

11 장

데이터베이스 관리와 보안

Sejong Oh
Dankook University

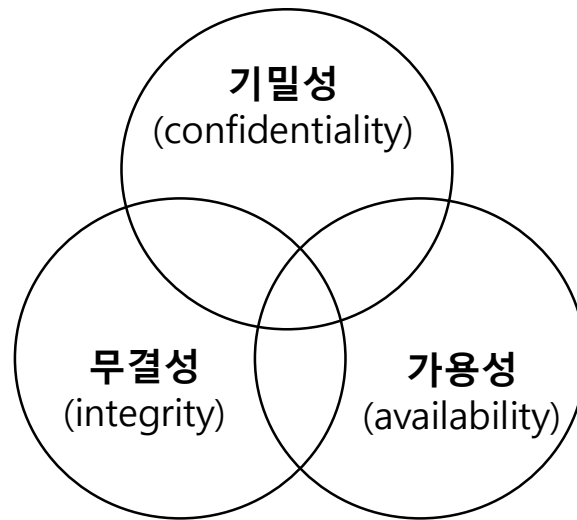
목차

- 1. 데이터베이스 보안 개요
- 2. 사용자 계정의 생성
- 3. 사용자 권한 관리
- 4. 역할의 관리
- 실습. 데이터베이스 내보내기 (export)

1. 데이터베이스 보안 개요

- 데이터 보안의 고려 요소

- 사용자가 데이터베이스에 대해 기대하는 것 중의 하나는 높은 보안성이다. 데이터베이스에 비즈니스 관련 데이터가 축적이 될수록, 현실 업무가 축적된 데이터에 기반하여 이루어질수록 데이터베이스 내에 저장된 데이터를 보호해야 할 필요성이 높아짐
- 데이터와 관련된 시스템의 보안성을 평가하거나 높이하고자 할 때 다음의 세가지 요소를 고려해야 한다



<그림 11-1> 데이터 보안의 3대 고려 요소

1. 데이터베이스 보안 개요

- 기밀성 (confidentiality)

- 승인되지 않은 사용자가 무단으로 데이터를 액세스하는 것을 방지하고, 승인된 사용자가 액세스하는 경우 데이터는 승인된 목적으로만 사용되도록 하는 것을 말한다.
- 기밀성은 프라이버시를 침해할 수 있는 정보의 공개로부터 데이터를 보호하는 것을 의미
- 기밀성을 위해 사용자의 데이터 액세스 권한은 담당하는 업무에 따라 필요한 만큼만 차등적으로 제공되어야 한다.

1. 데이터베이스 보안 개요

- 무결성 (integrity)

- 무결성은 우리가 이미 학습한 바와 같이 데이터를 일관성 있고 오류 없이 유지하는 것과 관련이 있음
- 데이터베이스 무결성은 데이터베이스 내에 있는 데이터가 불일치나 이상이 없도록 하는데 초점이 있지만 데이터베이스를 사용하는 전체 조직의 관점에서 무결성은 데이터베이스의 데이터뿐 아니라 조직의 업무 흐름과 사용자 및 데이터 액세스 패턴까지도 고려 대상이 된다.
- 예를 들면, 인터넷을 사용하여 제품 원가 계산 데이터에 액세스하는 재택 근무 직원은 근무시간 동안 또는 원가 계산 작업을 하는 특정 기간 동안만 액세스가 허용되어야 무결성이 지켜진다고 말할 수 있다.
- 또한 인쇄된 보고서는 적정 시점에 파쇄되어야 하고 USB 메모리에 데이터를 복사하는 일이나 외부에 이메일로 데이터 파일을 전송하는 것도 통제되어야 한다.

1. 데이터베이스 보안 개요

- 가용성 (availability)

- 가용성은 승인된 사용자가 요구할 때마다 승인된 데이터에 대한 액세스가 가능해야 함을 의미
- 어떤 이유로 시스템의 사용이 중단된다면 조직 전체에 손해를 끼칠 수 있기 때문
- 기밀성이나 무결성을 지나치게 높은 수준으로 유지하고자 하면 가용성이 떨어질 우려가 있으므로 적절한 수준에서 이루어져야 한다.
- 주말에는 모든 시스템에 대한 접근을 차단하는 정책을 사용한다면 기밀성은 높아지겠지만, 긴급한 업무를 처리해야하는 부서나 구성원에게는 가용성이 침해된 것이다.
- 따라서 시스템 가용성은 보안의 중요한 목표중 하나이다.

1. 데이터베이스 보안 개요

● 보안의 취약 요소

- 보안 위협은 시스템이 가진 취약성(vulnerability) 때문에 발생하는 경우가 많다

<표 12-1> 시스템의 취약 요소와 대응 방안

시스템 요소	보안 취약성의 예	대응방안
사용자	<ul style="list-style-type: none">• 비밀번호를 설정하지 않음• 비밀번호가 짧거나 생년월일이 포함• 사무실 문을 열어두고 퇴근• 사용자가 급여 정보를 장시간 화면에 띄움	<ul style="list-style-type: none">• 복잡한 암호를 사용• 다단계 인증을 사용• 보안 화면과 화면 보호기를 사용• 민감한 데이터에 대한 사용자 교육• 보안 카메라를 설치, 자동 도어록 사용
컴퓨터와 서비스	<ul style="list-style-type: none">• 데이터를 USB 메모리에 저장해둠• 컴퓨터가 여러 사람에 의해 공유됨• 권한이 없는 사용자가 컴퓨터 사용• 정전에 의한 하드디스크 손상• 컴퓨터 폐기시 하드디스크에 민감 정보 남아 있음	<ul style="list-style-type: none">• USB등 보조기억장치에 대한 엄격한 사용 정책 시행• 컴퓨터별로 사용자/권한 지정• 컴퓨터에 도난 방지장치 설정• 무정전 전원장치(UPS) 설치• 컴퓨터 폐기시 데이터 완전삭제 시행

1. 데이터베이스 보안 개요

<표 12-1> 시스템의 취약 요소와 대응 방안 (계속)

시스템 요소	보안 취약성의 예	대응방안
어플리케이션	<ul style="list-style-type: none">• SQL 삽입 공격• 어플리케이션 오류 (버퍼 오버플로우 등)• 이메일에 의한 공격 (스팸, 스미싱, 스니핑, 스푸핑)	<ul style="list-style-type: none">• SQL 삽입 취약성 제거• 시큐어 코딩 기법에 의한 어플리케이션 개발• 안티바이러스 필터 설치
데이터	<ul style="list-style-type: none">• 데이터가 여러 사람에게 오픈됨• 데이터가 원격지에서 액세스됨	<ul style="list-style-type: none">• 파일 시스템 보안 구현• 데이터 접근 권한 지정• 데이터 암호화

1. 데이터베이스 보안 개요

- 데이터베이스 보안 정책

- DBMS는 데이터를 보호하기 위한 다양한 방법들을 제공

- 인증(authentication)

- 다른 소프트웨어와는 달리 데이터베이스를 이용할 수 있기 위해서는 별도의 계정을 부여받아야하며, 로그인 과정을 통해 자신이 이용 권한이 있는 사용자임을 입증해야 한다.
- DBMS는 권한이 있는 사용자에 대한 계정, 비밀번호 정보를 관리한다.

- 접근제어(access control)

- 로그인을 통과한 사용자라 하더라도 데이터베이스 내의 모든 정보에 접근할 수 있는 것은 아니다.
- 데이터베이스 관리자는 사용자별로 권한을 할당하고, 사용자는 할당받은 권한 안에서만 데이터베이스를 이용할 수 있다.
- DBMS는 사용자로부터 어떤 요청이 있을 때 권한이 있는지를 먼저 검사하며, 권한이 없는 경우 요청을 거절한다.

1. 데이터베이스 보안 개요

◉ 감사(auditing)

- 감사는 사용자의 데이터베이스 작업을 모니터링하고 보안 로그(log)에 기록하는 것을 말한다.
- 데이터베이스 보안 담당자는 필요시 보안 로그를 통해 특정 사용자의 부정행위나 불법행위의 여부를 판단
- 보안 로그의 기록은 실행된 SQL문의 유형과 같은 개별 작업을 기반으로 하거나 사용자 이름, 응용 프로그램, 시간 등을 포함할 수 있는 요소의 조합을 기반으로 할 수 있다.
- 데이터베이스 관리자는 민감한 정보가 조회되거나 갱신될 때 이를 보안 로그에 기록하도록 설정할 수 있다.

◉ 암호화(encryption)

- 기본적으로 데이터베이스 전체는 침입자에 의해 탈취되어도 열어볼 수 없도록 데이터베이스 전체를 암호화 할수 있는 기능을 제공
- 비밀번호, 주민등록번호, 신용카드번호 등 테이블 내에 저장되는 개별 정보를 별도로 암호화 하는 방법을 제공

user	password
mysql.infoschema	\$A\$005\$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTH...
mysql.session	\$A\$005\$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTH...
mysql.sys	\$A\$005\$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTH...
root	\$A\$005\$+O.mdl!:\$J!!-&eY1 u !bBUgh9k.ggOKX4TAnPiLab5ffEepyeL...

1. 데이터베이스 보안 개요

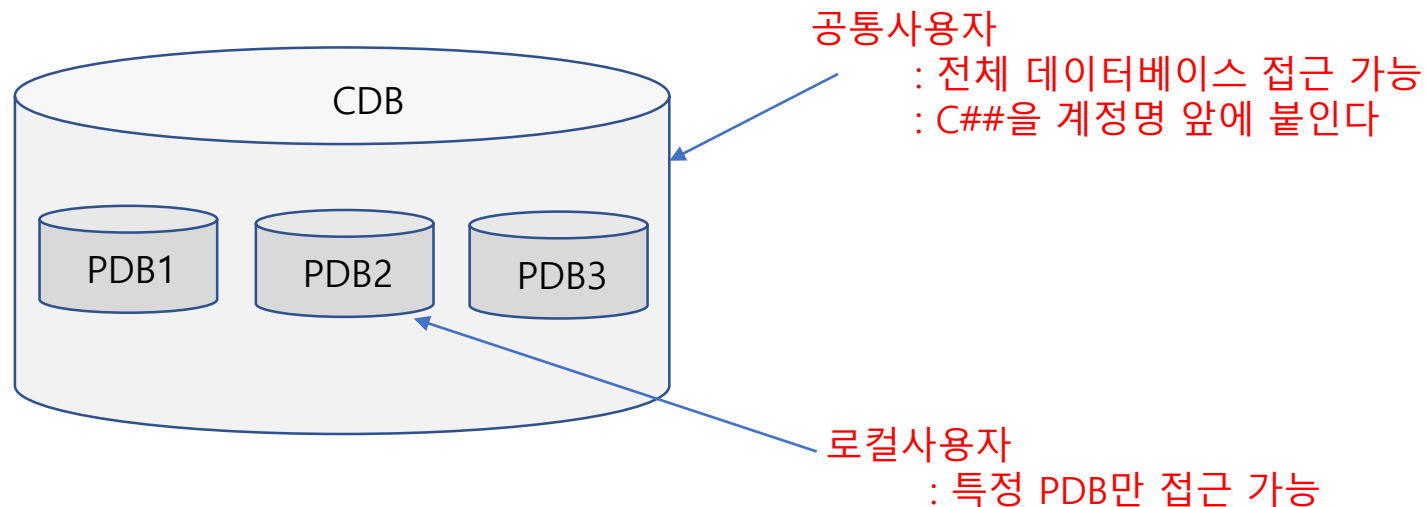
○ 무결성 제어(integrity control)

