Chapter 2. SELECT문 기초

이번 장에서는 테이블의 데이터를 검색하는데 사용되는 SELECT 문장을 살펴보고, SQL 문장과 SQL*Plus간의 동작원리 및 SQL*Plus 자체 명령어를 알아보도록 한다.

SELECT

테이블에서 데이터를 검색하기 위해서는 SELECT 문장을 사용하여야 하며 기본 문법은 다음과 같다.

```
SELECT *|{[DISTINCT] column|expression [alias], ... }
FROM table;
```

SELECT는 데이터를 검색하는 문장으로 대상 테이블에 어떠한 조작도 수행하지 않는다. 먼저, Oracle 데이터베이스에 SQL*Plus를 이용하여 SCOTT 계정으로 접속한다. 데이터를 검색하기 위해서는 어떤 테이블들이 저장되어 있는지 확인해야 하는데 테이블 목록을 확인하는 명령어는 다음과 같다.

```
SQL> SELECT * FROM TAB;

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

SQL>
```

출력 결과를 보면 SCOTT 계정이 사용할 수 있는 객체는 BONUS, DEPT, EMP, SALGRADE이며 모두 TABLE 임을 알 수 있다. 그렇다면 DEPT 테이블에는 어떤 컬럼이 있는지 확인해보자.

```
SQL> DESC DEPT
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                    NOT NULL NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
```

DEPT 테이블에는 DEPTNO, DNAME, LOC 컬럼이 있으며 해당 컬럼에 NULL 값이 입력될 수 있는지 여부와 각 컬럼의 데이터 타입이 나타나 있다. 여기서, 사용한 DESC 명령은 SELECT와 같은 표준 SQL 문장은 아니므로 다른 데이터베이스에서는 사용할 수 없다. DESC 명령은 SQL*Plus나 iSQL*Plus의 자체 명령어이다.

테이블내의 모든 데이터를 검색하려면 SELECT 뒤에 * 기호를 붙이고 FROM 뒤에 테이블을 기술해준다.

```
SQL> SELECT * FROM DEPT;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

테이블내의 특정 컬럼만을 검색하려면 SELECT 뒤에 해당 컬럼을 차례로 기술한다. 컬럼을 나열할 때는 쉼표(,)로 구분해야만 한다.

```
SQL> SELECT EMPNO, ENAME, JOB, HIREDATE FROM EMP;

     EMPNO ENAME      JOB       HIREDATE
---------- ---------- --------- --------
      7369 SMITH      CLERK     80/12/17
      7499 ALLEN      SALESMAN  81/02/20
      7521 WARD       SALESMAN  81/02/22
        ...
      7900 JAMES      CLERK     81/12/03
      7902 FORD       ANALYST   81/12/03
      7934 MILLER     CLERK     82/01/23

14 개의 행이 선택되었습니다.
```

SQL 문장의 작성 지침

SQL 문장을 작성할 때, 다음과 같은 규칙과 지침을 준수하면, SQL 문장의 가독성(Readability)이 향상될 뿐 아니라 긴 문장의 디버깅 작업도 수월해진다.

· SQL 문장은 대소문자를 구별하지 않는다.
· SQL 문장은 여러 줄에 걸쳐서 작성할 수 있다.
· 키워드(SELECT, FROM 등)는 줄여 쓸 수 없고, 여러 줄에 걸쳐서 작성할 수 없다.
· 일반적으로 각 키워드는 별도 라인에 작성하도록 한다.
· 각 라인에 들여쓰기를 하면 이해하기가 쉽다.

컬럼의 정렬 방식

SELECT 문장의 실행 결과는 각 컬럼의 데이터 타입에 따라 정렬 형태가 달라진다.

■ iSQL*Plus
컬럼명은 가운데 정렬되며 대문자로 표시되는 반면, 문자 타입과 날짜 타입 컬럼의 값은 좌측 정렬되며 숫자 타입 컬럼의 값은 우측 정렬된다.

EMPNO	ENAME	HIREDATE	SAL
7369	SMITH	80/12/17	800
7499	ALLEN	81/02/20	1600
7521	WARD	81/02/22	1250
7566	JONES	81/04/02	2975
7654	MARTIN	81/09/28	1250
7698	BLAKE	81/05/01	2850
7782	CLARK	81/06/09	2450
7788	SCOTT	87/04/19	3000
7839	KING	81/11/17	5000
7844	TURNER	81/09/08	1500
7876	ADAMS	87/05/23	1100
7900	JAMES	81/12/03	950
7902	FORD	81/12/03	3000
7934	MILLER	82/01/23	1300

그림 2-1. 컬럼의 정렬 방식

■ SQL*Plus

마찬가지로 문자 타입과 날짜 타입 컬럼 값은 좌측 정렬되며 숫자 타입 컬럼 값은 우측 정렬된다. 컬럼명도 같은 방식으로 정렬된다.

