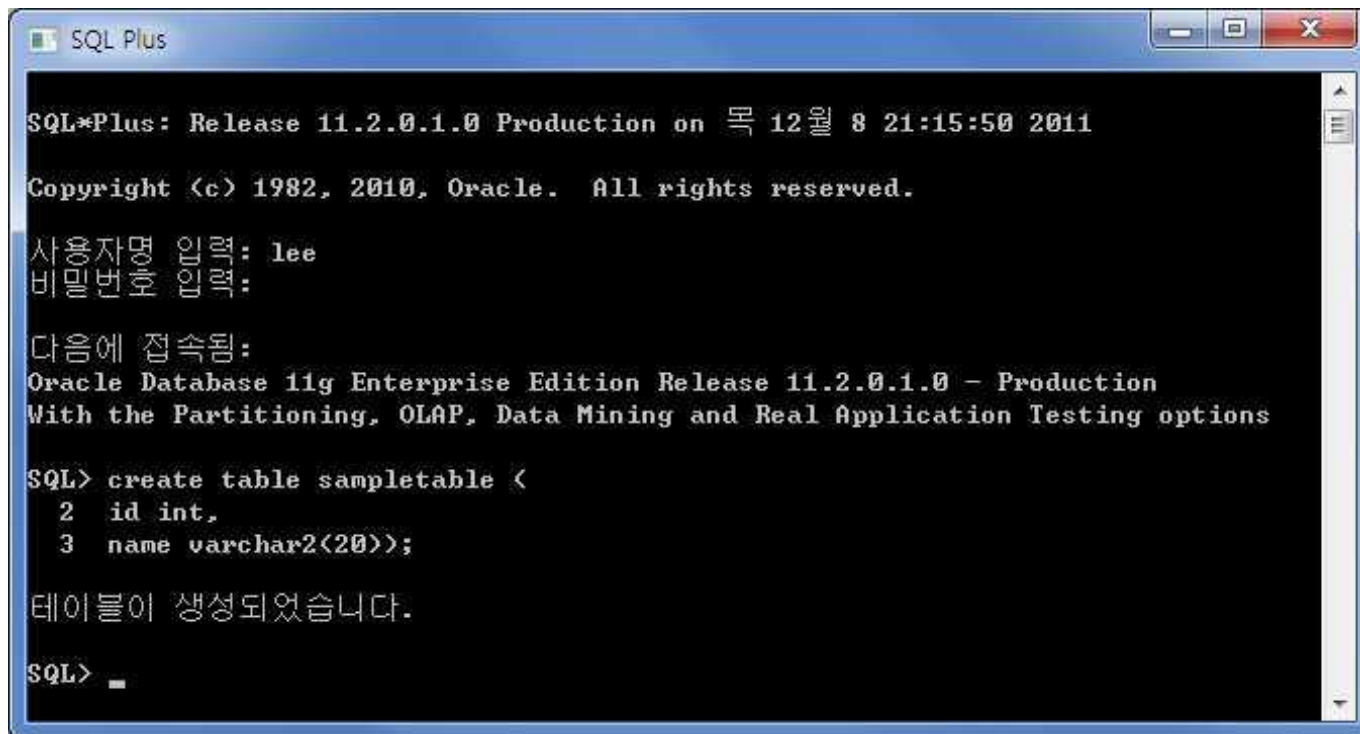


```
SQL> grant create session to lee;  
권한이 부여되었습니다.  
  
SQL> grant create table to lee;  
권한이 부여되었습니다.  
  
SQL>
```



# 사용자 계정 생성과 GRANT

- ▶ 사용자 lee는 로그인과 테이블 생성에 대한 권한을 부여받았으므로 로그인이 가능하며 동시에 테이블 생성도 가능



```
SQL*Plus: Release 11.2.0.1.0 Production on 목 12월 8 21:15:50 2011
Copyright (c) 1982, 2010, Oracle. All rights reserved.

사용자명 입력: lee
비밀번호 입력:

다음에 접속됨:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> create table sampletable (
  2  id int,
  3  name varchar2(20));

테이블이 생성되었습니다.

SQL> _
```

