

The background is a complex network of thin grey lines connecting various sized nodes. The nodes are colored in dark blue, light blue, and grey. Some nodes are simple dots, while others are larger circles with concentric rings. A large dark blue circle with a white center is prominent at the top center. A light blue circle with a white center is at the bottom left. A grey circle with a white center is at the top right. A large black rectangle is positioned in the lower right, containing the word 'ORACLE' in white capital letters. A thin light blue horizontal line is at the bottom of the black rectangle.

ORACLE

데이터베이스 생성(오라클)

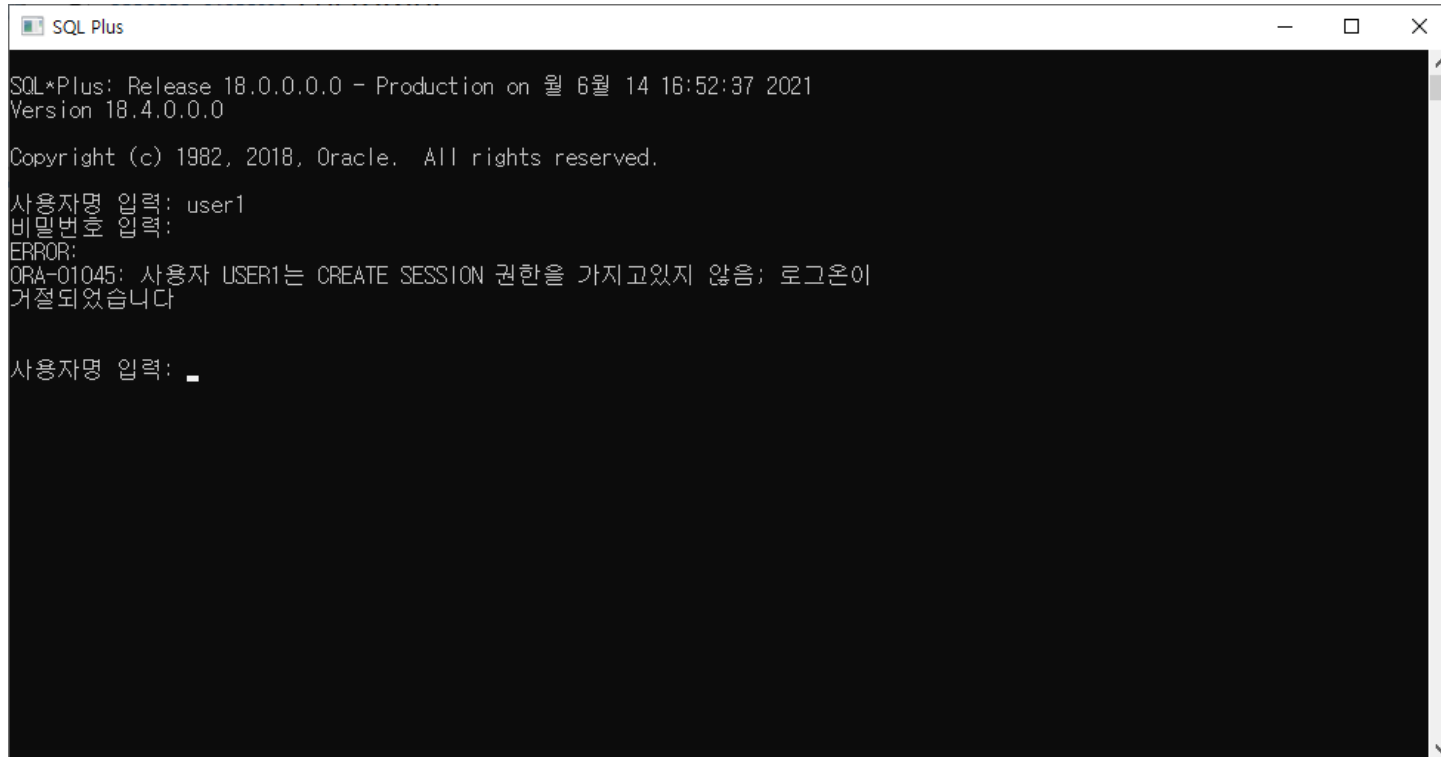
오라클은 기본적으로 사용자 추가 = 데이터베이스 추가

엄밀히 따지자면 스키마가 추가되는 것

SYSTEM 데이터베이스는 오라클 시스템을 운영하기 위한 전용 테이블

스크립트 허용

```
alter session set "_ORACLE_SCRIPT"=true;  
create user user1 identified by 1111;
```