```
SQL> SELECT EMPNO, ENAME, HIREDATE, SAL FROM EMP;

     EMPNO ENAME      HIREDATE          SAL
---------- ---------- ---------- ----------
      7369 SMITH      80/12/17          800
      7499 ALLEN      81/02/20         1600
      7521 WARD       81/02/22         1250
        ...
      7900 JAMES      81/12/03          950
      7902 FORD       81/12/03         3000
      7934 MILLER     82/01/23         1300

14 개의 행이 선택되었습니다.
```

산술 연산자

SQL 문장내의 숫자 타입 및 날짜 타입 데이터에는 +, -, *, / 와 같은 산술연산자를 사용할 수도 있다.

```
SQL> SELECT EMPNO, ENAME, SAL * 1.1 FROM EMP;

     EMPNO ENAME      SAL*1.1
---------- ---------- ----------
      7369 SMITH           880
      7499 ALLEN          1760
      7521 WARD           1375
        ...
      7900 JAMES          1045
      7902 FORD           3300
      7934 MILLER         1430

14 개의 행이 선택되었습니다.
```

연산자의 우선순위는 *, / 가 +, - 에 우선하며, 우선순위가 동등할 때는 좌측 연산자부터 우측 연산자로 연산이 진행된다. 우선순위를 강제적으로 지정하려면 괄호를 사용하도록 한다. 아래 SQL 문장들은 연산의 우선순위가 다르기 때문에 그 결과도 서로 다름을 알 수 있다.

```
SQL> SELECT EMPNO, ENAME, 1.1 * SAL + 200 FROM EMP;

     EMPNO ENAME      1.1*SAL+200
---------- ---------- -----------
      7369 SMITH             1080
      7499 ALLEN             1960
        ...
SQL> SELECT EMPNO, ENAME, 1.1 * (SAL + 200) FROM EMP;

     EMPNO ENAME      1.1*(SAL+200)
---------- ---------- -------------
      7369 SMITH               1100
      7499 ALLEN               1980
        ...
```

NULL 값의 정의 및 처리

NULL 이란 이용 불가능한(Unavailable), 지정되지 않은(Unassigned), 알 수 없는(Unknown), 적용할 수 없는(Inapplicable) 값을 의미한다. 한 가지 기억해야 할 것은 NULL은 0(Zero) 또는 공백(Space)과는 다르다는 점이다. 아래 SQL 문장을 수행해보면 값이 표시되지 않는 데이터가 NULL이다.

```
SQL> SELECT EMPNO, ENAME, SAL, COMM FROM EMP;

     EMPNO ENAME             SAL       COMM
---------- ---------- ---------- ----------
      7369 SMITH             800
      7499 ALLEN            1600        300
      7521 WARD             1250        500
      7566 JONES            2975
      7654 MARTIN           1250       1400
        ...
      7902 FORD             3000
      7934 MILLER           1300
```

NULL이 산술연산에 포함되면 그 결과는 연산에 상관없이 무조건 NULL이 된다.