# 테이블에 대한 권한의 부여와 회수 예

---

## ▶ 사용자 계정과 테이블을 다음의 과정으로 준비

- ① system 계정으로 로그인한 후에 다음과 같이 kim이란 계정을 추가로 생성하고 권한을 부여

(질의 61)

```
create user    kim  
identified by kim_pass  
default tablespace users  
quota 2M on  users  
  
grant create session to kim  
  
grant create table to kim
```

- ② lee로 로그인을 하여 student 테이블과 department 테이블을 생성하고, 레코드들을 삽입

# 테이블에 대한 권한의 부여와 회수 예

- ▶ lee로 로그인 한 후에 kim에게 student 테이블에 대한 select 권한을 다음과 같이 부여

(질의 62)

```
grant select on student to kim
```

- ▶ kim으로 로그인하여 student 테이블에 대한 select연산을 실행

(질의 63)

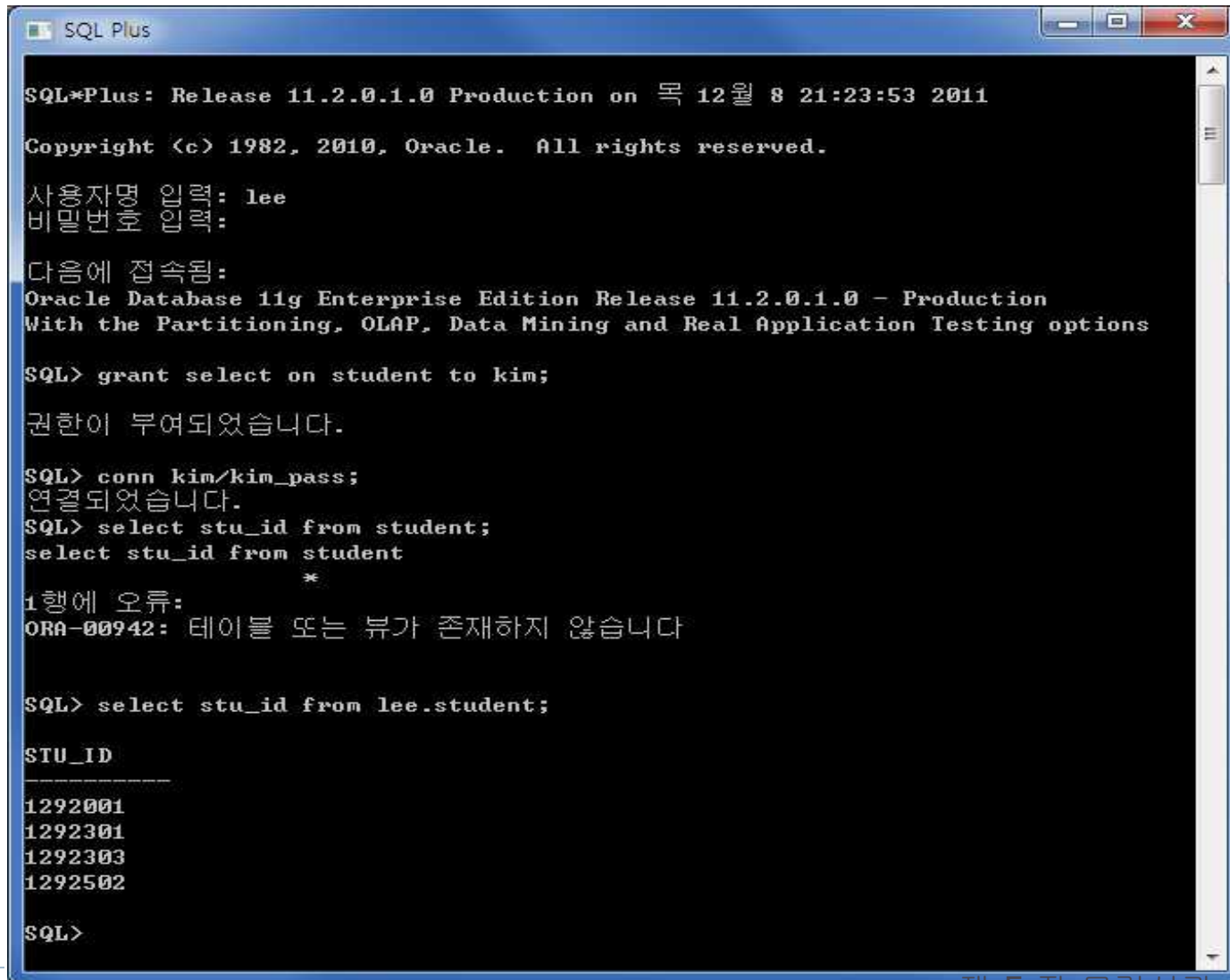
```
conn      kim/kim_pass
```

(질의 64)

```
select    stu_id  
from      lee.student
```

- ▶ 다른 사용자가 소유한 테이블에 대해 테이블 이름을 명시할 때는 '소유자 명.테이블명'과 같이 테이블 이름 앞에 소유자 명까지 기입

# 테이블에 대한 권한의 부여와 회수 예



```