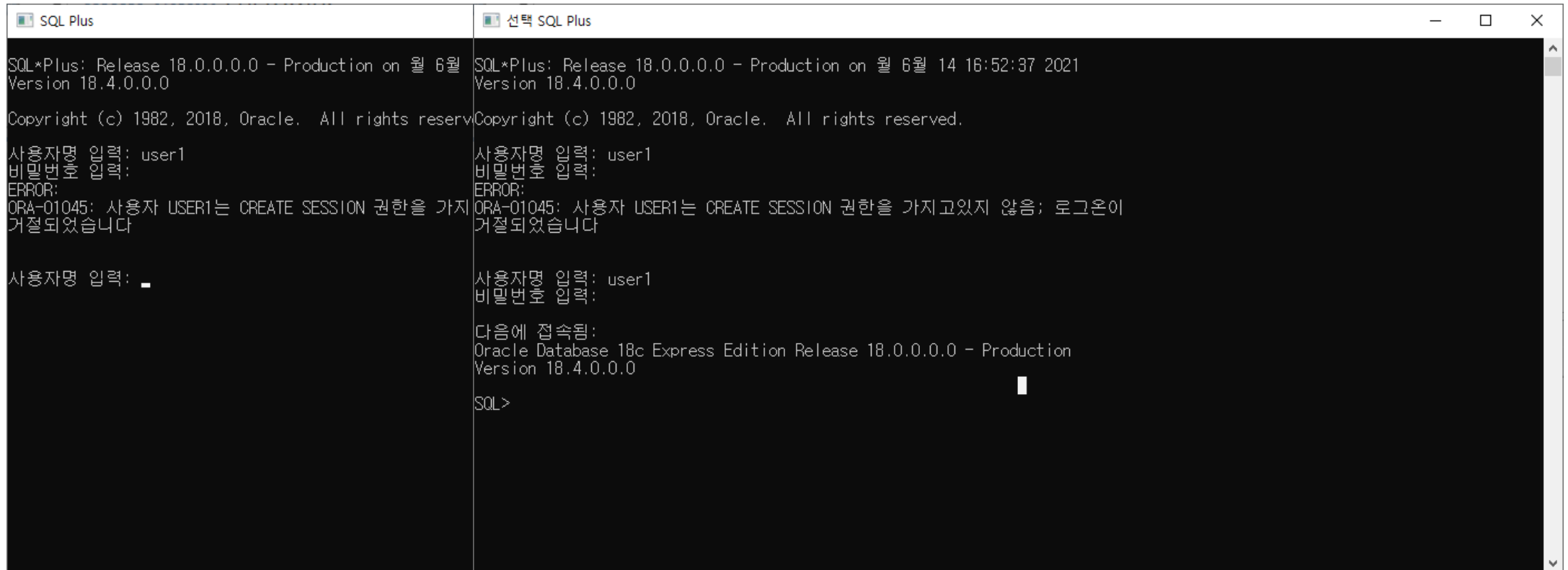


The screenshot shows a terminal window titled "SQL Plus". The text inside the window is as follows:

```
SQL*Plus: Release 18.0.0.0.0 - Production on 월 6월 14 16:52:37 2021  
Version 18.4.0.0.0  
  
Copyright (c) 1982, 2018, Oracle. All rights reserved.  
  
사용자명 입력: user1  
비밀번호 입력:  
ERROR:  
ORA-01045: 사용자 USER1는 CREATE SESSION 권한을 가지고있지 않음; 로그온이  
거절되었습니다  
  
사용자명 입력: _
```

테이블 생성 허용

grant create session to user1;