```
SQL> SELECT EMPNO, ENAME, COMM, COMM + 100 FROM EMP;

     EMPNO ENAME           COMM   COMM+100
---------- ---------- ---------- ----------
      7369 SMITH
      7499 ALLEN             300        400
      7521 WARD              500        600
      7566 JONES
        ...
      7902 FORD
      7934 MILLER
```

컬럼명에 별칭 사용

SELECT 문장의 실행 결과를 살펴보면 최상단의 컬럼명은 테이블내의 컬럼명과 동일하게 표시된다. 그러나 컬럼 별칭을 사용하면 최상단의 컬럼명을 별도로 지정한 별칭으로 표시되도록 할 수가 있다. 컬럼 별칭은 다음과 같은 특징을 갖는다.

- 결과 출력시 컬럼명을 변경한다.
- 산술 연산된 컬럼과 같이 사용하면 컬럼명을 보기 좋게 표현할 수 있다.
- SELECT 문장의 컬럼명 뒤에 AS라는 키워드를 붙이고 별칭을 작성하는데, AS는 생략되어도 상관 없다.
- 별칭에 공백이나 특수문자가 포함되거나 대소문자 구분이 필요한 경우 별칭에 " "를 붙인다.

컬럼 별칭을 사용하는 방법은 다음과 같다.

```
SQL> SELECT EMPNO AS 사번, ENAME AS 성명, SAL AS 급여 FROM EMP;

      사번 성명             급여
---------- ---------- ----------
      7369 SMITH             800
      7499 ALLEN            1600
      7521 WARD             1250
        ...
      7900 JAMES             950
      7902 FORD             3000
      7934 MILLER           1300

14 개의 행이 선택되었습니다.
```

AS 키워드는 생략 가능하며, 공백이 포함된 경우는 " "를 붙인다.

```
SQL> SELECT EMPNO AS "사 번", ENAME AS "성 명", SAL * 12 AS "연 봉" FROM EMP;

     사 번 성 명          연 봉
---------- ---------- ----------
      7369 SMITH            9600
      7499 ALLEN           19200
      7521 WARD            15000
        ...
      7900 JAMES           11400
      7902 FORD            36000
      7934 MILLER          15600

14 개의 행이 선택되었습니다.
```

연결 연산자(Concatenation Operator)

연결 연산자는 여러 개의 문자열을 한 개의 문자열로 결합시키는 연산자로서 다음과 같은

특징이 있다.

· 문자 타입 컬럼간 또는 문자 타입 컬럼과 문자열을 연결할 필요가 있을 때 사용한다.
· 컬럼간 또는 컬럼과 문자열 사이에 ||를 입력한다.
· 연결 연산자에 의한 연산 결과는 문자열이다.

```
SQL> SELECT ENAME||JOB FROM EMP;

ENAME||JOB
-------------------
SMITHCLERK
ALLENSALESMAN
WARDSALESMAN
...
JAMESCLERK
FORDANALYST
MILLERCLERK

14 개의 행이 선택되었습니다.
```

리터럴(Literal)

리터럴은 SELECT 문장에 포함된 컬럼명 또는 컬럼별칭 이외의 문자값, 숫자값, 날짜이며 다음과 같은 특징을 갖는다.

· 리터럴은 SELECT 문장 뒤에 기술되는 문자값, 숫자값, 날짜를 의미한다.
· 날짜와 문자 리터럴은 반드시 ‘ ’를 붙여야 한다.
· SELECT문에 포함된 각 리터럴 문자열은 결과 출력시 각각의 행에 한번씩 나타난다.

```
SQL> SELECT ENAME||'의 직급은 '||JOB||'이다' AS "사원별 직급" FROM EMP;

사원별 직급
--------------------------------
SMITH의 직급은 CLERK이다
ALLEN의 직급은 SALESMAN이다
WARD의 직급은 SALESMAN이다
...
JAMES의 직급은 CLERK이다
FORD의 직급은 ANALYST이다
MILLER의 직급은 CLERK이다

14 개의 행이 선택되었습니다.
```

DISTINCT를 이용한 중복 데이터 제거

다음과 같은 SQL 문장을 수행한 결과를 보면 해당 결과가 중복되어 있음을 알 수 있다.

```
SQL> SELECT JOB FROM EMP;

JOB
---------
CLERK
SALESMAN
SALESMAN
MANAGER
SALESMAN
MANAGER
MANAGER
ANALYST
PRESIDENT
SALESMAN
CLERK
CLERK
ANALYST
CLERK

14 개의 행이 선택되었습니다.
```

위 결과는 정상적인 것이지만 JOB 컬럼의 값이 중복되어 있어서 JOB의 종류를 쉽게 파악할 수가 없다. 이러한 경우에 DISTINCT 키워드를 사용하면 중복된 결과를 제거하고 출력하기 때문에 JOB의 종류를 쉽게 확인할 수 있다.