SQL*Plus: Release 11.2.0.1.0 Production on 목 12월 8 21:23:53 2011

Copyright (c) 1982, 2010, Oracle. All rights reserved.

사용자명 입력: lee
비밀번호 입력:

다음에 접속됨:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> grant select on student to kim;

권한이 부여되었습니다.

SQL> conn kim/kim_pass;
연결되었습니다.
SQL> select stu_id from student;
select stu_id from student
                *
1 행에 오류:
ORA-00942: 테이블 또는 뷰가 존재하지 않습니다

SQL> select stu_id from lee.student;

STU_ID
-----
1292001
1292301
1292303
1292502

SQL>
```

# WITH GRANT OPTION의 실행 예

- ▶ chang이라는 계정을 생성했다고 가정
- ▶ lee 계정으로 로그인한 후에 kim에게 student 테이블에 대한 select 권한을 부여

(질의 65)

**grant select on student to kim with grant option**

(lee가 실행)

- ▶ kim으로 로그인하여 chang에게 전파

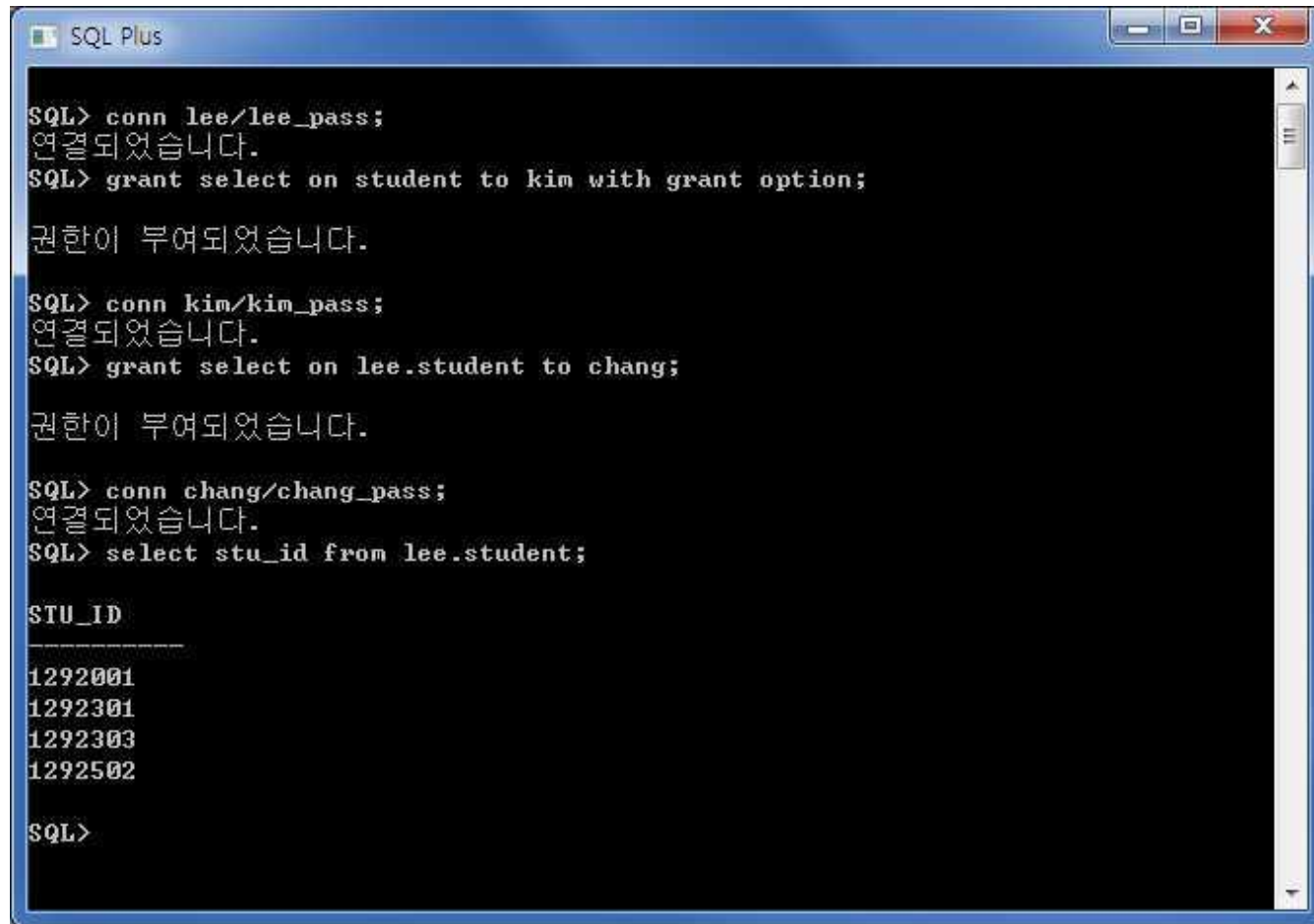
(질의 66)