The image shows two side-by-side terminal windows from SQL Plus. The left window, titled 'SQL Plus', shows a login attempt for 'user1' which fails with the error 'ORA-01045: 사용자 USER1는 CREATE SESSION 권한을 가지 거절되었습니다' (ORA-01045: user USER1 does not have the CREATE SESSION privilege; access denied). The right window, titled '선택 SQL Plus', shows the same login attempt, but then displays a successful connection message: '다음에 접속됨: Oracle Database 18c Express Edition Release 18.0.0.0.0 - Production Version 18.4.0.0.0' followed by the 'SQL>' prompt.

```
SQL*Plus: Release 18.0.0.0.0 - Production on 월 6월
Version 18.4.0.0.0

Copyright (c) 1982, 2018, Oracle. All rights reserved.

사용자명 입력: user1
비밀번호 입력:
ERROR:
ORA-01045: 사용자 USER1는 CREATE SESSION 권한을 가지
거절되었습니다

사용자명 입력: _
```

```
선택 SQL Plus

SQL*Plus: Release 18.0.0.0.0 - Production on 월 6월 14 16:52:37 2021
Version 18.4.0.0.0

Copyright (c) 1982, 2018, Oracle. All rights reserved.

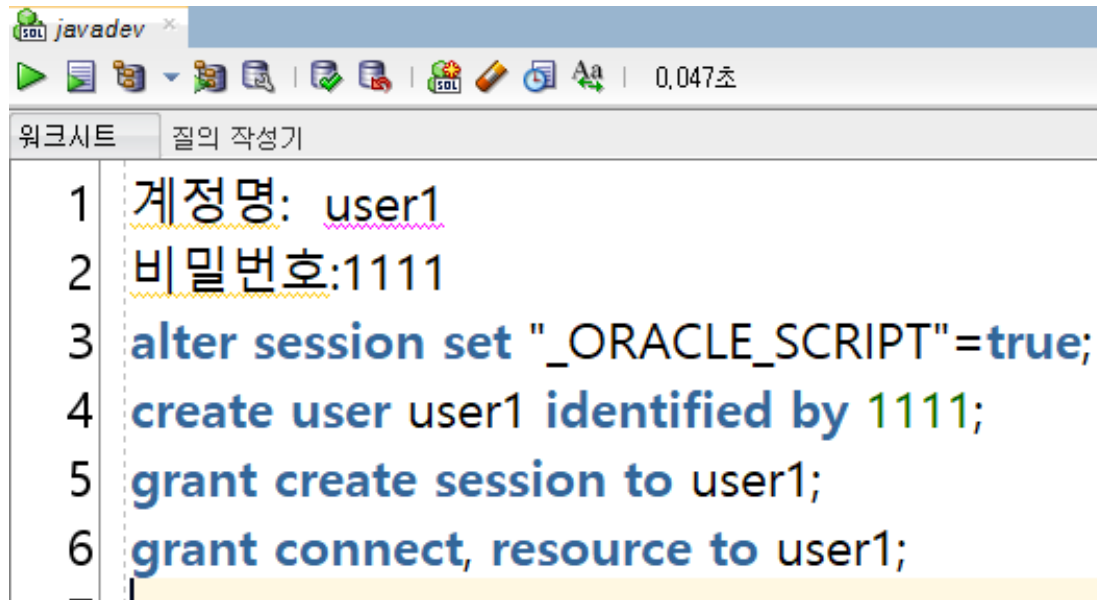
사용자명 입력: user1
비밀번호 입력:
ERROR:
ORA-01045: 사용자 USER1는 CREATE SESSION 권한을 가지고있지 않음; 로그인
거절되었습니다

사용자명 입력: user1
비밀번호 입력:

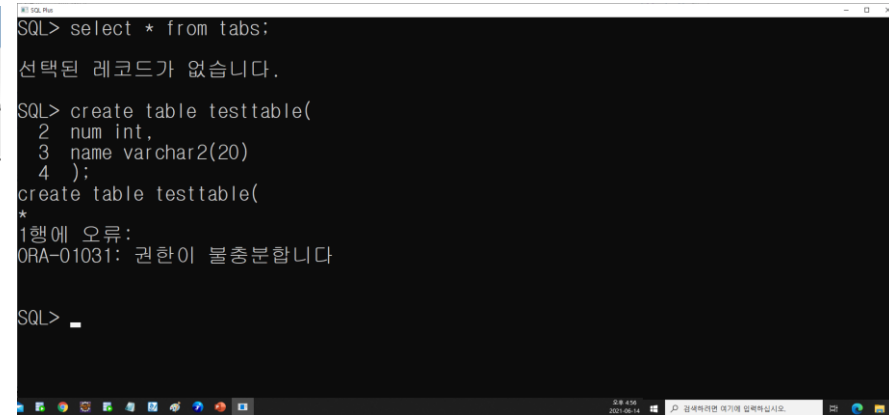
다음에 접속됨:
Oracle Database 18c Express Edition Release 18.0.0.0.0 - Production
Version 18.4.0.0.0

SQL>
```

계정 생성



```
1 계정명: user1
2 비밀번호:1111
3 alter session set "_ORACLE_SCRIPT"=true;
4 create user user1 identified by 1111;
5 grant create session to user1;
6 grant connect, resource to user1;
```

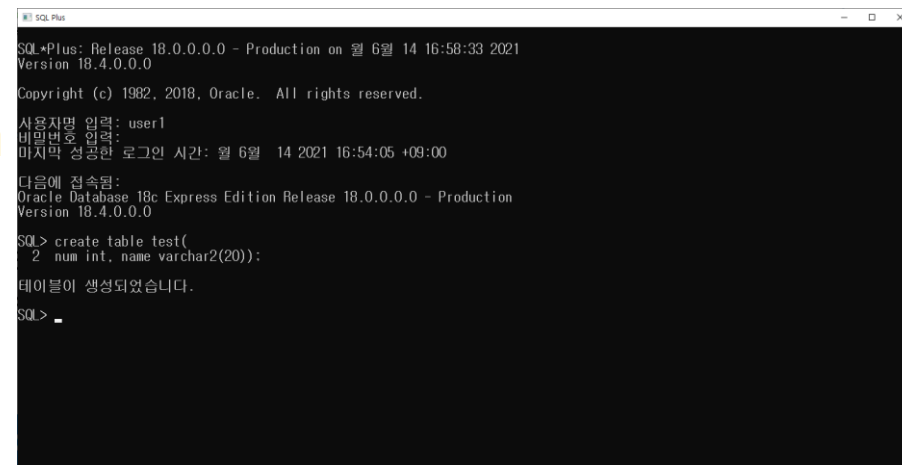


```
SQL> select * from tabs;

선택된 레코드가 없습니다.

SQL> create table testtable(
  2 num int,
  3 name varchar2(20)
  4 );
create table testtable(
*
1행에 오류:
ORA-01031: 권한이 불충분합니다

SQL> _
```



```
SQL>Plus: Release 18.0.0.0.0 - Production on 월 6월 14 16:58:33 2021
Version 18.4.0.0.0

Copyright (c) 1982, 2018, Oracle. All rights reserved.

사용자명 입력: user1
비밀번호 입력:
마지막 성공한 로그인 시간: 월 6월 14 2021 16:54:05 +09:00

다음에 접속함:
Oracle Database 18c Express Edition Release 18.0.0.0.0 - Production
Version 18.4.0.0.0

SQL> create table test(
  2 num int, name varchar2(20));

테이블이 생성되었습니다.

SQL> _
```

테이블 공간 사용 허용

```
SQL> insert into test values(1,'hong');
insert into test values(1,'hong')
      *
```