- 데이터 무결성은 데이터베이스 내에 저장된 데이터가 오류가 없고 데이터들간 모순이 없는 상태로 유지되어야 하는 요구사항을 말한다
- DBMS는 여러 방법으로 데이터 무결성을 유지
- 외래키는 참조 무결성을 유지하기 위한 방법
- DBMS는 허가되지 않은 사용자의 데이터 변경이나 파괴, 저장, 그리고 데이터를 손상시킬 수 있는 시스템 오류, 고장들로부터 데이터베이스를 보호하는 기능을 제공
- DBMS는 적절한 시스템 통제와 다양한 백업 및 복구 절차 등을 통하여 데이터베이스를 보호

2. 사용자 계정의 생성

● 개요

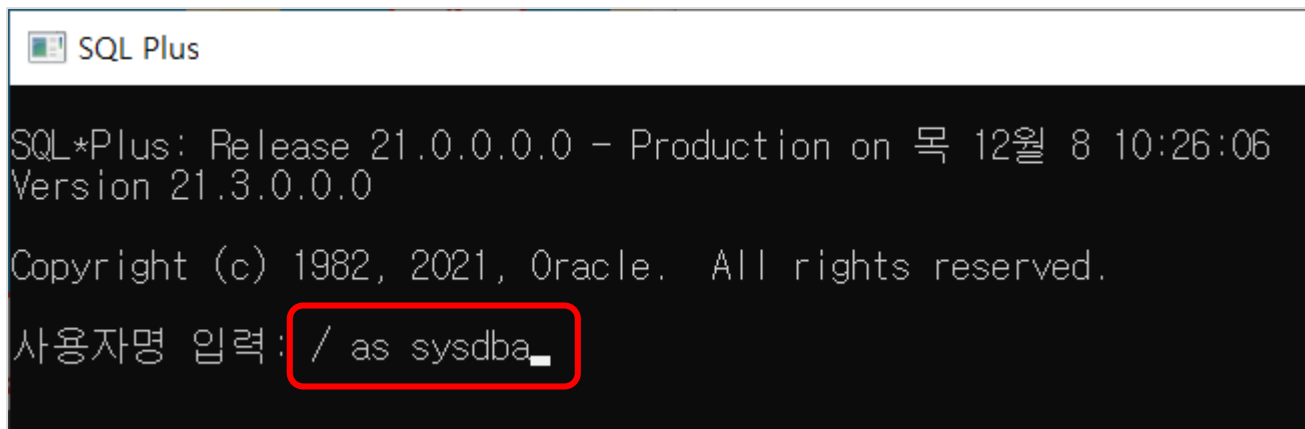
- DBMS는 데이터베이스의 보호를 위해 운영체제와는 별도로 사용자 및 사용자들에 대한 권한을 관리
- 오라클에서는 2계층의 데이터베이스가 존재하고, 그에 맞추어 두가지 유형의 사용자가 있음을 상기
- 공통 사용자는 CDB 및 PDB 모두에 접근이 가능한 사용자이며 로컬 사용자는 특정 PDB에 대해서만 접근이 가능



<그림 11-3> 오라클 데이터베이스 구조와 사용자

2. 사용자 계정의 생성

- 공통 사용자는 CDB에 관리자 계정으로 로그인한 상태에서 생성할 수 있으며 로컬 사용자는 특정 PDB에 관리자 계정으로 로그인한 상태에서 생성할 수 있다.



```
SQL Plus

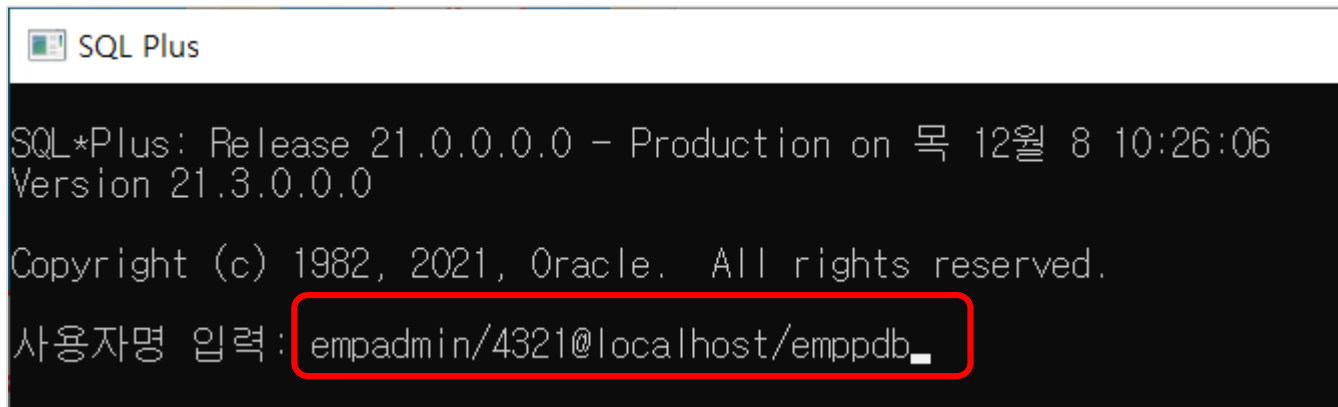
SQL*Plus: Release 21.0.0.0.0 - Production on 목 12월 8 10:26:06
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

사용자명 입력: / as sysdba_
```



CDB에 접속
공통사용자 생성 가능



```
SQL Plus

SQL*Plus: Release 21.0.0.0.0 - Production on 목 12월 8 10:26:06
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

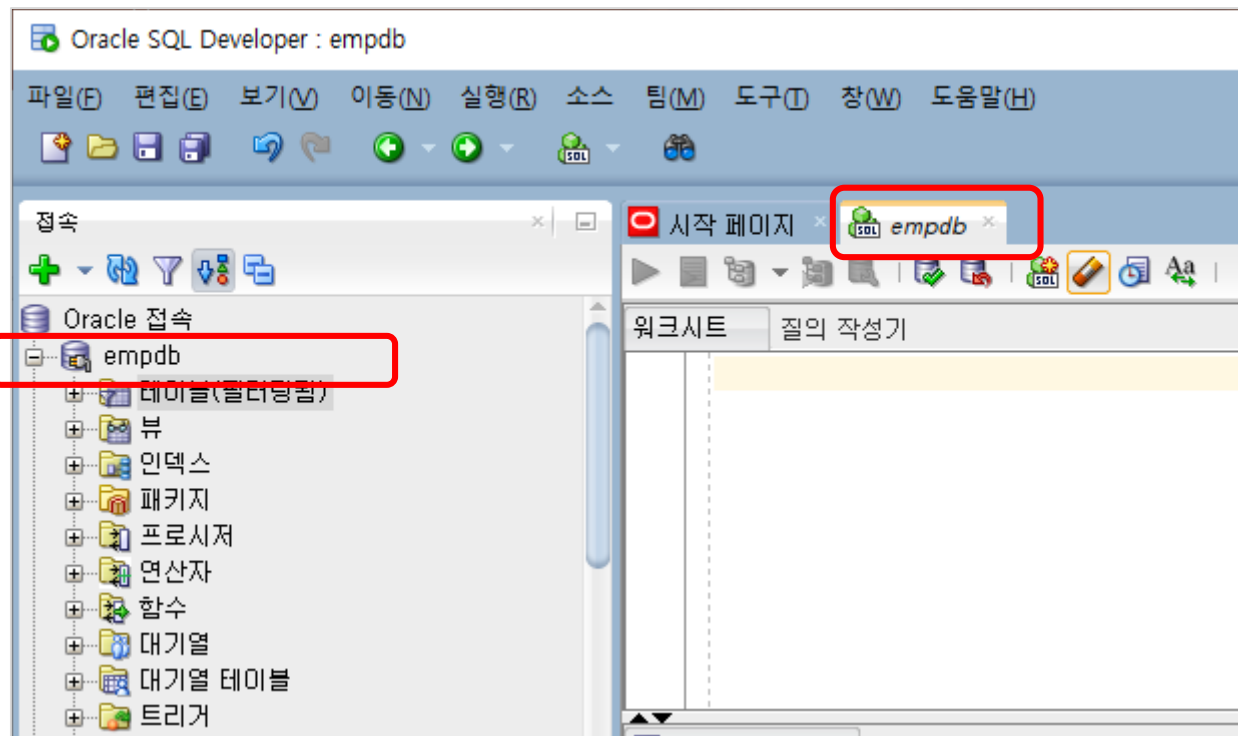
사용자명 입력: empadmin/4321@localhost/emppdb_
```



PDB(emppdb)에 접속
로컬사용자 생성 가능

2. 사용자 계정의 생성

- 오라클에서 로컬 사용자의 생성
 - 사용자 생성 및 관리는 관리자 권한이 필요
 - emppdb에는 scott 계정이 있는데 관리자 권한도 가지고 있다
 - 접속 경로 empdb로 접속하여 권한 작업을 진행