**grant select on lee.student to chang**

(kim 실행)

- ▶ chang으로 로그인하여 student 테이블에 대한 select 명령 실행 가능

# WITH GRANT OPTION의 실행 예



```
SQL> conn lee/lee_pass;
연결되었습니다.
SQL> grant select on student to kim with grant option;

권한이 부여되었습니다.

SQL> conn kim/kim_pass;
연결되었습니다.
SQL> grant select on lee.student to chang;

권한이 부여되었습니다.

SQL> conn chang/chang_pass;
연결되었습니다.
SQL> select stu_id from lee.student;

STU_ID
-----
1292001
1292301
1292303
1292502

SQL>
```

# REVOKE

---

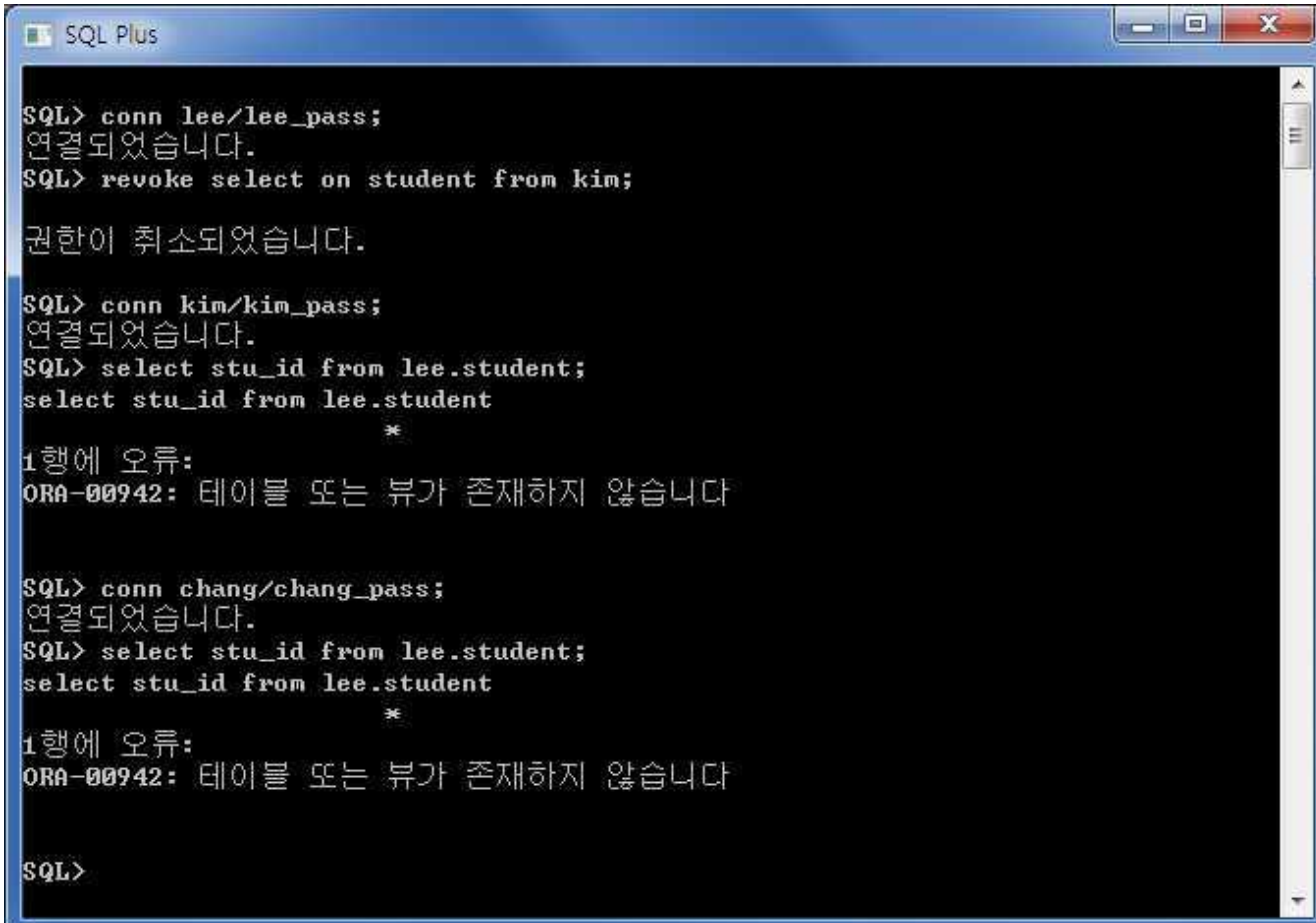
- ▶ lee가 kim에게 부여된 revoke문을 이용하여 권한을 회수

(질의 67)

**revoke select on student from** kim

- ▶ 간접적으로 권한을 부여 받은 chang에게도 자동적으로 권한을 회수함

# REVOKE



```
SQL> conn lee/lee_pass;
연결되었습니다.
SQL> revoke select on student from kim;

권한이 취소되었습니다.

SQL> conn kim/kim_pass;
연결되었습니다.
SQL> select stu_id from lee.student;
select stu_id from lee.student
          *
1행에 오류:
ORA-00942: 테이블 또는 뷰가 존재하지 않습니다

SQL> conn chang/chang_pass;
연결되었습니다.
SQL> select stu_id from lee.student;
select stu_id from lee.student
          *
1행에 오류:
ORA-00942: 테이블 또는 뷰가 존재하지 않습니다

SQL>
```



## 참고: with grant option과 with admin option

- ▶ 오라클에서는 with grant option과 with admin option을 모두 지원함
- ▶ with grant option
  - ▶ 객체 권한을 다른 사용자에게 전파할 때 사용
- ▶ with admin option
  - ▶ 시스템 권한을 전파할 때 사용
  - ▶ 예)

(질의 68)

**grant create table to kim with admin option**

# 롤의 생성과 관리

- ▶ 오라클의 기본적인 롤 종류
  - ▶ 오라클에서 권한의 종류는 200여 개
  - ▶ 롤의 사용이 필수적
  - ▶ 오라클에서 사전에 정의된 기본 롤