1행에 오류:
ORA-01950: 테이블스페이스 'USERS'에 대한 권한이 없습니다.

```
SQL> _
```

```
1 계정명: user1
2 비밀번호:1111
3 alter session set "_ORACLE_SCRIPT"=true;
4 create user user1 identified by 1111;
5 grant create session to user1;
6 grant connect, resource to user1;
7 alter user user1 default tablespace users quota unlimited on users;
```

오라클 접속

새로 만들기/데이터베이스 접속 선택

| 접속 이름 | 접속 세부정보 |
|-----------|--------------------|
| javadev | system@//local... |
| 로컬-DWIT | dwit@//localhos... |
| 로컬-HR | HR@//localhost:... |
| 로컬-Shop | Shop@//localho... |
| 로컬-sqlDB | sqlDB@//localh... |
| 로컬-SYSTEM | SYSTEM@//loc... |

접속 이름(N) user1db
사용자 이름(U) user1
비밀번호(P)

☒ 비밀번호 저장(Y) ☐ 접속 색상

Oracle

접속 유형(Y) 기본 기본값

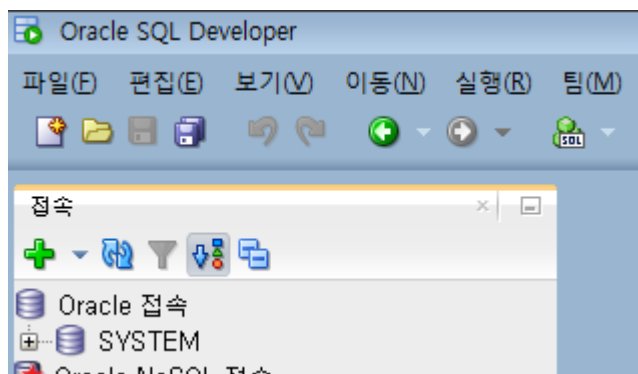
호스트 이름(A) localhost
포트(B) 1521
☒ SID(I) xe
☐ 서비스 이름(E)