```
SQL> SELECT DISTINCT JOB FROM EMP;

JOB
---------
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN
```

SQL과 SQL*Plus의 상호작용

SQL은 Oracle 서버와 직접 통신하는 명령어로서 사용자가 SQL 문장을 SQL*Plus에 입력하면 SQL 버퍼라는 메모리에 저장되고 새로운 SQL 문장이 입력될 때까지 버퍼에 남아 있게 된다.

SQL*Plus는 사용자가 입력한 SQL 문장을 인식하고 데이터베이스 서버에 해당 문장을 전송하여 사용자가 원하는 작업을 수행되도록 해주는 프로그램이다. 뿐만 아니라, SQL*Plus는 자체 명령어를 다수 포함하고 있다. (예, DESC)

다음은 SQL 문장과 SQL*Plus 명령어의 차이를 정리한 것이다.

표 2-1. SQL 문장과 SQL*Plus 명령어의 차이

SQL	SQL*Plus
데이터 접근을 위해 Oracle 서버와 통신하는 언어	SQL 문장을 인식하여 Oracle 서버에 보내주는 역할
ANSI 표준 SQL에 기초함	SQL 문장을 실행하기 위한 Oracle 인터페이스
데이터베이스 내의 데이터 또는 테이블을 변경함	데이터베이스내의 데이터 조작을 허용하지 않음
SQL 버퍼에 저장됨	한번에 한개의 라인에 입력하며 SQL 버퍼에 저장되지 않음
여러개의 라인을 연결하기 위한 별도의 문자가 필요 없음	"-" 기호를 사용하여 SQL*Plus 명령어의 여러 개의 라인에 걸쳐서 기술할 수 있음
축약될 수 없음	축약될 수 있음
문장 끝에 종료 문자가 필요함	명령어의 끝에 종료 문자가 불필요함
데이터 포맷팅을 위해 별도 기능을 사용	데이터의 포맷팅에 주로 사용

SQL*Plus 명령어

다음은 자주 사용되는 SQL*Plus 명령어이다.

■ 테이블의 구조 확인하기

테이블의 구조를 확인하는 명령어는 DESCRIBE이지만, 축약해서 DESC로 기술해도 무관하다.

```
SQL> DESC EMP
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------
 EMPNO                                     NOT NULL NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)
```

■ SQL*Plus 편집 명령

편집 명령은 버퍼내에 저장된 SQL 문장을 수정하는 명령어이다.

표 2-2. SQL*Plus 명령어 목록

명령	설명
A[PPEND] *text*	버퍼내의 마지막 라인의 끝에 *text*를 추가
C[HANGE] / *old* / *new*	현재 라인에서 *old* 문자열을 *new* 문자열로 교체
C[HANGE] / *text* /	현재 라인에서 *text* 문자열을 삭제
CL[LEAR] BUFF[ER]	버퍼내의 모든 내용을 삭제
DEL	현재 라인을 삭제
I[NPUT]	버퍼내의 현재 라인에 새로운 라인 입력
I[NPUT] *text*	현재 라인에서 *text*를 삽입
L[IST]	버퍼내의 모든 라인을 검색
L[IST] *n*	버퍼내의 *n*번째 라인을 검색
L[IST] *m n*	버퍼내의 *m*~*n*번째 라인을 검색
R[UN]	버퍼 내용을 실행
n	*n*번째 라인을 현재 라인으로 지정
n text	*n*번째 라인을 *text*로 교체
0 text	버퍼의 앞에 *text*를 삽입

아래는 SQL 버퍼내에 저장된 문장을 수정하여 재실행하는 예이다.

```
SQL> SELECT EMPNO, ENAME, JOB, SAL
  2  FROM EMP;

     EMPNO ENAME      JOB              SAL
---------- ---------- --------- ----------
      7369 SMITH      CLERK            800
      7499 ALLEN      SALESMAN        1600
        ...
      7902 FORD       ANALYST         3000
      7934 MILLER     CLERK           1300

14 개의 행이 선택되었습니다.