2. 사용자 계정의 생성

○ (1) 사용자 계정의 생성

<문법>

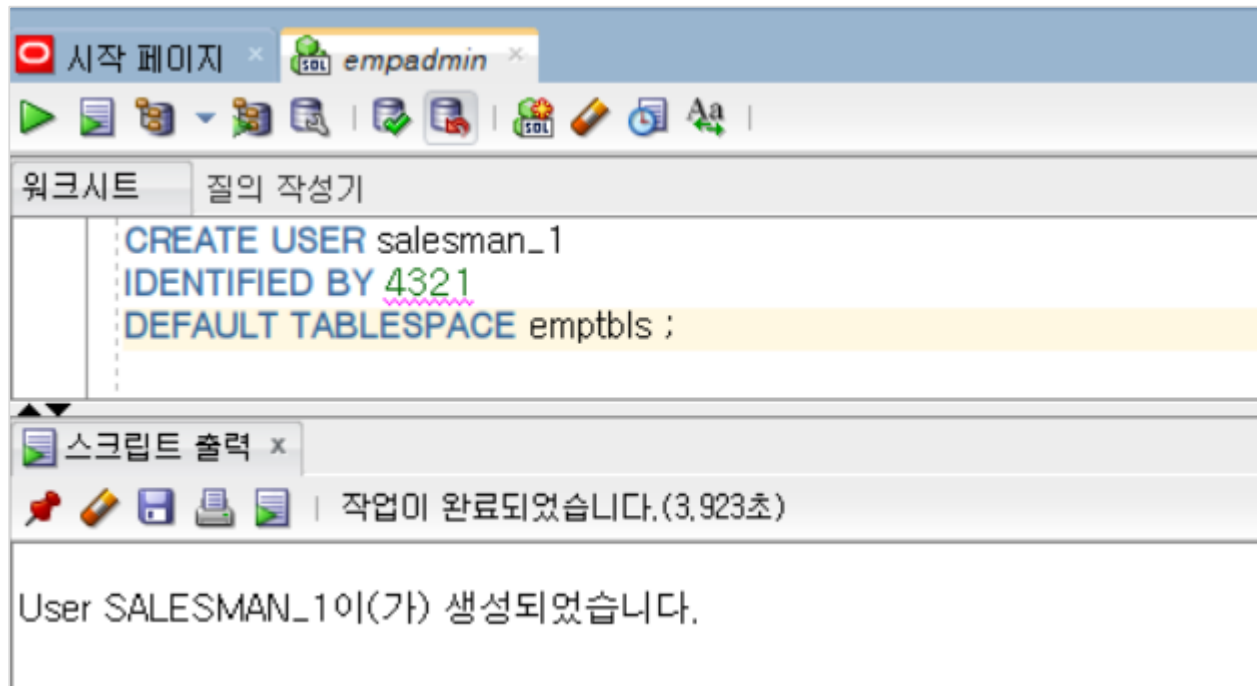
```
CREATE USER 사용자계정명  
IDENTIFIED BY 비밀번호  
DEFAULT TABLESPACE 테이블스페이스 이름 ;
```

DEFAULT TABLESPACE : 사용자계정이 미래에 테이블을 만들게 되면 테이블 데이터가 저장될 테이블스페이스를 지정

<예제>

```
CREATE USER salesman_1  
IDENTIFIED BY 4321  
DEFAULT TABLESPACE emptbls ;
```

2. 사용자 계정의 생성



2. 사용자 계정의 생성

- 생성된 사용자 계정의 확인

```
SELECT username, default_tablespace  
FROM dba_users  
WHERE account_status = 'OPEN' ;
```

	USERNAME	DEFAULT_TABLESPACE
1	SYS	SYSTEM
2	SYSTEM	SYSTEM
3	SYSRAC	SYSTEM
4	EMPADMIN	SYSTEM
5	SALESMAN_1	EMPTBLS
6	SCOTT	EMPTBLS

2. 사용자 계정의 생성

- GUI를 통한 사용자 생성

- SQL Developer의 GUI를 통해서 사용자를 생성하는 것도 가능
- ① 접속창의 empadmin 항목중 [다른사용자]→[사용자 생성]



2. 사용자 계정의 생성

- ② 사용자 생성을 위한 창이 표시되면 다음과 같이 입력

사용자 생성

사용자 | 부여된 롤 | 시스템 권한 | 할당량 | SQL

사용자 이름: salesman_2

새 비밀번호: ****

비밀번호 확인: ****

☐ 비밀번호가 만료됨(다음번에 로그인할 때 사용자가 변경해야 함)

☐ 운영체제 사용자

☐ 계정이 잠겨 있습니다.

☐ 에디션 사용

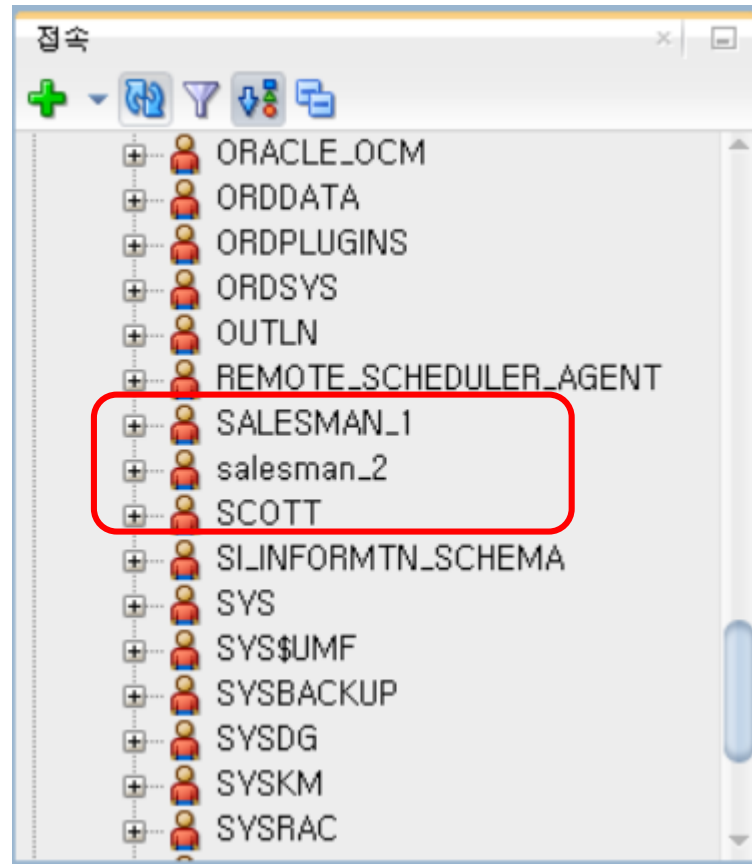
기본 테이블스페이스: EMPTBLS

임시 테이블스페이스:

도움말(H) 적용(A) 닫기(C)

2. 사용자 계정의 생성

- ③ 접속창의 empadmin 항목중 [다른사용자]의 하위 항목에서 생성된 사용자를 확인



2. 사용자 계정의 생성

Note. 오라클의 관리자 계정

오라클을 설치하게 되면 데이터베이스 관리를 위한 관리자 계정들이 함께 생성이 된다. 여러 유형의 관리자 계정이 있는데 다음의 계정들은 기억할 필요가 있다.

SYS

- 최상위 권한을 갖는 관리자.
- 오라클 데이터베이스의 근간이 되는 자료사전(data dictionary)의 소유자
- 데이터베이스의 생성 및 삭제 권한이 있음

SYSTEM

- 권한은 SYS와 동일하지만 데이터베이스 생성/삭제 권한은 없음.

우리가 실습에서 자주 접했던 SYSDBA는 사용자 계정의 이름이 아니라 특정 권한들 묶어놓은 역할(role)의 이름이다. 데이터베이스 생성 및 사용자 생성 권한 등을 포함한다. 역할에 대해서는 11.4절에서 학습한다.

3. 사용자 권한 관리

- 오라클에서의 권한은 크게 시스템 권한과 오브젝트 권한으로 구분

<테이블 11-2> 오라클 권한 분류

권한	설명
시스템 권한	데이터베이스 수준의 작업을 수행하기 위한 권한 (예: 유저생성, 권한관리, 테이블생성, 모든 테이블 조회 등)
오브젝트 권한	테이블, 인덱스, 함수 및 프로시저 등 오브젝트에 대한 작업 권한 (예: 테이블에 대한 튜플 입력,수정,삭제 등)

- 권한의 부여와 회수에 사용되는 SQL 명령어

명령어	기능
GRANT	사용자에게 권한을 부여한다
REVOKE	사용자에게 권한을 회수한다

3. 사용자 권한 관리

Note.

- 권한부여 작업에는 권한을 부여하는 '부여자'와 권한을 부여받는 '대상자'가 존재
- '부여자'는 **자신이 소유한 권한중에서만** 일부 또는 전부를 '대상자'에게 부여할 수 있다. (자신이 소유하지 않은 권한은 '대상자'에게 부여할 수 없다)
- 따라서 권한부여 작업은 보통 충분한 권한을 가진 관리자가 수행
- 사용자 A 가 생성한 모든 데이터베이스 객체는 사용자 A 의 소유이며 사용자 A는 자신이 생성한 객체에 대해 모든 권한을 갖는다.

3. 사용자 권한 관리

GRANT 명령어

<문법>

```
GRANT 부여할 권한 [ON 부여 대상]
TO 사용자계정 또는 역할
[WITH GRANT OPTION]
[PUBLIC] ;
```

- 부여할 수 있는 권한의 종류 : CREATE, INSERT, UPDATE, DELETE 등 우리가 지금까지 배운 거의 모든 SQL 명령어를 포함
- 모든 권한을 주고자 하는 경우는 ALL PRIVILEGES를 지정
- 한번에 여러 개의 권한을 지정하는 것도 가능 (권한들을 콤마로 구분하여 나열)
- 권한의 부여 대상은 데이터베이스의 모든 객체들
- WITH GRANT OPTION : 권한을 부여받는 사용자 또는 역할이 또 다른 사용자나 역할에게 자신의 권한을 부여 할 수 있는지 여부를 지정 (선택사항)
- PUBLIC : 해당 권한을 데이터베이스에 존재하는 모든 사용자에게 할당할 때 사용 (선택사항)

3. 사용자 권한 관리

부여 가능한 권한의 예와 설명

부여할 권한	설명
CONNECT	데이터베이스 연결 권한
RESOURCE	데이터베이스 객체 생성, 변경, 삭제 권한
DBA	DB 관리자 권한
ALL PRIVILEGES	모든 권한
CREATE, ALTER, DROP	CREATE, ALTER, DROP 권한
INSERT, UPDATE, DELETE	INSERT, UPDATE, DELETE 권한
SELECT	SELECT 권한

3. 사용자 권한 관리

- 권한 부여의 예

```
GRANT connect TO salesman_1 ;  
GRANT select, insert ON emp TO salesman_1 ;  
GRANT select ON dept TO salesman_1 ;
```

- 부여된 권한의 확인

```
SELECT *  
FROM DBA_TAB_PRIVS  
WHERE grantor = 'SCOTT' ;
```

3. 사용자 권한 관리

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 4(0,444초)

	GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE	HIERARCHY	C
1	SALESMAN_1	SCOTT	EMP	SCOTT	INSERT	NO	NO	NC
2	SALESMAN_1	SCOTT	EMP	SCOTT	SELECT	NO	NO	NC
3	SALESMAN_1	SCOTT	DEPT	SCOTT	SELECT	NO	NO	NC
4	PUBLIC	SYS	SCOTT	SCOTT	INHERIT PRIVILEGES	NO	NO	NC

권한 대상자

객체 소유자

대상 객체

권한 부여자

부여 권한

권한 재부여
가능 여부

3. 사용자 권한 관리

- SQL Plus를 통한 salesman_1 계정의 권한 확인 (1)

사용자명 입력: salesman_1/4321@localhost/emppdb

SQL Plus

```
SQL*Plus: Release 21.0.0.0.0 - Production on 목 12월 8 15:33:43 2022
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

사용자명 입력: salesman_1/4321@localhost/emppdb

다음에 접속됨:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
```

- saleman_1 계정에 connect 권한을 부여했기 때문에 데이터베이스 로그인이 가능

3. 사용자 권한 관리

- SQL Plus를 통한 salesman_1 계정의 권한 확인 (2)

```
SQL> SELECT * FROM dept ;
```

```
SELECT * FROM dept
```

```
*
```

1행에 오류:

ORA-00942: 테이블 또는 뷰가 존재하지 않습니다

- dept 테이블에 대한 select 권한을 부여했는데 왜 테이블을 볼 수 없는 것일까? →
dept 가 salesman_1이 생성한 객체가 아니기 때문

3. 사용자 권한 관리

<조회 권한은 있지만 자신이 생성하지 않은 테이블의 조회 방법>

```
SQL> SELECT * FROM scott.dept ;
```

```
DEPTNO DNAME LOC
```

```
-----
```

```
10 ACCOUNTING NEW YORK
```

```
20 RESEARCH DALLAS
```

```
30 SALES CHICAGO
```

```
40 OPERATIONS BOSTON
```

- 테이블 이름 앞에 생성한 사용자의 이름을 붙인다.