☐ OS 인증 ☐ Kerberos 인증

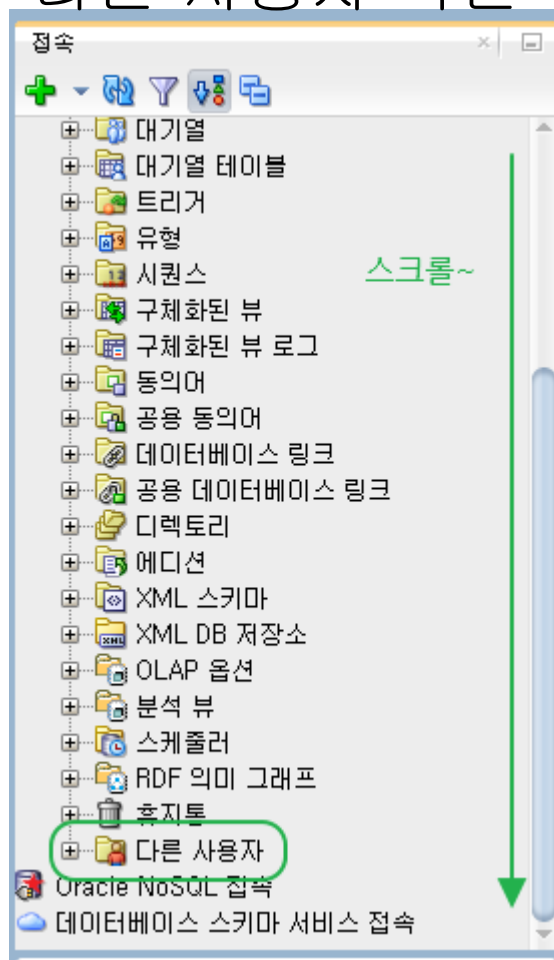
상태: 성공

사용자추가

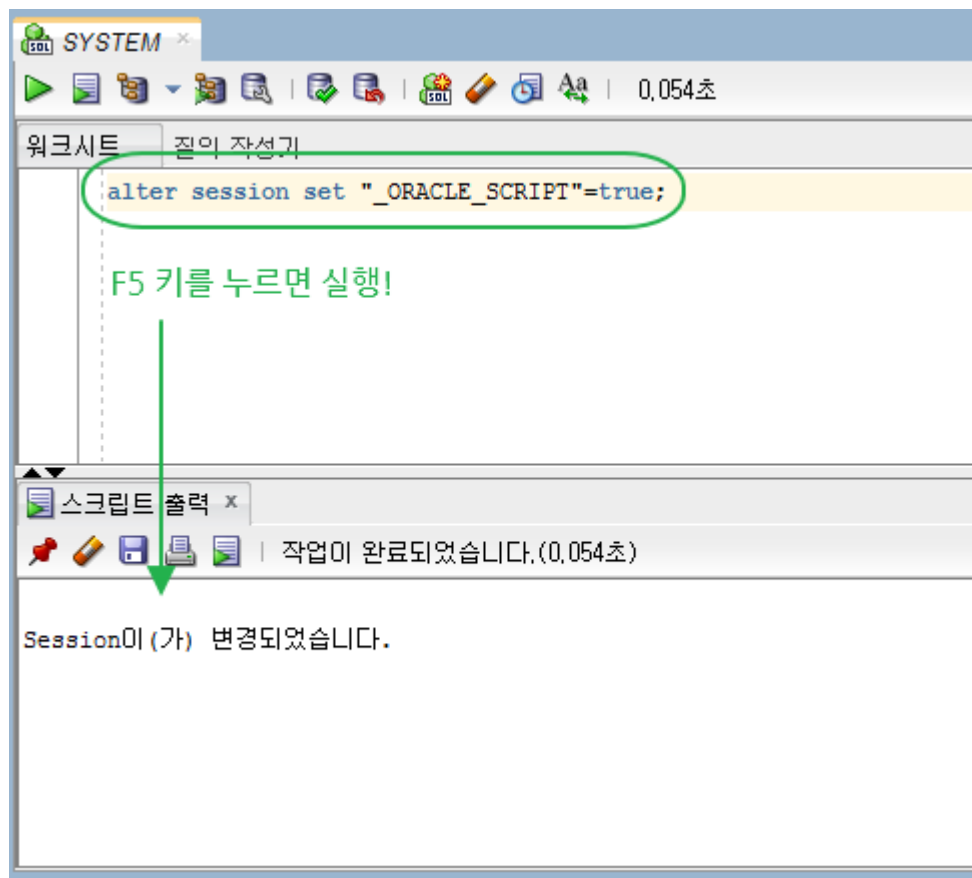
System으로
데이터 베이스 접속



다른 사용자 확인



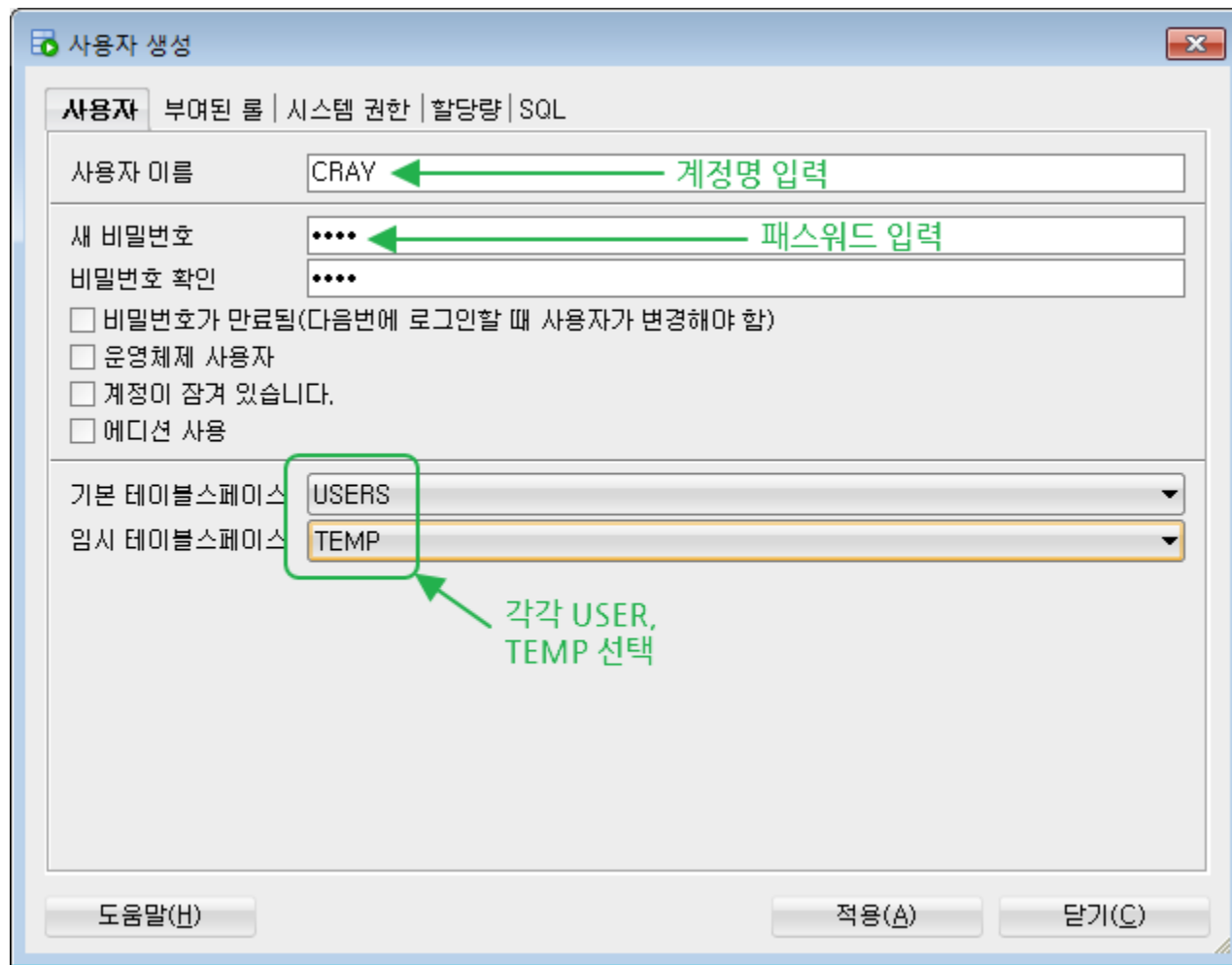
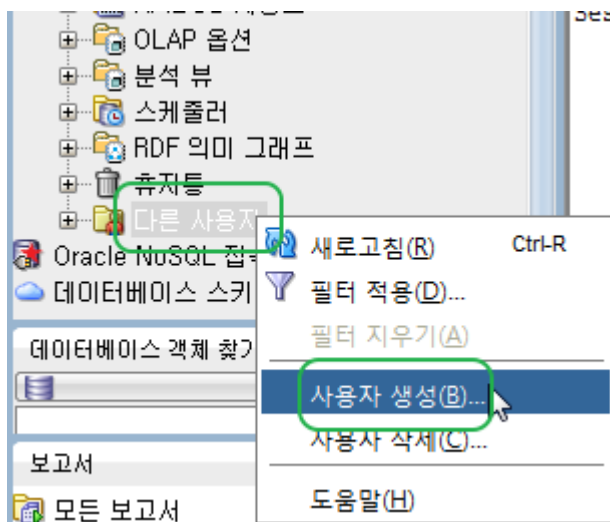
사용자 추가 허용 설정 변경



