Database Programming Lab02

Prof. 심준호(jshim@sookmyung.ac.kr/새힘관407)
T.A. 이은서(les97@sookmyung.ac.kr/명신관 316B)

Join & Sub-query

- Join
  - 하나 이상의 테이블을 연결하여 데이터를 검색하는 방법으로 보통 두 개 이상의 공통된 값 기본 키 및 외래 키 값을 사용하여 조인을 실행한다. 조인의 종류에는 Cartesian Product, Equi Join, Non-Equijoin, Self Join, Outer Join이 있다.
- Sub-query
  - 하나의 SQL문(주 질의 : Main Query)에 중첩된 SELECT문으로 서브 질의는 주 질의 이전에 한 번 실행되며, 결과는 주 질의에 의해 사용된다. 유형으로는 단일 행(Single Row) 서브 질의, 다중 행(Multiple Rows) 서브 질의, 다중 열(Multiple Columns) 서브 질의가 있다.

실습 1

** Lab01의 실습2에서 만든 테이블 참고

** 각각의 출력 전에 **show user;** 보여주세요.

(1) 과목번호 200011 인 수업을 듣는 학생의 학번, 이름, 학년, 전공 을 출력

(2) 과목번호 200013 인 수업을 가르치는 교수님의 교번, 이름, 대학, 전공 을 출력

Scott 계정

- 관리자 계정으로 돌아가기
- Scott 계정 비밀번호 ora로 변경 후 로그인
  - alter user **scott** identified by **ora**;
  - conn scott/ora;
  - show user;
  - select * from tab; 으로 DEPT, EMP 테이블 유무 확인!!

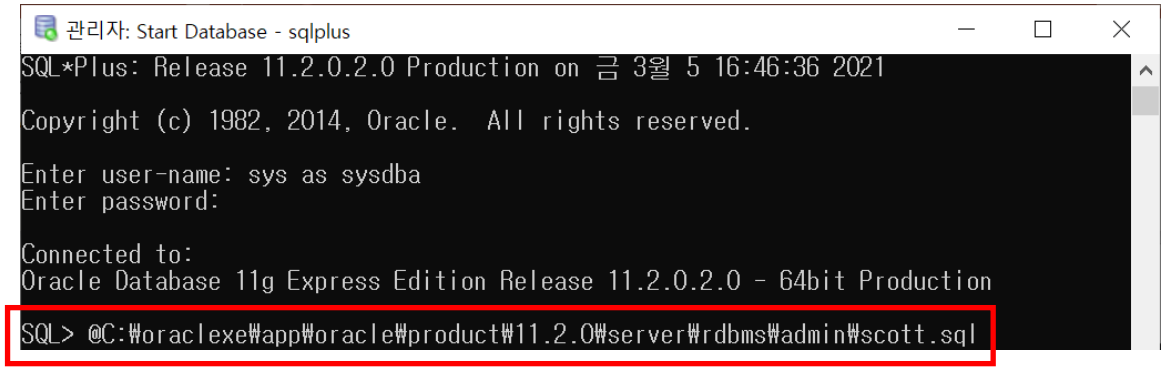
```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
BONUS                          TABLE
DEPT                           TABLE
EMP                            TABLE
SALGRADE                       TABLE
```