3. 사용자 권한 관리

- SQL Plus를 통한 salesman_1 계정의 권한 확인 (3)

```
SQL> UPDATE scott.dept SET loc = 'SEOUL' WHERE deptno = 10 ;
```

```
UPDATE scott.dept SET loc = 'SEOUL' WHERE deptno = 10
```

```
*
```

```
1행에 오류:
```

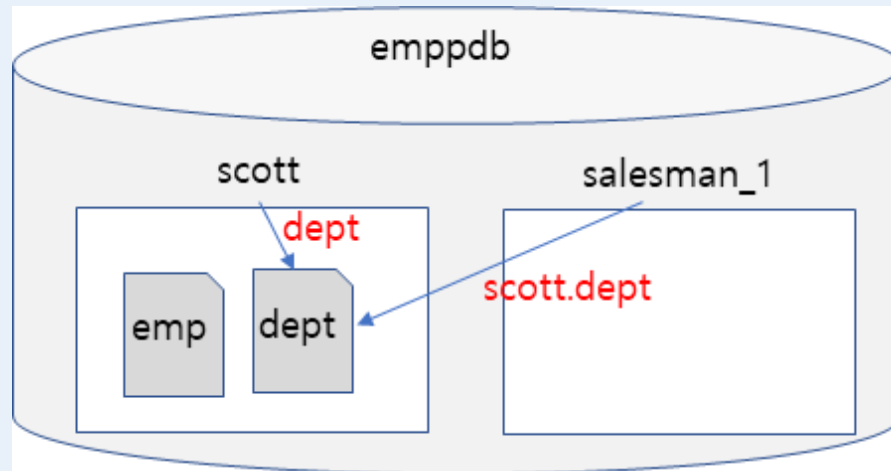
```
ORA-01031: 권한이 불충분합니다
```

- update 권한은 부여받지 못했기 때문에 실행 실패

Note.

오라클에서는 사용자 계정을 중심으로 데이터베이스 객체 관리를 한다. 그렇기 때문에 같은 데이터베이스 안에 있는 객체라 하더라도 누가 생성한 것인가에 따라 접근 방법이 다르다.

- 1) 내가 만든 객체: 모두 접근 가능
- 2) 타사용자가 만든 객체 : 접근 불가
- 2) 타사용자가 만든 객체인데 접근권한을 부여받은 경우 : 접근 가능 (단 스키마 이름을 객체 앞에 붙여야함)



오라클에서 스키마는 특정 사용자가 생성한 객체 전체를 가리키는 용어로 **사용자 이름이 곧 스키마 이름**이다. 위 그림에서 emppdb에는 두 개의 스키마가 있고, emp, dept 테이블은 scott 스키마에 포함되어 있다.

4. 역할(role)의 관리

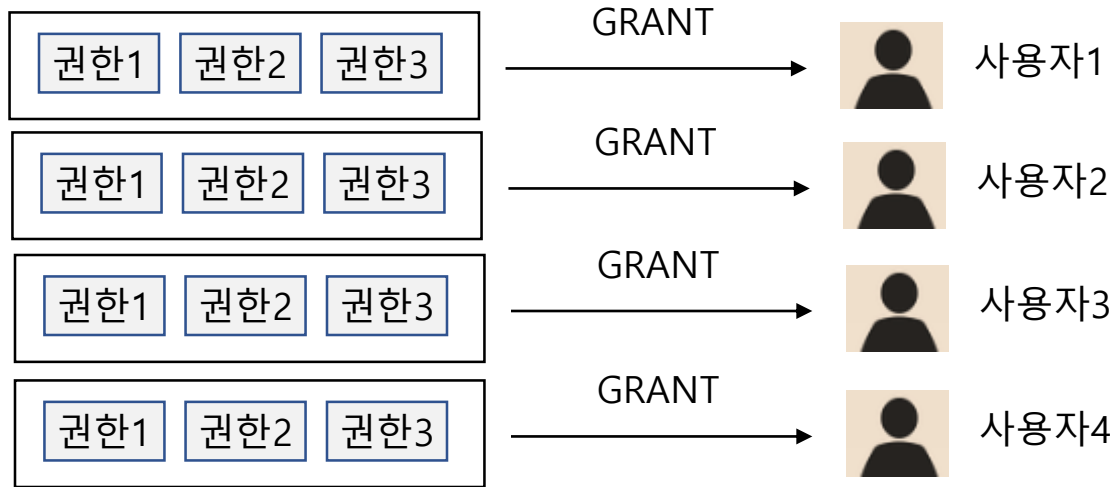
- 역할(role)의 필요성

- 50명의 개발자가 팀을 나누어서 개발에 참여하는 상황을 가정
- 개발자들이 각각 하나의 계정을 할당 받음
- 데이터베이스 관리자는 50개의 계정을 생성해야 하고 각 계정에 대한 권한 부여 작업을 50회 실시

<예상되는 어려움>

- 같은 팀에 속한 개발자들은 부여받는 권한도 동일하거나 유사할 것이다. 따라서 같은 팀에 속한 개발자들의 권한 부여는 같은 일을 중복적으로 수행하는 것이 된다.
- 어떤 팀에 10명의 개발자가 속해 있는데, 그 팀에 새로운 권한의 부여가 필요한 경우, 10명의 개발자에게 각각 권한을 부여해야 한다. 만일 이 과정에서 실수한다면 어떤 개발자는 정상적으로 업무를 수행할 수 없을 것이다.
- 권한 관리는 팀단위로 이루어짐에도 불구하고 데이터베이스 내에서는 계정별로 이루어지므로 불편함이 있다. 계정이 많아질수록 권한관리가 어렵게 된다.

4. 역할(role)의 관리

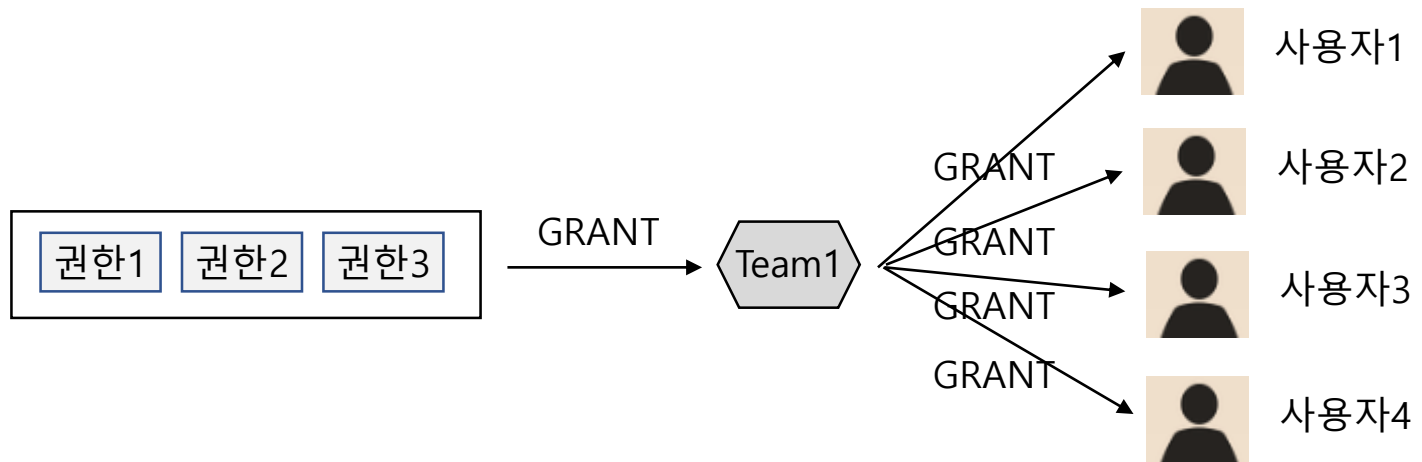


<그림 11-4> 권한의 개별 부여 상황

4. 역할(role)의 관리

해결 방안

- 공통 권한을 갖는 사용자가 다수 있다면 권한을 개별 사용자에게 일일이 부여하기 보다는 역할을 생성하여 공통 권한을 그 역할에 부여한 뒤 사용자들에게 다시 역할을 부여하는 방식을 사용

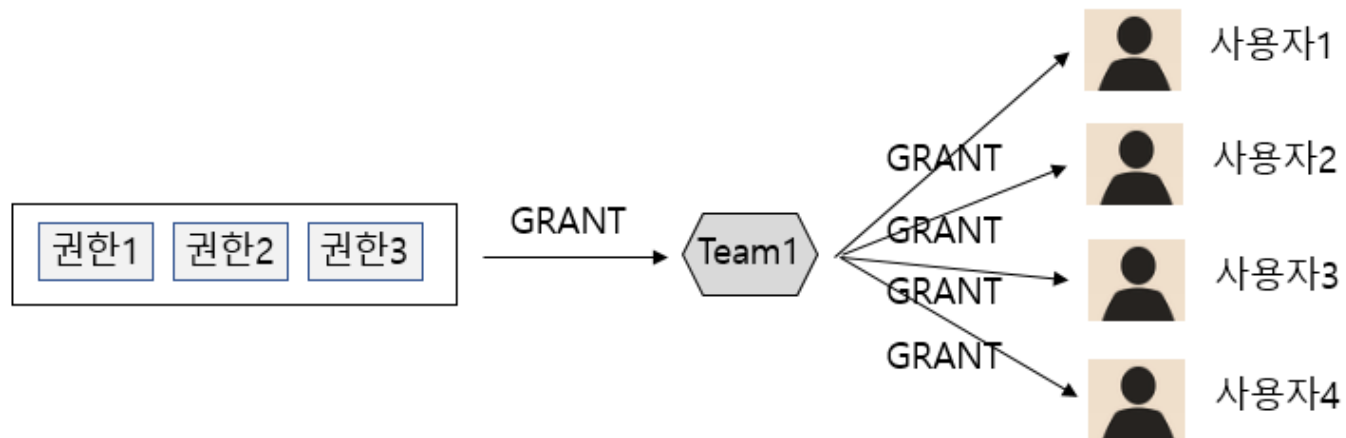


<그림 11-5> 역할을 통한 권한의 부여

4. 역할(role)의 관리

● 역할의 장점

- 어떤 팀에 새로운 개발자가 들어오면 이미 생성된 역할을 부여함으로써 그 개발자가 필요로하는 권한들을 손쉽게 할당할 수 있다.
- 어떤 팀에 업무를 위해 새로운 권한이 필요한 경우 그 팀의 역할에 새로운 권한을 부여한다. 그러면 새로운 권한이 그 팀의 역할을 할당 받은 사용자들에게 자동적으로 부여되는 효과가 있다.
- 어떤 팀에 더 이상 필요가 없게된 권한이 있으면 그 팀의 역할에서 권한을 회수한다. 그러면 그 권한은 그 역할을 할당 받은 모든 사용자들에게서 회수되는 효과가 있다.