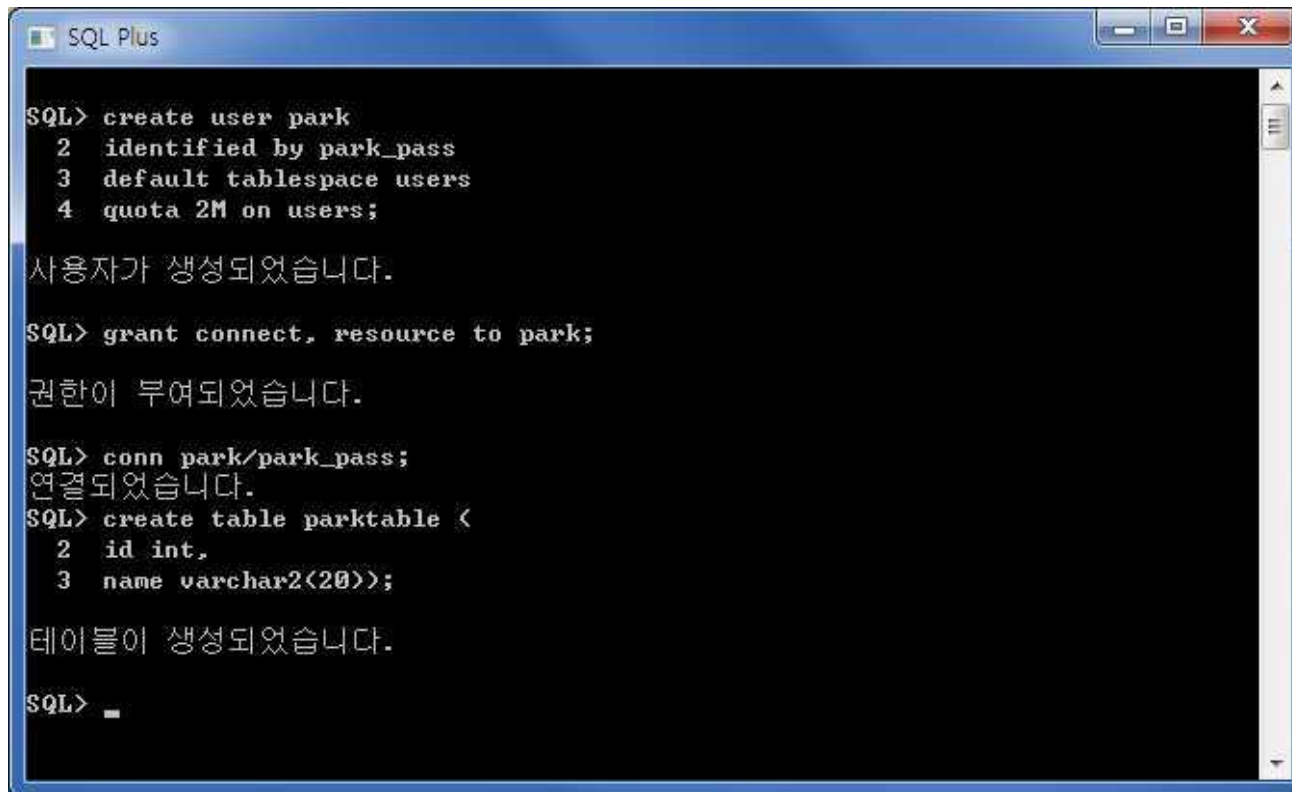
롤 이름	권한 종류	설명
<b>connect</b>	<b>create session</b>	사용자가 데이터베이스에 접속할 수 있는 기본적인 시스템 권한
<b>resource</b>	<b>create table</b> <b>create trigger</b> <b>create procedure</b> <b>create cluster</b> <b>create sequence</b>	사용자가 객체를 생성하고 관리할 수 있는 시스템 권한
<b>DBA</b>	데이터베이스 관리자 권한	시스템 및 객체 관리에 대한 모든 권한

# 롤의 생성과 관리

- ▶ park에게 connect와 resource 권한을 부여

(질의 69)

**grant** connect, resource **to** park



```
SQL> create user park
  2  identified by park_pass
  3  default tablespace users
  4  quota 2M on users;

사용자가 생성되었습니다.

SQL> grant connect, resource to park;

권한이 부여되었습니다.

SQL> conn park/park_pass;
연결되었습니다.

SQL> create table parktable (
  2  id int,
  3  name varchar2(20));

테이블이 생성되었습니다.

SQL> _
```