SQL> L
  1  SELECT EMPNO, ENAME, JOB, SAL
  2* FROM EMP
SQL> 1
  1* SELECT EMPNO, ENAME, JOB, SAL
SQL> A ,COMM
  1* SELECT EMPNO, ENAME, JOB, SAL,COMM
SQL> R
  1  SELECT EMPNO, ENAME, JOB, SAL,COMM
  2* FROM EMP

     EMPNO ENAME      JOB              SAL       COMM
---------- ---------- --------- ---------- ----------
      7369 SMITH      CLERK            800
      7499 ALLEN      SALESMAN        1600        300
        ...
      7902 FORD       ANALYST         3000
      7934 MILLER     CLERK           1300

14 개의 행이 선택되었습니다.
```

■ SQL*Plus 파일 명령

파일 명령은 SQL*Plus 버퍼 내용을 저장, 불러오기, 편집 등의 작업을 수행하는 명령어이다.

표 2-3. SQL*Plus 파일 명령

명령	설명
SAV[E] *filename*[.*ext*] [REP[LACE]|APP[END]]	버퍼의 내용을 외부 파일로 저장함. REPLACE는 지정된 파일명과 같은 파일이 외부에 존재하면 파일을 교체하며, APPEND는 기존의 파일 뒤에 내용을 추가함. 디폴트 확장자는 sql
GET *filename*[.*ext*]	외부 파일의 내용을 버퍼로 읽음
STA[RT] *filename*[.*ext*]	외부 파일을 실행
@*filename*	외부 파일을 실행(START 명령과 동일)
ED[IT]	버퍼를 수정하기 위해 지정된 편집기를 실행하고 버퍼의 내용을 읽음
ED[IT] [*filename*[.*ext*]]	외부 파일을 수정하기 위해 지정된 편집기를 실행
SPO[OL] [*filename*[.*ext*]] OFF|OUT	SQL*Plus 내의 모든 실행 결과를 외부 파일로 저장함. OFF는 저장을 종료하며, OUT은 마찬가지로 저장을 종료하고 프린터로 전송
EXIT	SQL*Plus를 종료

버퍼 내용을 저장하고 재실행하는 방법은 다음과 같다.

```
SQL> SELECT EMPNO, ENAME, JOB, SAL, COMM FROM EMP;

     EMPNO ENAME      JOB              SAL       COMM
---------- ---------- --------- ---------- ----------
      7369 SMITH      CLERK            800
      7499 ALLEN      SALESMAN        1600        300
      7521 WARD       SALESMAN        1250        500
        ...
      7900 JAMES      CLERK            950
      7902 FORD       ANALYST         3000
      7934 MILLER     CLERK           1300

14 개의 행이 선택되었습니다.

SQL> SAVE C:WEMP.SQL
file C:WEMP.SQL(이)가 생성되었습니다
SQL> START C:WEMP.SQL

     EMPNO ENAME      JOB              SAL       COMM
---------- ---------- --------- ---------- ----------
      7369 SMITH      CLERK            800
      7499 ALLEN      SALESMAN        1600        300
      7521 WARD       SALESMAN        1250        500
        ...
      7900 JAMES      CLERK            950
      7902 FORD       ANALYST         3000
      7934 MILLER     CLERK           1300

14 개의 행이 선택되었습니다.
```

화면의 내용을 모두 저장하는 방법은 다음과 같다.

```
SQL> SPOOL C:WEMP.TXT
SQL> SELECT EMPNO, ENAME, JOB, SAL FROM EMP;

     EMPNO ENAME      JOB              SAL
---------- ---------- --------- ----------
      7369 SMITH      CLERK            800
      7499 ALLEN      SALESMAN        1600
      7521 WARD       SALESMAN        1250
      7566 JONES      MANAGER         2975
      7654 MARTIN     SALESMAN        1250
      7698 BLAKE      MANAGER         2850
      7782 CLARK      MANAGER         2450
      7788 SCOTT      ANALYST         3000
      7839 KING       PRESIDENT       5000
      7844 TURNER     SALESMAN        1500
      7876 ADAMS      CLERK           1100
      7900 JAMES      CLERK            950
      7902 FORD       ANALYST         3000
      7934 MILLER     CLERK           1300

14 개의 행이 선택되었습니다.