4. 역할(role)의 관리

- 오라클에서 역할의 생성과 권한의 부여

<문법>

```
CREATE ROLE 역할이름 ;
```

- 개발1팀의 업무를 위해 emppdb의 emp 테이블에 대한 select, insert, update 권한과 dept 테이블에 대한 select 권한이 필요하다고 가정 (4명의 개발자)

4. 역할(role)의 관리

-- 사용자 계정 생성

```
CREATE USER user1 IDENTIFIED BY 1234 ;
```

```
CREATE USER user2 IDENTIFIED BY 1235 ;
```

```
CREATE USER user3 IDENTIFIED BY 1236 ;
```

```
CREATE USER user4 IDENTIFIED BY 1237 ;
```

-- 역할의 생성

```
CREATE ROLE team1 ;
```

-- 역할에 권한 부여

```
GRANT select, insert, update ON emp TO team1 ;
```

```
GRANT select ON dept TO team1 ;
```

-- 역할을 사용자에게 부여

```
GRANT team1 TO user1, user2, user3, user4 ;
```

4. 역할(role)의 관리

- 개발1팀의 user1 에게만 dept 테이블에 대한 insert, update 권한을 부여

```
GRANT insert, update ON dept TO user1 ;
```

- 역할에 부여된 권한의 확인

```
SELECT *  
FROM DBA_TAB_PRIVS  
WHERE grantee = 'TEAM1' ;
```

	GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE
1	TEAM1	SCOTT	EMP	SCOTT	INSERT	NO
2	TEAM1	SCOTT	DEPT	SCOTT	SELECT	NO
3	TEAM1	SCOTT	EMP	SCOTT	SELECT	NO
4	TEAM1	SCOTT	EMP	SCOTT	UPDATE	NO

4. 역할(role)의 관리

- 사용자 user1에 부여된 권한과 역할의 확인

-- 직접 부여된 권한 확인

```
SELECT *  
FROM DBA_TAB_PRIVS  
WHERE grantee = 'USER1' ;
```

-- 부여된 역할 확인

```
SELECT *  
FROM DBA_ROLE_PRIVS  
WHERE grantee = 'USER1' ;
```

	GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE	HIERARCHY	COMMON
1	USER1	SCOTT	DEPT	SCOTT	INSERT	NO	NO	NO
2	USER1	SCOTT	DEPT	SCOTT	UPDATE	NO	NO	NO

	GRANTEE	GRANTED_ROLE	ADMIN_OPTION	DELEGATE_OPTION	DEFAULT_ROLE	COMMON
1	USER1	TEAM1	NO	NO	YES	NO

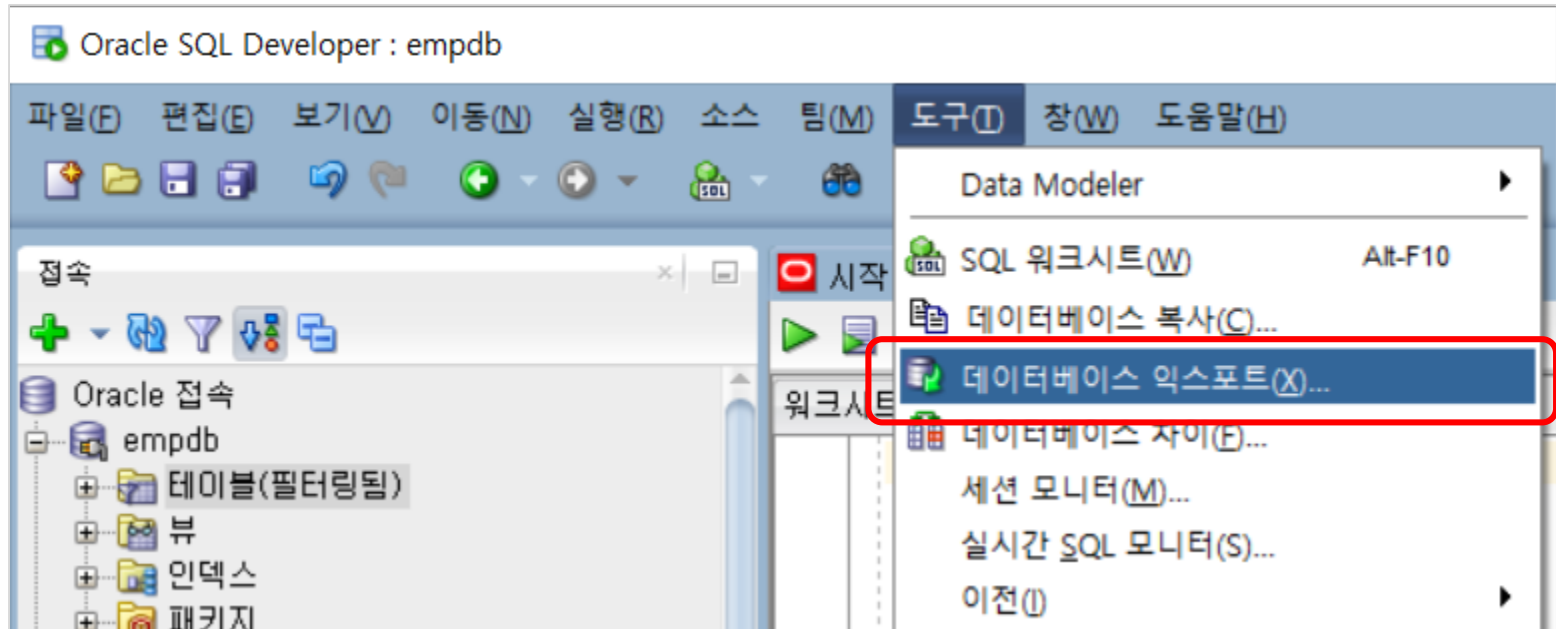
실습: 데이터베이스 내보내기(Export)

- 개요

- 데이터베이스의 내용을 밖으로 내보낼 수 있는 기능 실습
- 밖에 있는 SQL script를 실행할 수 있는 기능 실습

- 데이터베이스 내보내기

- ① SQL Developer 메인 메뉴에서 [도구]→[데이터베이스 익스포트]를 선택한다.



실습: 데이터베이스 내보내기(Export)

- ② 익스포트 마법사가 실행되면 첫단계로서 내보낼 데이터베이스 접속 경로 및 SQL 스크립트를 저장할 피파일을 지정하고, 필요한 옵션들을 설정한다.

익스포트 마법사 - 단계 1/5

소스/대상

소스/대상

익스포트할 유형

객체 지정

데이터 지정

익스포트 요약

접속(C): empdb 내보낼 데이터베이스 접속경로

☒ DDL 익스포트(E)

☐ BYTE 키워드 추가(B) ☒ 뷰에 FORCE 추가(W) ☐ 계단식 삭제(P) ☐ 종속 항목(N)

☐ 삭제(D) ☒ 권한 부여(A) ☒ 분할 ☒ 보기 쉽게 인쇄(N)

☒ 스키마 표시(S) ☒ 저장 영역(G) ☒ 테이블스페이스 ☒ 터미네이터(T)

버전: COMPATIBLE

☒ 데이터 익스포트(O)

형식(F): insert ☒ 스키마 표시(S)

행 터미네이터(L): 환경 기본값

☐ 다음마다 커밋 포함 100 행

다른 이름으로 저... 단일 파일 ☐ 압축됨(B) 인코딩(I): UTF8

파일(F): D:\test\wemppdb.sql 찾아보기(R)...

SQL 스크립트 저장 파일

☐ 요약으로 이동합니다(M).