# 롤의 생성과 사용자 배정

---

- ▶ 롤 생성
  - ▶ create role문은 데이터베이스 관리자 또는 데이터베이스 관리자로부터 롤을 생성하는 권한을 부여 받은 사용자만이 실행
- ▶ student테이블의 소유자가 lee라고 가정
  - ▶ system 계정에서 다음을 실행

(질의 70)

```
create role stu_role
```

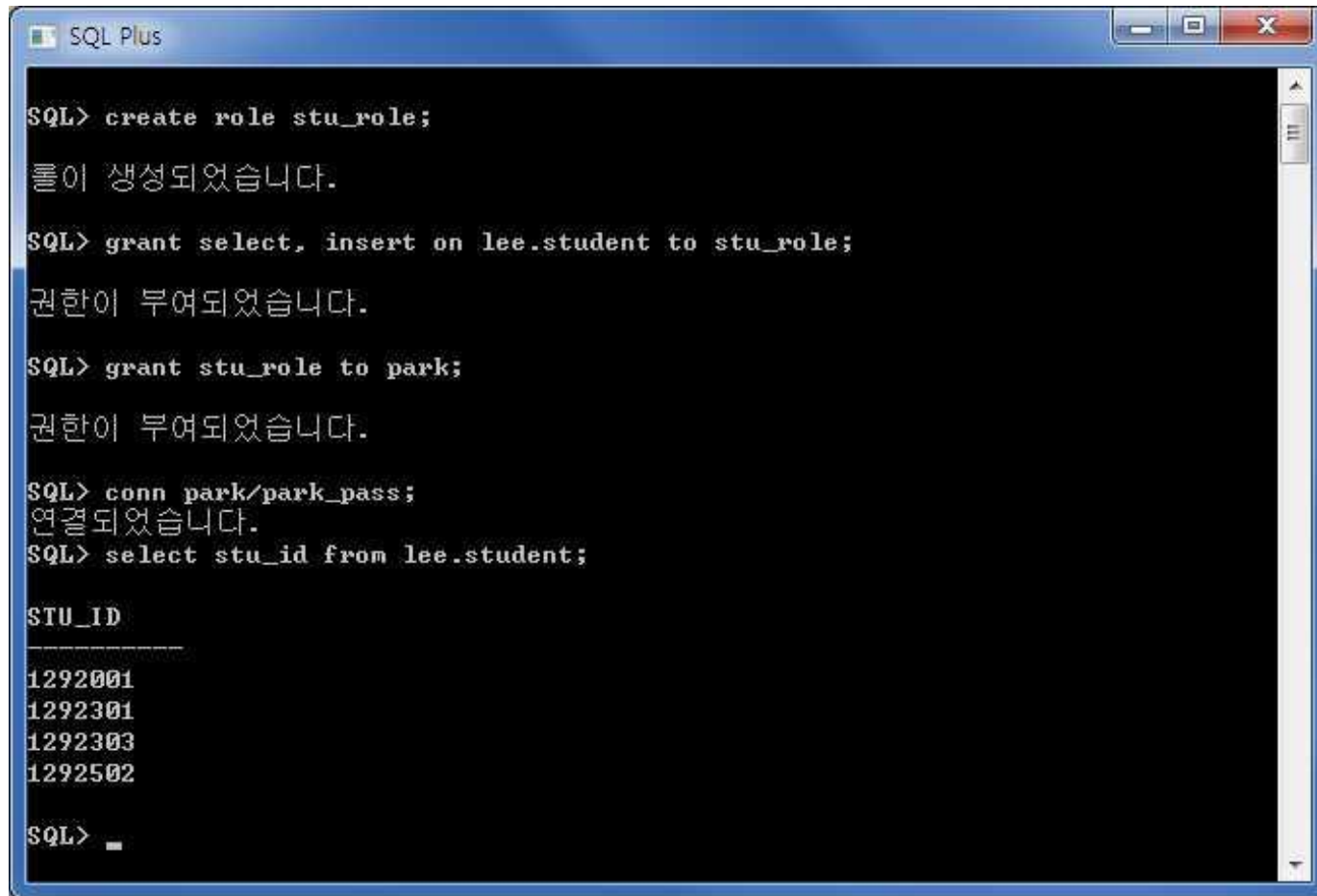
(질의 71)

```
grant select, insert on lee.student to stu_role
```