SQL> SPOOL OFF
```

저장된 EMP.TXT를 확인해보자

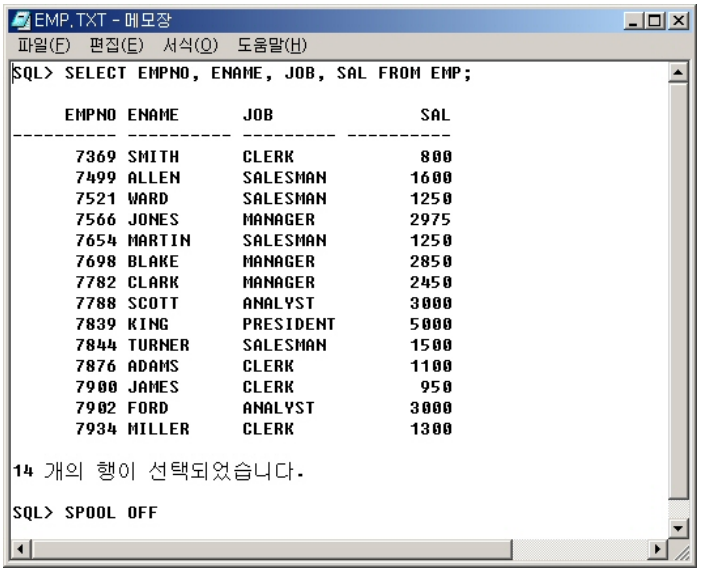

그림 2-2. 스풀된 내용 확인

iSQL*Plus

Oracle 9i에서부터 추가된 iSQL*Plus는 과거 SQL*Plus를 대체하는 새로운 도구이다. Oracle 데이터베이스와 함께 설치되는 Apache 웹 서버를 이용하면 사용자 컴퓨터에 SQL*Plus를 일일이 설치하지 않고 운영체제에 포함된 브라우저로 Oracle을 사용할 수 있다. iSQL*Plus는 SQL*Plus와는 달리 버퍼가 존재하지 않으므로 SQL*Plus의 편집명령은

사용할 수 없지만, 직접 iSQL*Plus 화면에서 SQL 문장을 수정 할 수 있다. 또한, SQL*Plus에서 사용하는 파일 명령어도 iSQL*Plus에서는 GUI 기반으로 제공되므로 단순히 해당 버튼을 클릭하여 SQL*Plus에서 수행하던 작업을 거의 유사하게 수행할 수 있다. iSQL*Plus가 GUI 환경으로 편리하기는 하지만 저자와 같이 텍스트 기반의 SQL*Plus 환경에 익숙한 작업자는 여전히 SQL*Plus를 선호하기 때문에 앞으로도 SQL*Plus는 계속 지원될 예정이다.

■ iSQL*Plus에서 스크립트 저장하기

1. iSQL*Plus에 접속하여, 다음과 같이 SQL 문장을 입력하고 **실행** 버튼을 클릭한다.

그림 2-3 iSQL*Plus 작업환경