`alter session set "_ORACLE_SCRIPT"=true;`

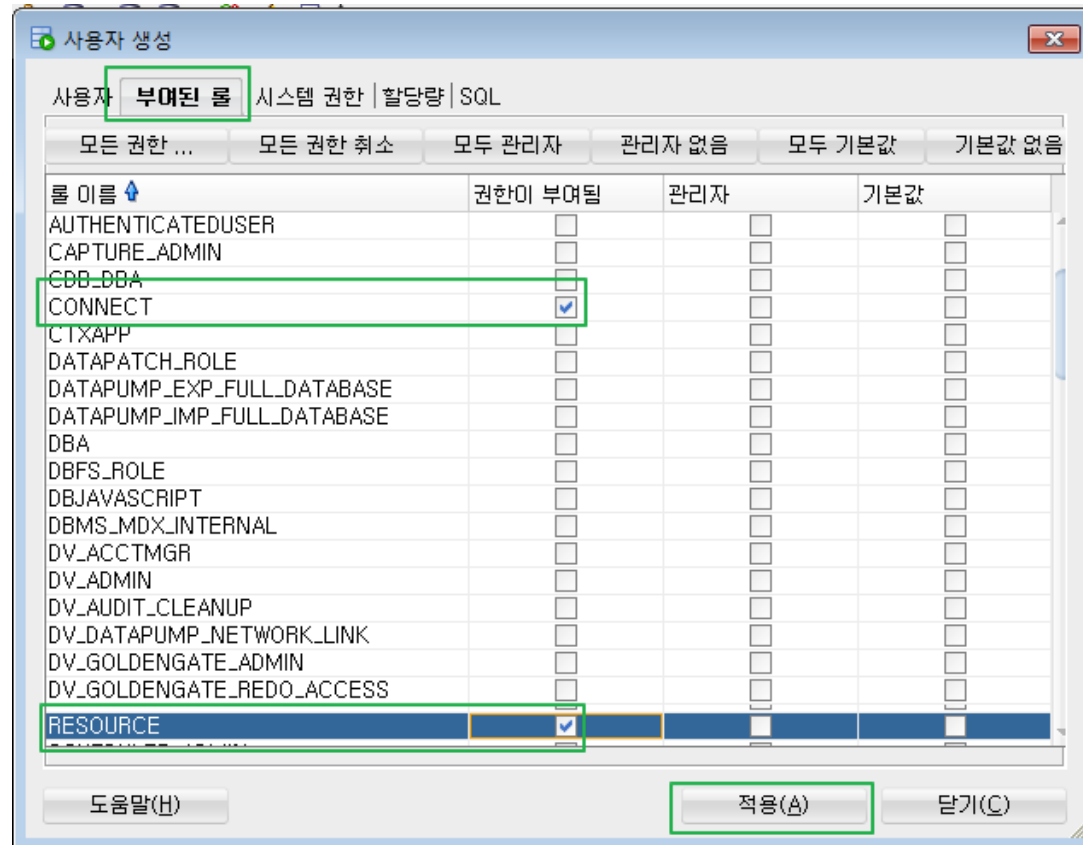
사용자추가

사용자 생성

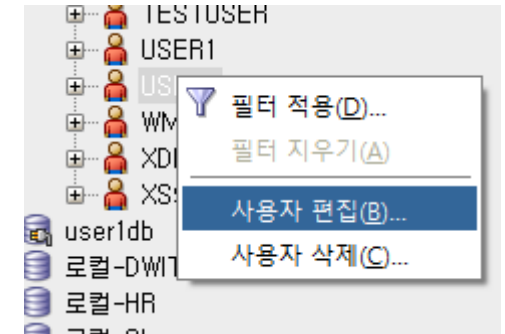
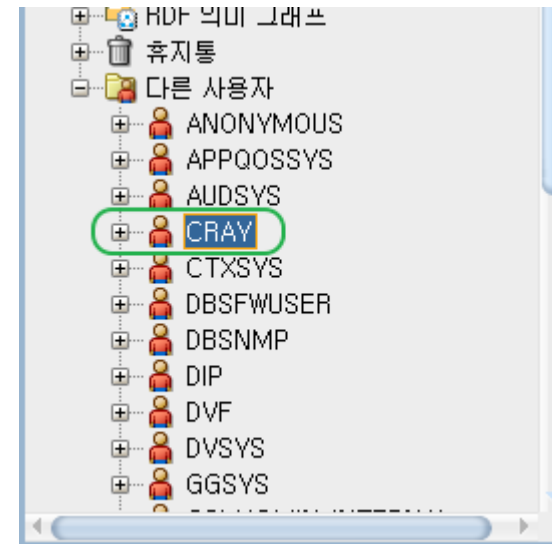