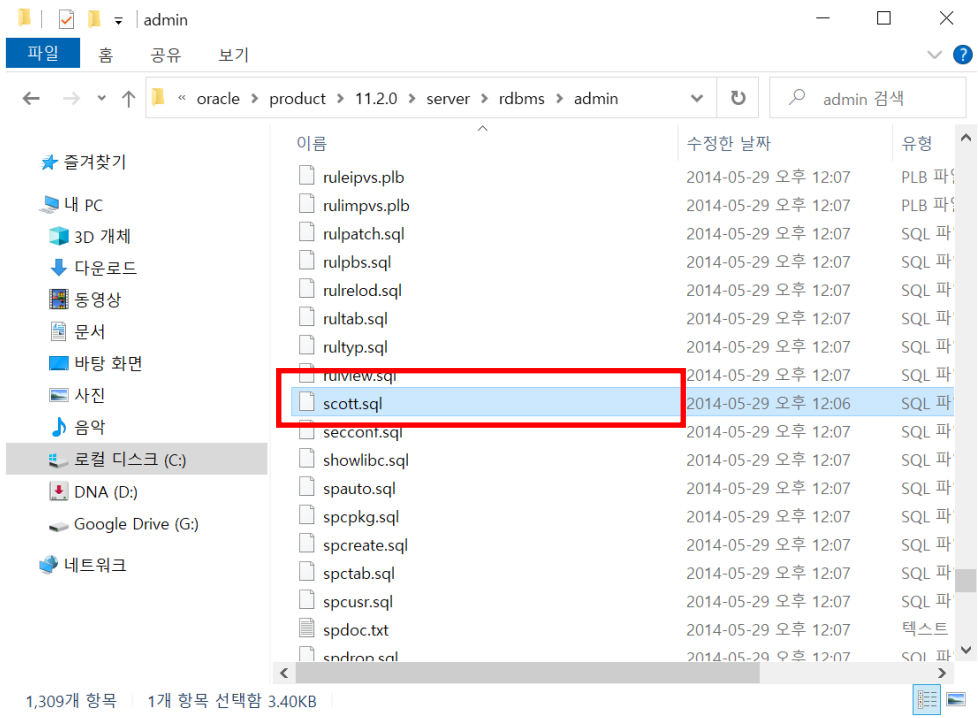
  - if) scott 계정 활성화 안된 경우
    - (관리자 계정에서) alter user scott identified by ora account unlock;

  - if) scott 계정이 없거나, 테이블이 존재하지 않을 경우 (다음 장)

Scott 계정 생성하기 (1)

- SQL Plus 접속 → sys 계정으로 로그인
- 오라클 폴더 내 에서 scott.sql 파일 확인
- SQL Plus에서 @scott.sql 경로 입력
- ex) @C:₩oraclexe₩app₩oracle₩product₩11.2.0₩server₩rdbms₩admin₩scott.sql
- Scott 계정 생성 완료





Scott 계정 생성하기 (2)

- Scott 계정 비밀번호 ora로 변경 후 로그인
  - alter user **scott** identified by **ora**;
  - conn scott/ora;
  - show user;
  - select * from tab; 으로 DEPT, EMP 테이블 유무 확인

```
SQL> @C:₩oraclexe₩app₩oracle₩product₩11.2.0₩server₩rdbms₩admin₩scott.sql
SQL> alter user scott identified by ora;

User altered.

SQL> conn scott/ora;
Connected.
SQL> show user;
USER is "SCOTT"
SQL>
```

실습 2

(1) 연봉을 1000미만으로 받는 사원이 소속된 부서 중 부서번호가 최소인 부서와 동일한 부서에서 근무하는 사원들의 정보를 출력하고 화면을 캡쳐하세요.

(2) 세일즈맨의 최저 급여를 받는 사원보다 많은 급여를 받는 'CLERK'인 직원들의 정보를 출력하고 화면을 캡쳐하세요.

(3) 각각 출력해 (sql문+결과창)캡쳐하고 ALL,ANY의 차이점을 간단히 **설명하세요**.

```
select * from emp where sal<ALL(SELECT SAL FROM EMP WHERE JOB='SALESMAN');
select * from emp where sal>ALL(SELECT SAL FROM EMP WHERE JOB='SALESMAN');
select * from emp where sal<ANY(SELECT SAL FROM EMP WHERE JOB='SALESMAN');
select * from emp where sal>ANY(SELECT SAL FROM EMP WHERE JOB='SALESMAN');
```

Integrity Constraints

- 테이블에 부적절한 자료가 입력되는 것을 방지하기 위해서 여러가지 규칙을 적용해 놓는 것으로 테이블에 행이 삽입, 갱신, 삭제 될 때마다 제약조건이 적용된다. 제약조건은 테이블이 생성된 후에 정의가 가능하고 일시적으로 DISABLE, ENABLE이 가능하다. 무결성 제약 조건의 종류로는 NOT NULL, PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK가 있다.