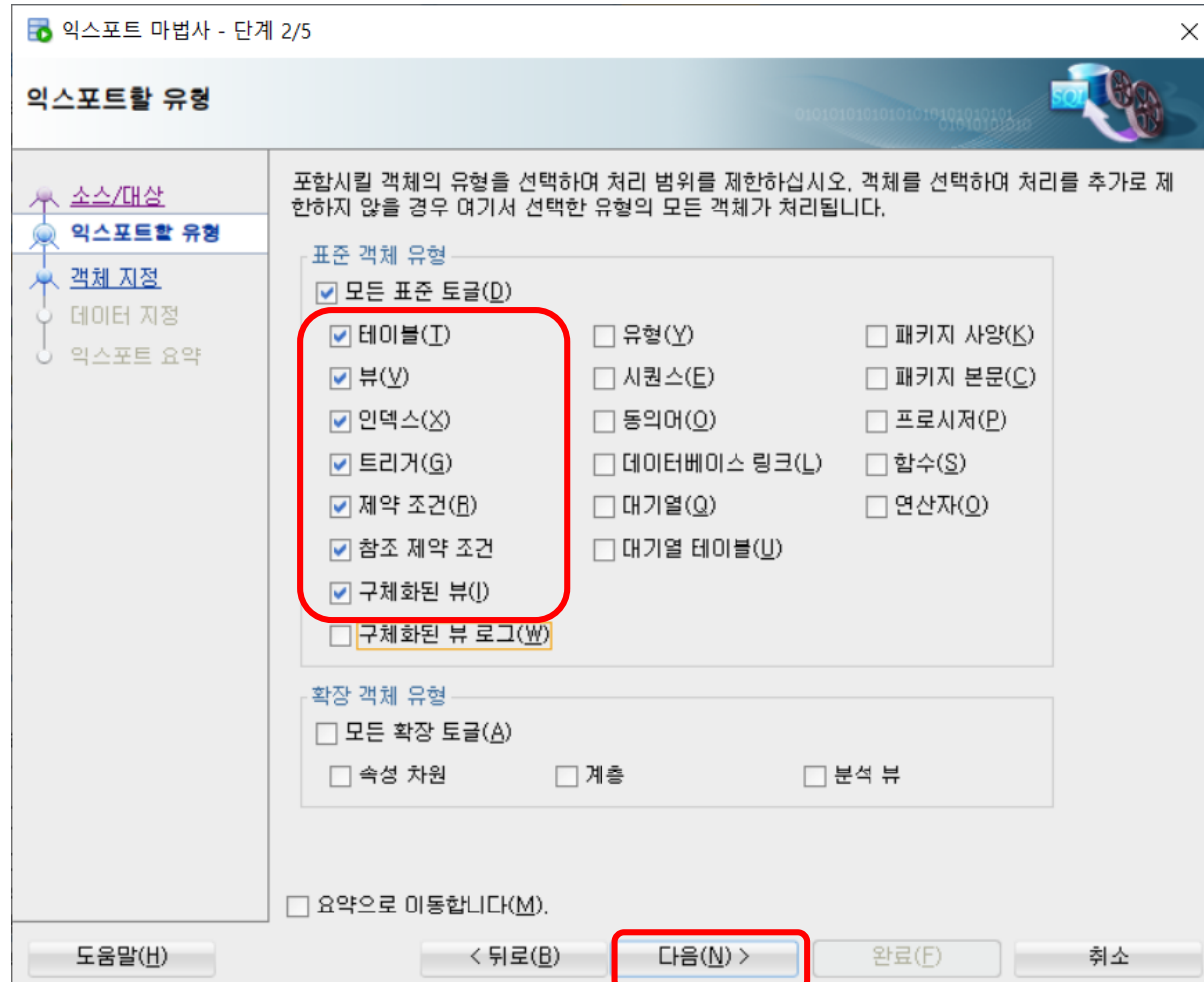
도움말(H) < 뒤로(B) 다음(N) > 완료(F) 취소

데이터베이스
구조를
CREATE문 형태
로 저장

테이블 데이터를
INSERT문 형태로
저장

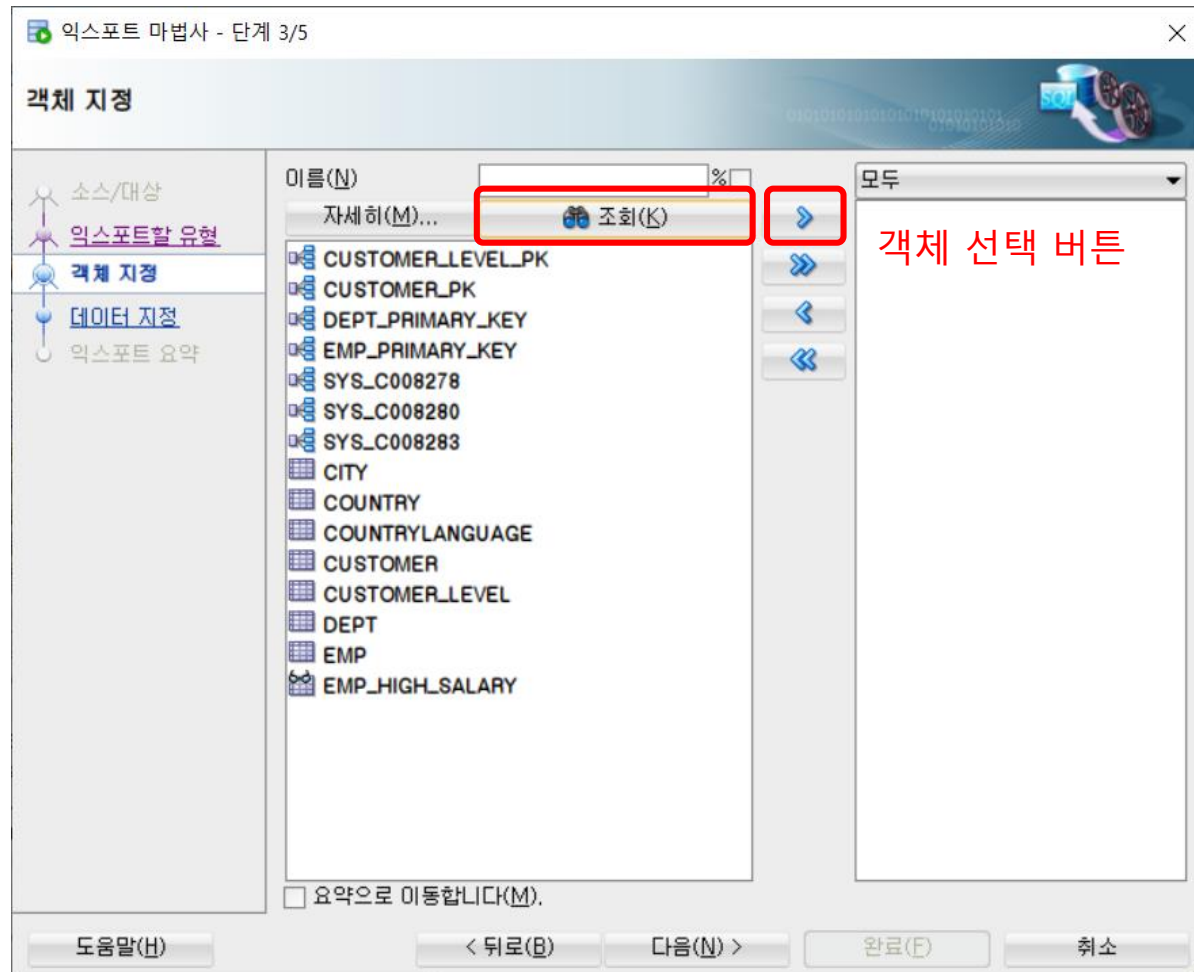
실습: 데이터베이스 내보내기(Export)

- ③ 두 번째 단계에서는 내보내기를 원하는 객체의 종류를 선택한다.



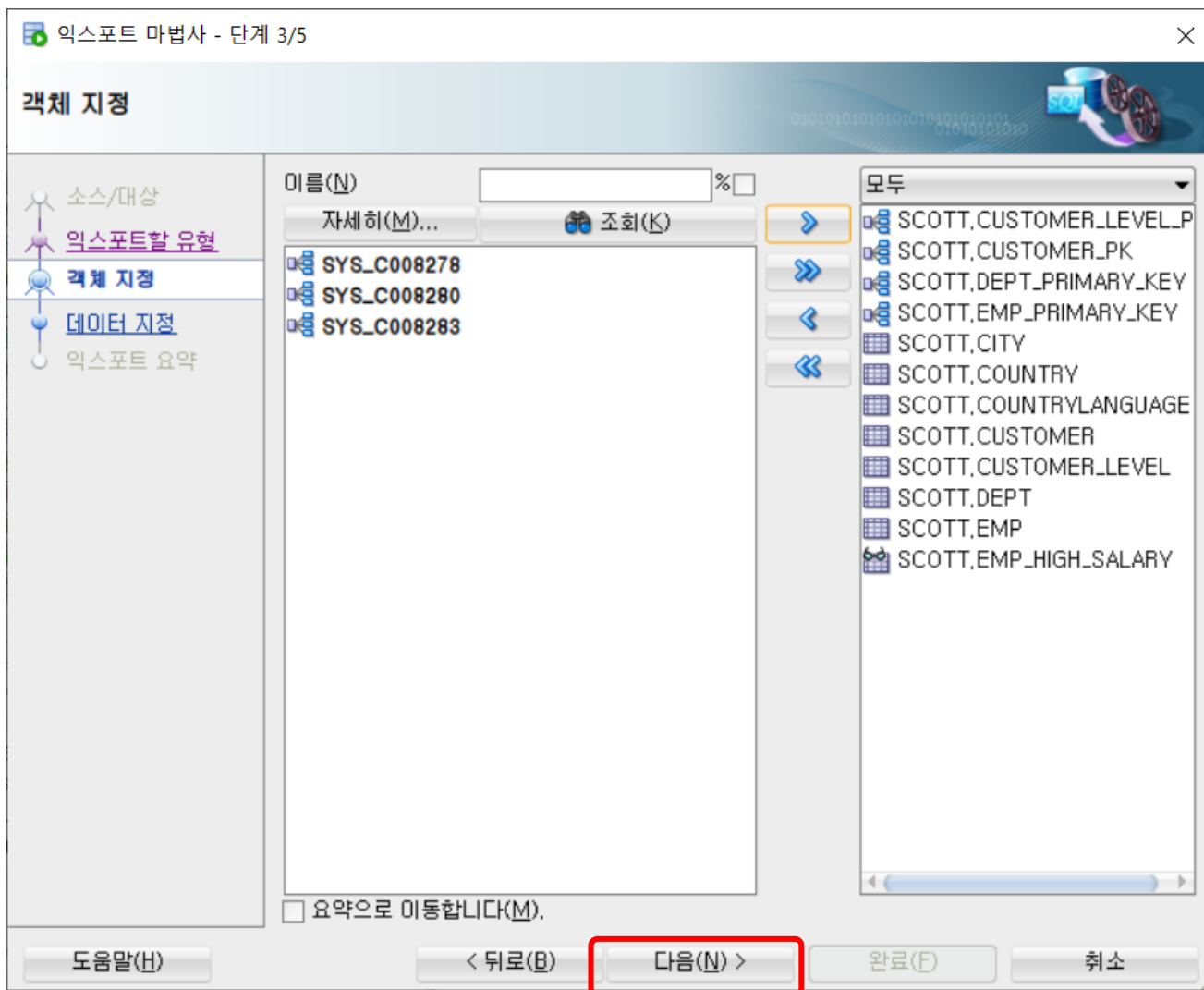
실습: 데이터베이스 내보내기(Export)

- ④ [조회] 버튼을 클릭하면 객체 리스트가 표시되는데, 여기서 내보낼 객체를 선택한 후 객체 선택 버튼을 클릭한다. 그러면 내보낼 객체들이 오른쪽 창으로 이동한다