사용자추가

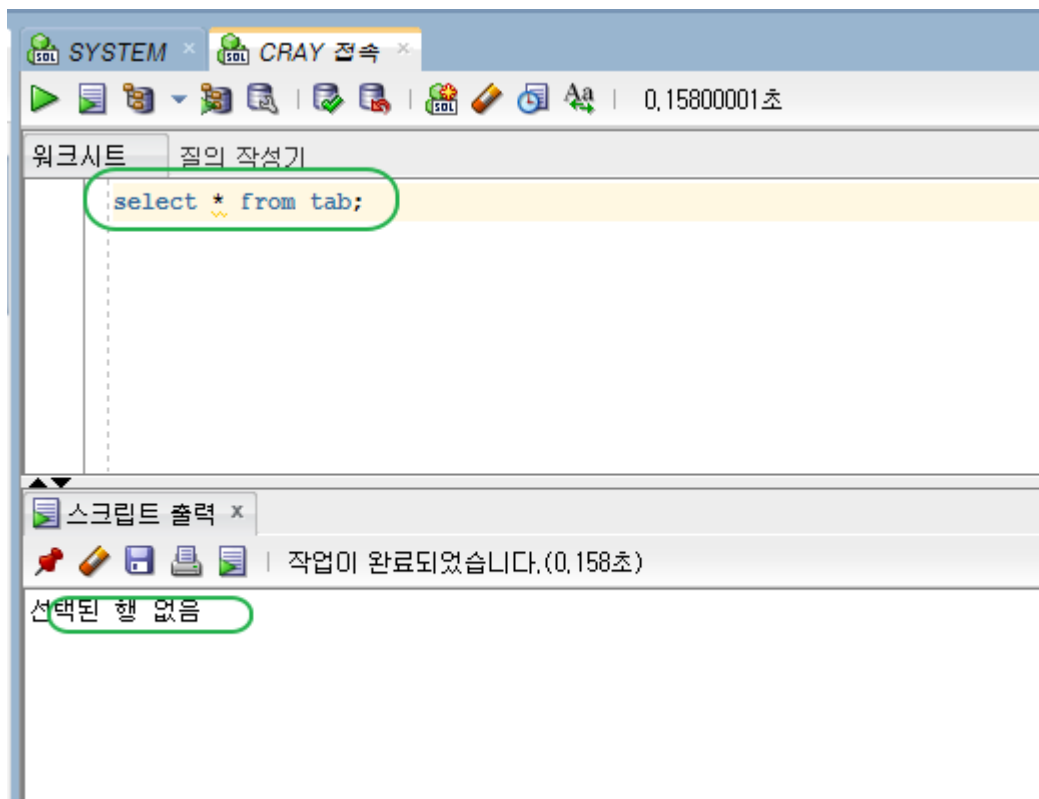
접속 및 리소스 사용 권한 부여



사용자 등록 확인



사용자 접속 확인



VIEW, INDEX, PROCEDURE, FUNCTION, TRIGGER

create view 뷰이름
as (select문)

create index 인덱스이름
on 테이블명(필드명,필드명...)

create procedure 프로시저이름
is
begin
처리할 문법
end;

create procedure 프로시저이름(cnt in number)
is
begin
처리할 문법
end;
실행 : execute 프로시저명;

create function 함수이름
return number;
is
사용할 변수
begin
사용할 변수:=값
return 사용할 변수
end;

create trigger 트리거이름
after/before insert/update
on 사건이발생할테이블명
begin
처리할 명령
end;

프로시저 (PROCEDURE)

create or replace procedure 함수명

is

begin

문장1;

문장2;

end;

create or replace procedure del_all

is

begin

delete from test1;

delete from test2;

end;

프로시저 (PROCEDURE)

테이블1 만들기

```
create table test1(  
num int,  
name varchar2(10)  
);
```

데이터 입력하기

```
insert into test1 values (1,'hong');  
insert into test1 values (2,'hong');  
insert into test1 values (3,'hong');
```

데이터 확인하기

```
select * from test1;
```

테이블2 만들기

```
create table test2(  
num int,  
name varchar2(10)  
);
```

데이터 입력하기

```
insert into test2 values (1,'hong');  
insert into test2 values (2,'hong');  
insert into test2 values (3,'hong');
```

데이터 확인하기

```
select * from test2;
```

프로시저 입력파라메다

create or replace procedure test1numadd(cnt in number)

is

begin

update test1 set num=cnt;

end;

만들어진 프로시저 실행하기

```
select * from test1;
```

```
select * from test2;
```

만들어진 프로시저 실행하기


```
execute del_all;
```


프로시저 예제

//아래 구문을 프로시저를 이용해서 한번에 처리하시오.

```
insert into test1 values(2);
```

```
insert into test2 values(2);
```



```
create procedure testinsert  
is  
begin  
insert into test1 values(2);  
insert into test2 values(2);  
end;  
execute testinsert;  
select * from test1;  
select * from test2;
```