(질의 72)

```
grant stu_role to park
```

# 롤의 생성과 사용자 배정



```
SQL> create role stu_role;
롤이 생성되었습니다.

SQL> grant select, insert on lee.student to stu_role;
권한이 부여되었습니다.

SQL> grant stu_role to park;
권한이 부여되었습니다.

SQL> conn park/park_pass;
연결되었습니다.

SQL> select stu_id from lee.student;

STU_ID
-----
1292001
1292301
1292303
1292502

SQL> _
```

# 롤의 생성과 사용자 배정

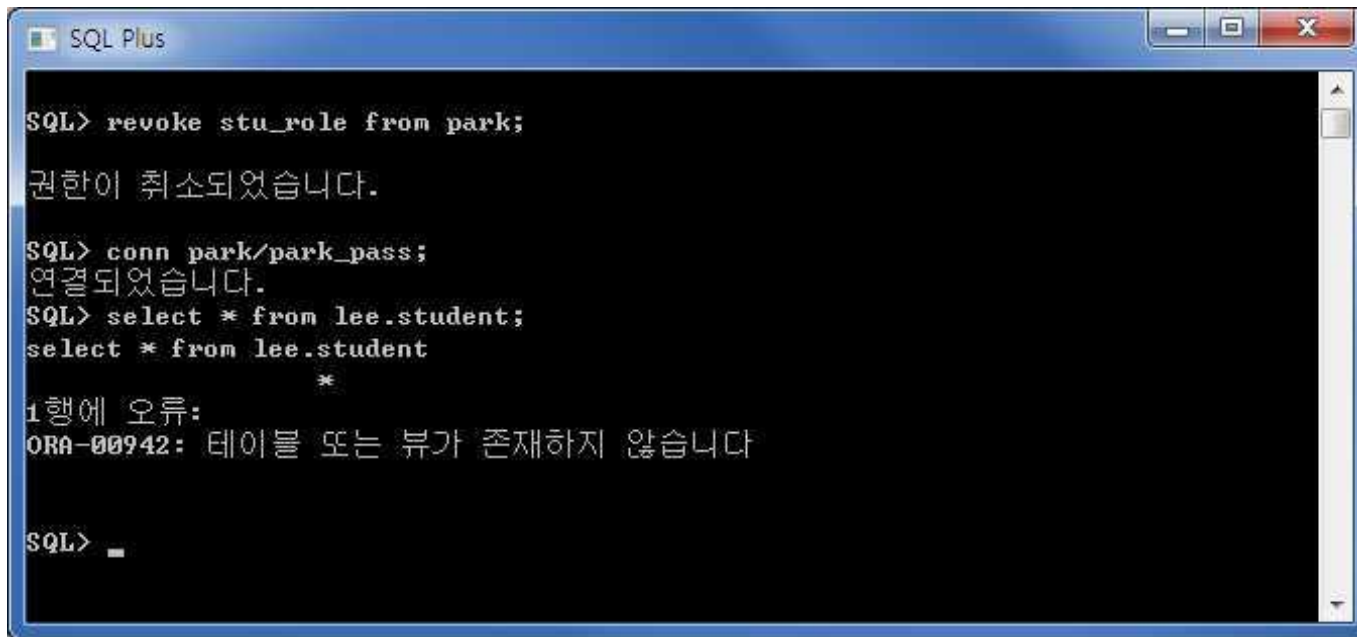
## ▶ 롤에서 사용자 배제

- ▶ 예) stu\_role에 배정된 park을 다음과 같이 revoke문으로 배제

(질의 73)

**revoke** stu\_role **from** park

(system 계정)



```
SQL> revoke stu_role from park;
권한이 취소되었습니다.

SQL> conn park/park_pass;
연결되었습니다.
SQL> select * from lee.student;
select * from lee.student
                *
1행에 오류:
ORA-00942: 테이블 또는 뷰가 존재하지 않습니다

SQL> _
```