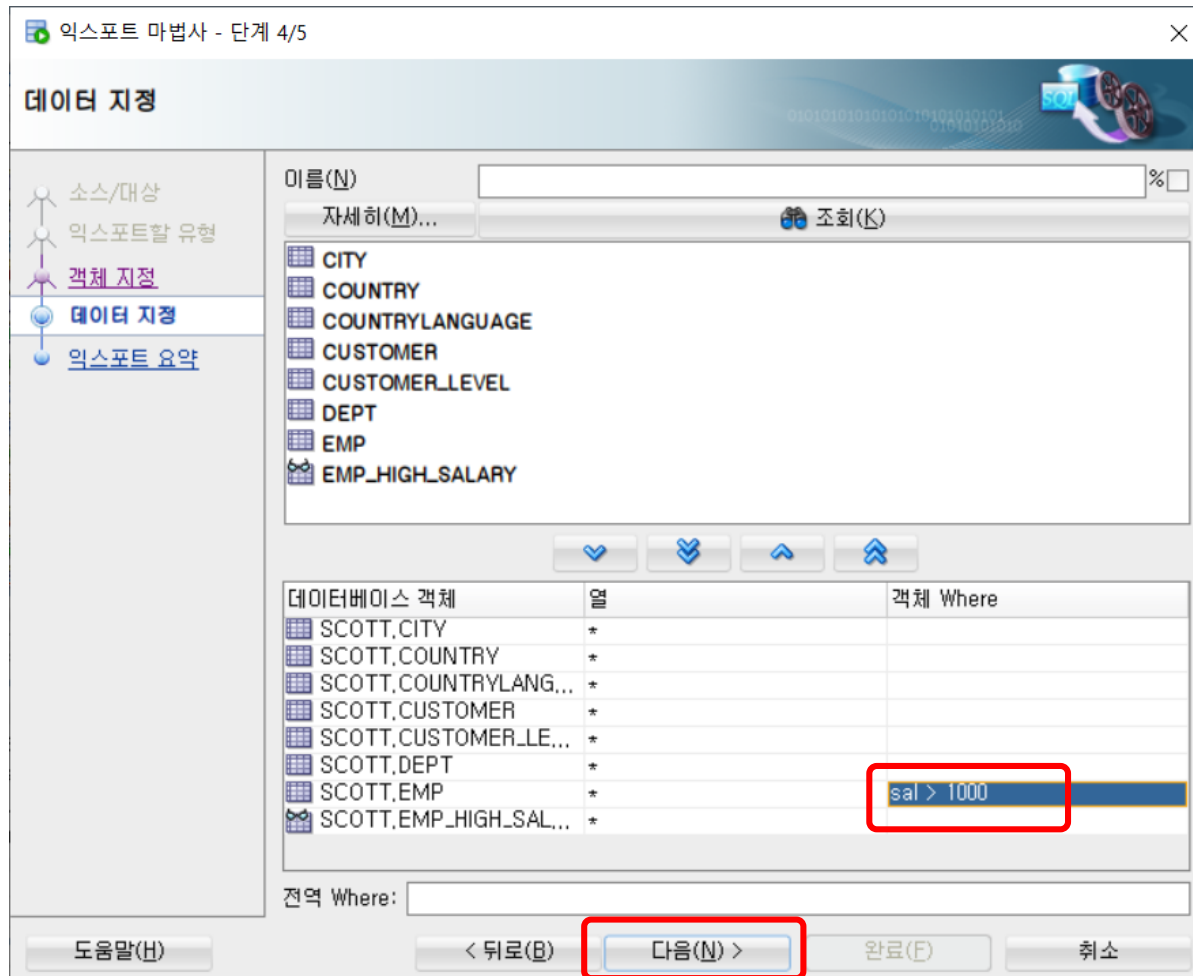
실습: 데이터베이스 내보내기(Export)

<선택된 객체의 이동 후 모습>



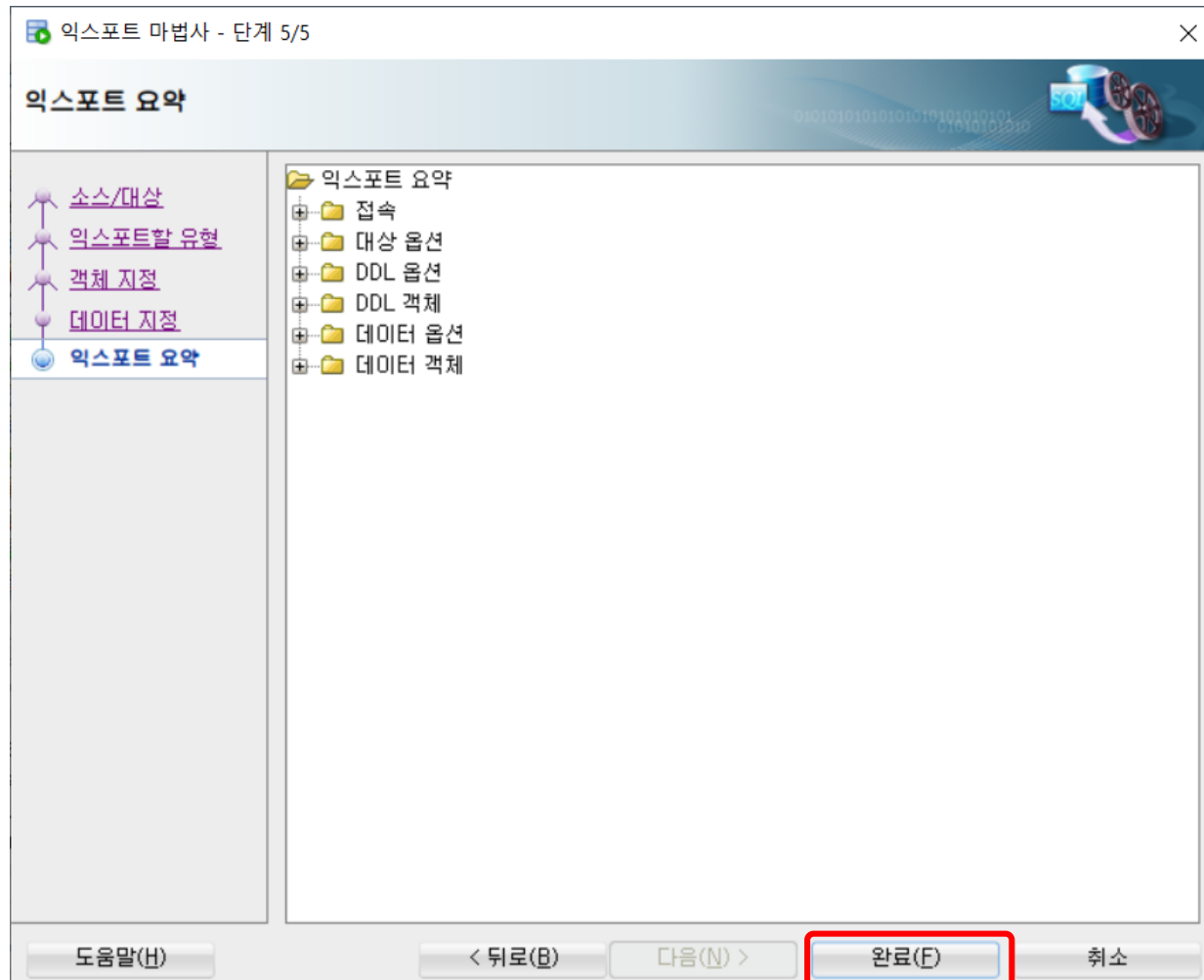
실습: 데이터베이스 내보내기(Export)

- ⑤ 테이블별로 내보낼 튜플들을 선택한다. 아무 지정도 하지 않으면 전체 튜플을 내보낸다.
 - 아래의 경우는 emp 테이블에서 sal(급여)가 1000 이상인 튜플만 선택하여 내보내도록 지정한 것이다. SELECT문의 WHERE절을 서술한다고 생각하면 된다.



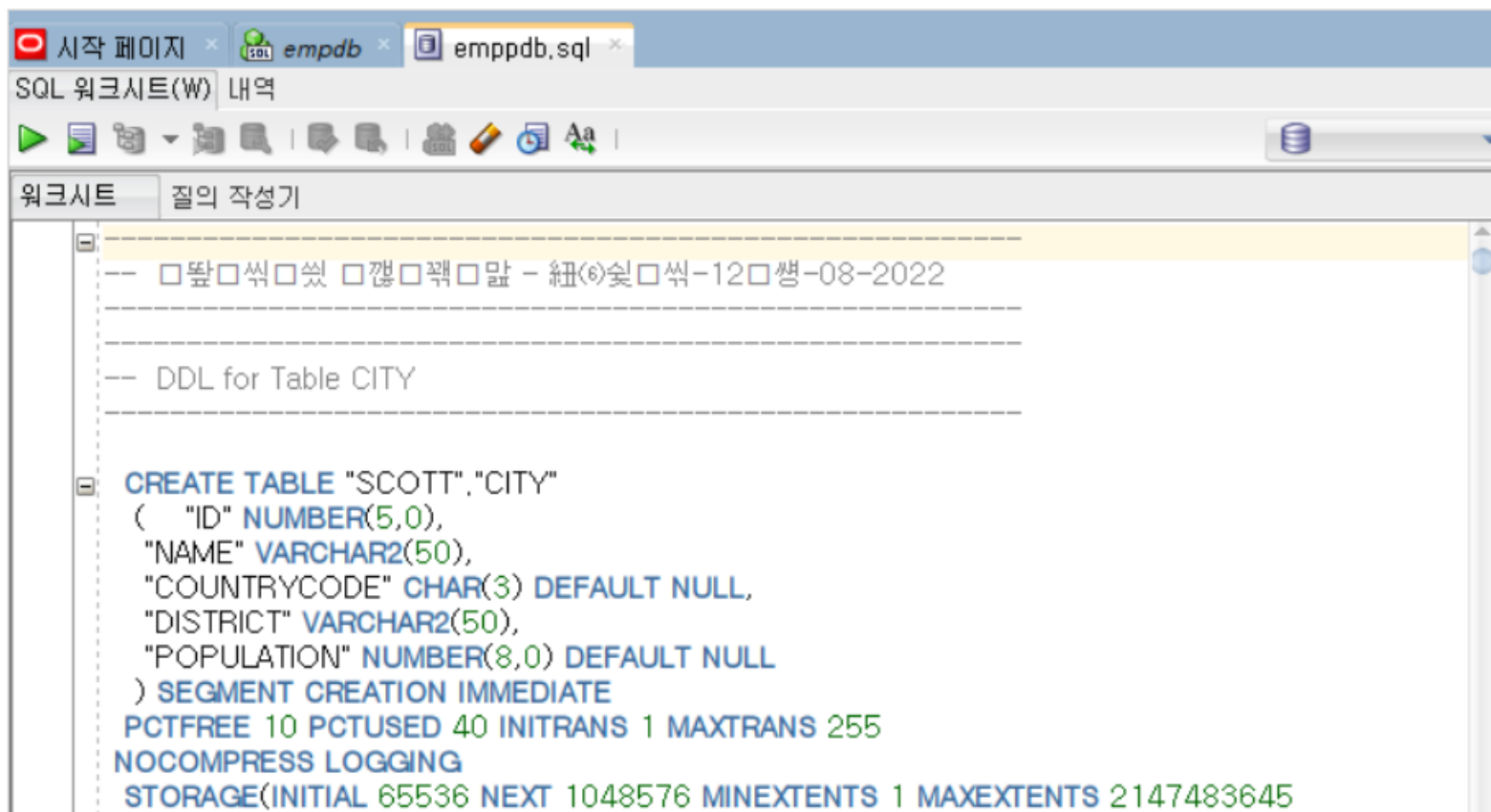
실습: 데이터베이스 내보내기(Export)

- ⑥ 내보내기 작업에 대한 요약을 보여준다. [완료]버튼을 클릭하면 내보내기가 실행된다. 내보내기의 실행은 약간의 시간을 필요로 한다.