문제 해결

//위의 문제점은 변화는 값에 대해 처리는 어떻게 할 것인가?

create or replace procedure testinsert(id in number)

is

begin

insert into test1 values(id);

insert into test2 values(id);

end;

execute testinsert(3);

FUNCTION

-- Function 만들기

create function testfunc

return number

is

a number;

begin

a:=10;

return a;

end;

결과 확인

select count(*) from emp;

select testfunc() from dual;

SELECT결과를 INTO활용 결과 담기

```
create or replace function testfunc1
return number
is
r number:=0;
--DECLARE r INT default 0;
begin
--r:=select count(*) from emp;
select count(*) into r from emp;
--DBMS_OUTPUT.PUT_LINE(r);
return r;
end;
```

--test1테이블에서 번호를 이름하면 이름이 출력되는 함수를 작성하시오.

create or replace function selectname(n in number) --변수명이 필드명과 같으면 안됨.

return varchar2

is

result varchar2(30); --변수 선언시 크기까지 정의

begin

select name into result from test1 where num=n;

return result;

end;

/

select selectname(3) from dual;

트리거

트리거는 전 후로 일어나는 상황에 대해 그 조건이 만족할 때 실행되는 명령어의 집합을 의미합니다.

아래트리거는 test1테이블이 입력 된 후 begin~end명령이 실행되는 것을 의미합니다.

```
CREATE OR REPLACE TRIGGER TRG_01
```

```
AFTER INSERT
```

```
ON test1
```

```
BEGIN
```

```
insert into test2 values (1,'hong');
```

```
insert into test2 values (2,'hong');
```

```
insert into test2 values (3,'hong');
```

```
END;
```

트리거 확인

```
select * from test1;
```

```
select * from test2;
```

```
insert into test1 values (1,'hong');
```


INDEX

```
create index idx_test1_num on test1(num);
```

```
select * from emp;  
create index deptno_index on emp(deptno);  
create index ex_indexes on emp(ename,deptno);  
select * from emp;  
drop index deptno_index;  
drop index ex_indexes;
```

VIEW

일반적인 문장

```
select ename, hiredata from emp where deptno=10;
```

뷰생성

```
create view emp_NEWYORK as select ename, hiredata from emp where deptno=10;
```

```
select * from emp_newyork;
```

뷰덮어쓰기

```
create or replace view emp_newyork
```

```
as select ename,hiredata,sal from emp where deptno=10;
```

VIEW

뷰는 다른 두테이블을 join을 통해 다른 하나의 테이블로 만들어 졌을 경우
이 테이블이 자주 사용한다고 판단되면 view를 생성

Create view 뷰이름(필드명,필드명) AS (select A.name B.age from a,b)

Select * from 뷰이름

Drop view 뷰이름



create or replace view tview

as

select test1.num as 번호, test1.name 사용자1, test2.name 사용자2 from test1, test2 where
test1.num=test2.num;

SEQUENCE

```
create sequence test1_num_seq increment by 1 start with 1;
```

시퀀스의 값을 증가시키는 명령

```
Insert into test1 values(test1_num_seq.nextval, ' lee');
```

시퀀스 값의 현재값을 확인하는 명령

```
select test1_num_seq.currval from dual;
```

ROWNUM

select rownum,test1.* **from** test1;

반드시 테이블명.으로 처리해야한다.

트랜잭션 (COMMIT, ROLLBACK, SAVEPOINT)

```
savepoint s;
```

```
drop table test;
```

```
create table test(  
id number,  
name varchar2(30)  
);
```

```
SQL> savepoint s1;
```

```
SQL> insert into test values(1,'hong');
```

```
SQL> savepoint s2;
```

```
SQL> insert into test values(2,'hong');
```

```
SQL> savepoint s3;
```

```
SQL> insert into test values(3,'hong');
```

```
SQL> select * from test;
```

```
SQL> rollback to s3;
```

```
SQL> select * from test;
```

JOIN

```
select test1.*,test2.* from test1,test2 where test1.num=test2.num;
```


그룹 함수를 이용하여 값 입력

```
select max(num),name from test1 group by name;
```

```
select * from test1 where num=max(num);
```

```
select * from test1 where num=(select max(num) from test1);
```

검색

```
select * from test1 where num between 3 and 7;
```

현재 행번호 출력

```
select rownum, num, name from test1;
```

```
select rownum, num, name from test1 order by rownum desc;
```

SUB QUERY

AS 명령은 별명과 함께 복사 역할

```
SQL> SELECT DISTINCT NAME FROM TEST1;
```

```
SQL> CREATE VIEW TESTVIEW AS  
      SELECT DISTINCT NAME FROM TEST1;
```

```
SQL> CREATE TABLE TESTCOPY AS SELECT DISTINCT NAME FROM TEST1;
```

```
SQL> SELECT * FROM TESTVIEW;
```

```
SQL> SELECT * FROM TESTCOPY;
```

SEQUENCE

```
create sequence test1_num_seq start with 100 increment by 2;  
select test1_num_seq.nextval from dual;  
select test1_num_seq.currval from dual;
```