실습 3

* 다시 본인 계정(db학번)으로 로그인

** book, review 테이블 스노보드 파일 수정해서 사용

(1) 현재 사용자 보여주고, BOOK1 (title : not null, author, pub_year, publisher) 을 갖는 테이블을 생성하고, 테이블 구조를 보인 후, NOT NULL 제약 조건을 위배하는 INSERT 문을 이용하여 레코드 삽입을 시도한 것을 보여주고 조건을 만족하도록 insert문을 고쳐서 삽입해준다. (화면 캡처)

(2) 제약조건을 삭제하지 않고 , BOOK1 테이블의 NOT NULL 조건을 적용시키지 않으려면 어떻게 해야 하는가 ? (명령문만 직접 보고서에 써주세요.)

(1) Book2(book 테이블 수정)의 title을 외래키로 갖는 review테이블 생성하고 무결성 제약조건에 위배되는 레코드 삽입하고 이를 보여준다.(화면 캡쳐)

NULL 값 처리(1/3)

- **NULL 값이란 ?**
  - 아직 지정되지 않은 값
  - '0', ''(빈 문자), ' '(공백) 등과 다른 특별한 값
  - 비교 연산자로 비교 불가능 함
  - NULL 값 연산을 수행하면 결과 역시 NULL 값으로 반환이 됨

- **집계 함수를 사용할 때 주의할 점**

  : 대상 데이터를 **특정 그룹** (Group By) 으로 묶은 다음 이 그룹에 대해 총합 , 평균 , 최대값 , 최소값 등을 구하는 함수

  - **" NULL + 숫자 "** 연산의 결과는 NULL
  - 집계 함수 계산시 , NULL 이 포함된 행은 집계 계산에서 제외
  - 해당되는 행이 하나도 없는 경우 **SUM, AVG** 함수의 결과는 **NULL, COUNT** 함수 결과는 0

NULL 값 처리(2/3)

- **NULL 값에 대한 연산과 집계 함수**
- [table] Mybook

```
CREATE TABLE Mybook (
    bookid number,
    price number
);
insert into Mybook values(1, 10000);
insert into Mybook values(2, 20000);
insert into Mybook values(3, NULL);
```

bookid	price
1	10000
2	20000
3	NULL

```
SELECT price+100
FROM Mybook
WHERE bookid=3;
```

PRICE+100

```
SELECT SUM(price), AVG(price), COUNT(*), COUNT(price)
FROM Mybook;
```

SUM(PRICE)	AVG(PRICE)	COUNT(*)	COUNT(PRICE)
30000	15000	3	2

```
SELECT SUM(price), AVG(price), COUNT(*)
FROM Mybook
WHERE bookid >= 4;
```

SUM(PRICE)	AVG(PRICE)	COUNT(*)
		0

NULL 값 처리(3/3)

- NULL 을 확인하는 방법 : IS NULL / IS NOT NULL
  - NULL 값을 찾을 때 : '=' 연산자가 아닌 ' IS NULL ' 을 사용
  - NULL 값이 아닌 값을 찾을 때 : '<>' 연산자가 아닌 ' IS NOT NULL ' 을 사용

[TABLE] Mybook

bookid	price
1	10000
2	20000
3	NULL

```
SELECT *
FROM Mybook
WHERE price IS NULL;
```

```
   BOOKID      PRICE
---------- ----------
         3
```

```
SELECT *
FROM Mybook
WHERE price='';
```

```
SQL> select * from mybook where price = '';
선택된 레코드가 없습니다.
```

실습 4

- Mybook 테이블을 생성하고 NULL 에 관한 다음 SQL문 실행해보세요. (SQL+결과) 캡쳐하고 NULL 에 대한 개념을 정리하며 각각의 문제마다 간단히 설명하세요.

Mybook

bookid	price
1	10000
2	20000
3	NULL

(1) SELECT bookid, NVL(price, 0)
FROM Mybook;

(2) SELECT *
FROM Mybook
WHERE price IS NULL;

(3) SELECT *
FROM Mybook;
WHERE price =‘ ’;

(4) SELECT bookid, price+100
FROM Mybook;

(5) SELECT SUM(price), AVG(price), COUNT(*)
FROM Mybook;
WHERE bookid >= 4;

(6) SELECT COUNT(*), COUNT(price)
FROM Mybook;

(7) SELECT SUM(price), AVG(price)
FROM Mybook;

Submission

- Lab02.docx
  - 제출파일명: **학번_이름.docx**

- Due
  - **2021.03.29(월) 11:55pm전 까지**