실습: 데이터베이스 내보내기(Export)

- ⑦ 내보내기 작업이 완료되면 워크시트에 생성된 SQL 스크립트가 표시된다,

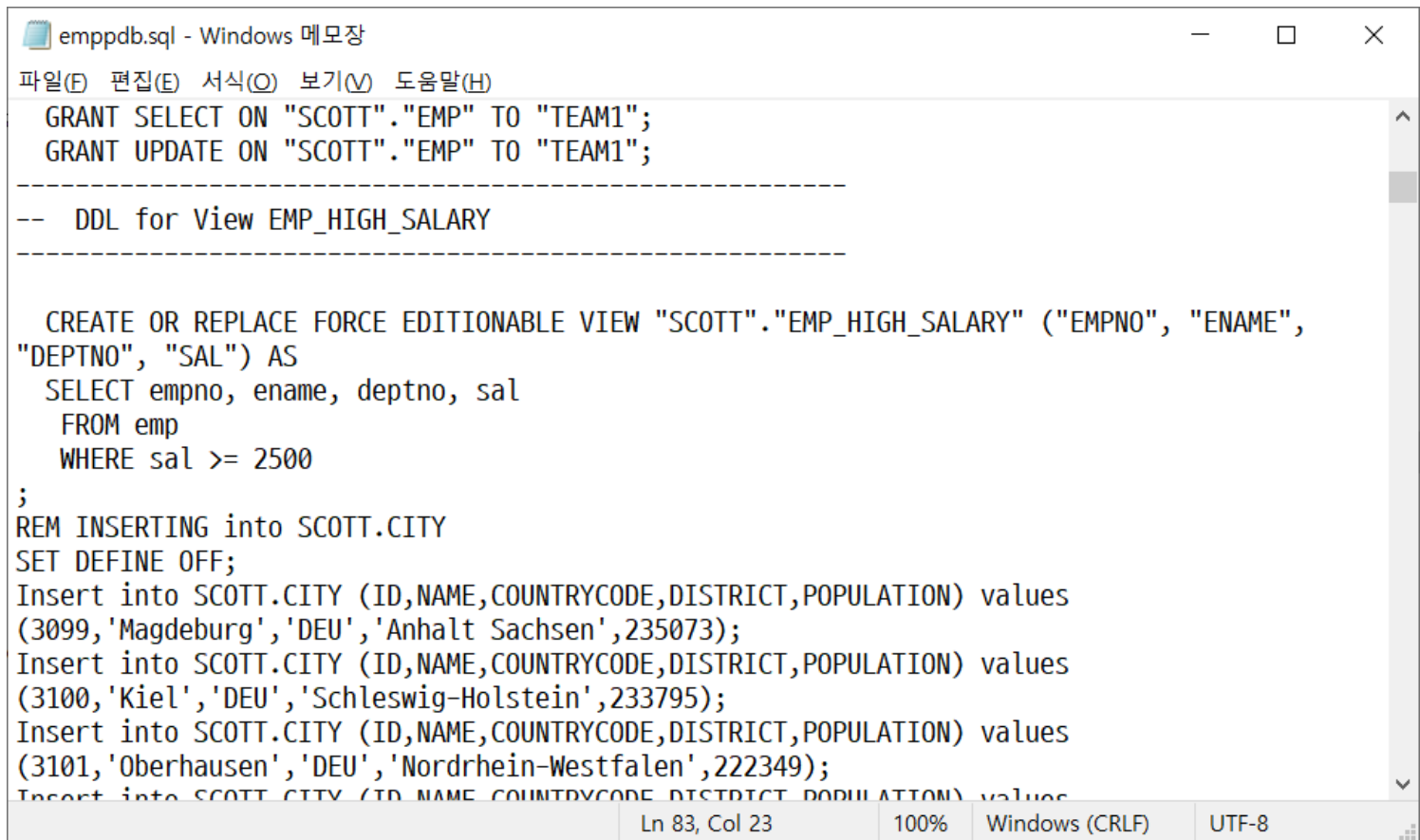


The screenshot shows the SQL Developer interface with the 'empdpdb.sql' script open. The script contains the following SQL code:

```
-- 2022-08-12 생성 - 6월 6일 - 12월 8일 - 2022  
  
-- DDL for Table CITY  
  
CREATE TABLE "SCOTT"."CITY"  
(  
  "ID" NUMBER(5,0),  
  "NAME" VARCHAR2(50),  
  "COUNTRYCODE" CHAR(3) DEFAULT NULL,  
  "DISTRICT" VARCHAR2(50),  
  "POPULATION" NUMBER(8,0) DEFAULT NULL  
) SEGMENT CREATION IMMEDIATE  
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255  
NOCOMPRESS LOGGING  
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
```


실습: 데이터베이스 내보내기(Export)

- ⑧ 이제 내보내기를 저장한 파일을 열어서 내용을 살펴보도록 한다.
 - 확장자가 .SQL인 파일은 메모장으로도 읽을 수 있다.
 - 파일 내용을 보면 데이터 구조를 설정하는 CREATE문과 데이터를 생성하기 위한 INSERT문이 보인다.



```
emppdb.sql - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
GRANT SELECT ON "SCOTT"."EMP" TO "TEAM1";
GRANT UPDATE ON "SCOTT"."EMP" TO "TEAM1";

-----

-- DDL for View EMP_HIGH_SALARY
-----

CREATE OR REPLACE FORCE EDITIONABLE VIEW "SCOTT"."EMP_HIGH_SALARY" ("EMPNO", "ENAME",
"DEPTNO", "SAL") AS
  SELECT empno, ename, deptno, sal
     FROM emp
    WHERE sal >= 2500
;
REM INSERTING into SCOTT.CITY
SET DEFINE OFF;
Insert into SCOTT.CITY (ID,NAME,COUNTRYCODE,DISTRICT,POPULATION) values
(3099,'Magdeburg','DEU','Anhalt Sachsen',235073);
Insert into SCOTT.CITY (ID,NAME,COUNTRYCODE,DISTRICT,POPULATION) values
(3100,'Kiel','DEU','Schleswig-Holstein',233795);
Insert into SCOTT.CITY (ID,NAME,COUNTRYCODE,DISTRICT,POPULATION) values
(3101,'Oberhausen','DEU','Nordrhein-Westfalen',222349);
Insert into SCOTT.CITY (ID,NAME,COUNTRYCODE,DISTRICT,POPULATION) values

Ln 83, Col 23      100%  Windows (CRLF)  UTF-8
```

실습: 데이터베이스 내보내기(Export)

Note.

- 데이터베이스의 내용을 SQL문의 형태로 저장하면 나중에 필요시 SQL문을 실행해 줌으로써 원래 데이터베이스를 복원할 수 있다.
- 또한 파일의 내용을 조금만 수정하면 MySQL이나 MS SQL SERVER와 같은 다른 DBMS 제품에서도 데이터베이스의 복원이 가능해 진다.

실습: 데이터베이스 내보내기(Export)

- 외부의 SQL 스크립트 파일 실행하기

- 데이터베이스의 내용을 SQL 스크립트 형태로 내보내기를 하여서 저장할수도 있지만 반대로 SQL 스크립트 파일의 내용을 불러와서 실행할 수도 있다.

- 두가지 방법

a) SQL 스크립트 파일의 내용을 복사 후 SQL Developer 워크시트에 붙여 넣기를 한 후 실행한다.

b) SQL 스크립트 파일을 바로 실행한다.

- 두번째 방법을 실습

실습: 데이터베이스 내보내기(Export)

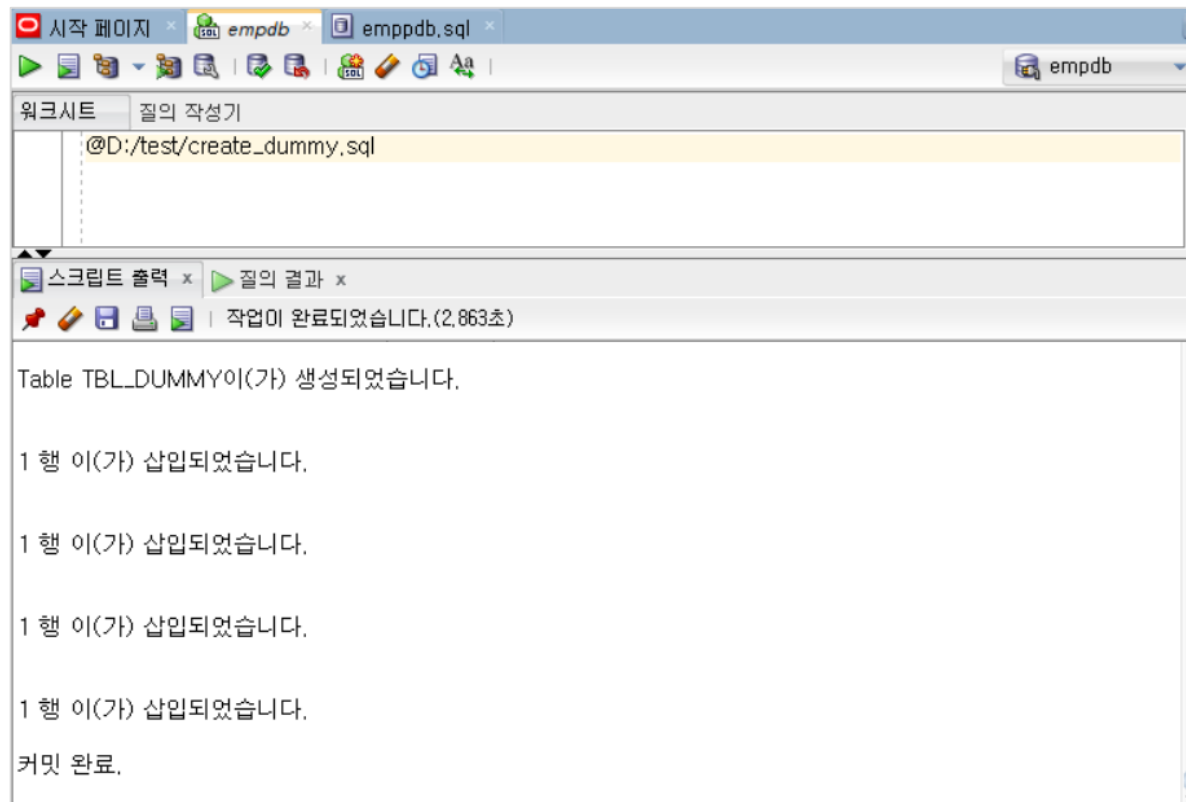
- ① 다음과 같이 create_dummy.sql 파일을 작성한다.

```
CREATE TABLE tbl_dummy (  
    fruit_name varchar(30)  
);  
INSERT INTO tbl_dummy VALUES ('APPLE') ;  
INSERT INTO tbl_dummy VALUES ('ORANGE') ;  
INSERT INTO tbl_dummy VALUES ('BANANA') ;  
INSERT INTO tbl_dummy VALUES ('MANGO') ;  
COMMIT ;
```

실습: 데이터베이스 내보내기(Export)

- ② create_dumy.sql 파일이 D:/test/ 에 저장되어 있다고 하면 SQL Developer 워크시트에서 다음과 같이 입력한 후 실행 아이콘을 클릭하면 create_dumy.sql에 저장된 내용이 실행 된다.

```
@D:/test/create_dummy.sql
```



실습: 데이터베이스 내보내기(Export)

- ③ tbl_dummy의 내용을 조회해 보면 입력 내용의 확인이 가능하다.