2. **스크립트 저장** 버튼을 클릭하면, 파일 다운로드 대화상자가 나타나고 **저장** 버튼을 클릭하여 저장할 파일이름을 저장한다.

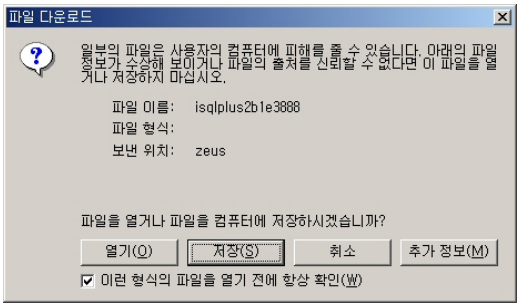

그림 2-4 파일 다운로드 대화상자

■ iSQL*Plus에서 스크립트 불러오기

1. iSQL*Plus에 접속하여, 불러올 파일 이름을 **파일 또는 URL**에 입력한다. 또한, **찾아보기** 버튼을 클릭하여 검색도 가능하다.

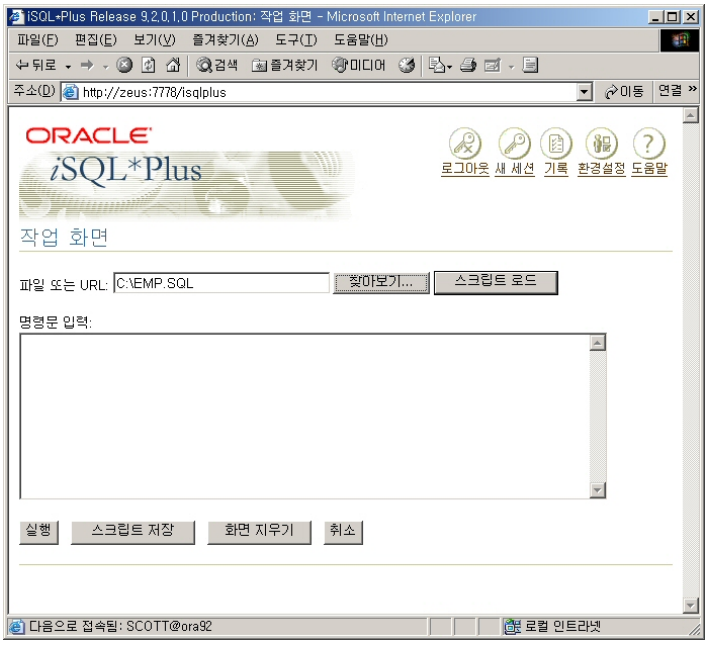

그림 2-5. 스크립트 파일 지정

2. **스크립트 로드** 버튼을 클릭하여 파일 내용을 명령문 입력 창으로 불러 올 수 있다.

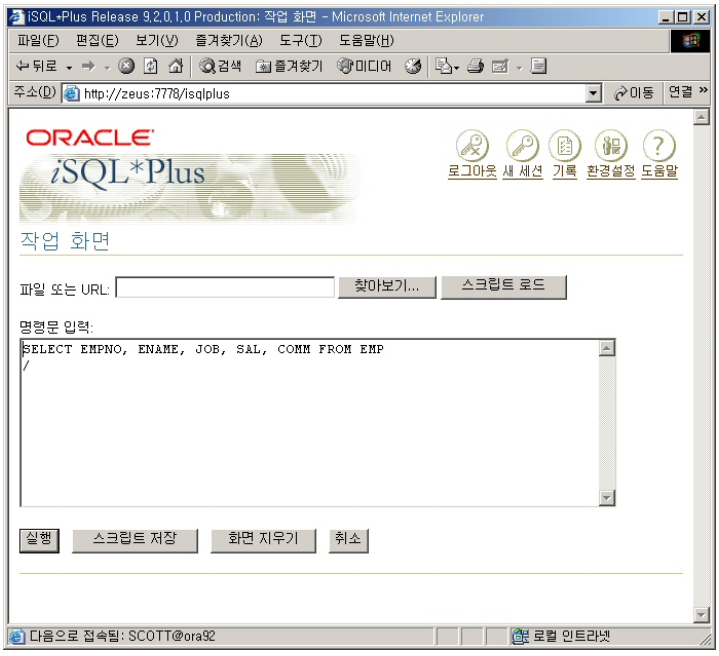

그림 2-6. 스크립트 로드

복습

1. 테이블의 목록을 확인할 수 있는 SQL 문장은 무엇인가?

2. 테이블의 구조를 확인할 수 있는 SQL*Plus 명령어는 무엇인가?

3. 사원 테이블에서 직원들의 연봉(SAL * 12)을 계산하여, 컬럼명은 “연 봉”으로 출력하는 SQL 문장을 작성하라.

4. NULL에 대한 설명 중 올바른 것을 모두 고르시오?
   a. NULL은 공백(space)을 의미한다.
   b. NULL과의 연산은 NULL이다.
   c. NULL은 숫자로 zero(0)을 의미한다.
   d. NULL은 Unknown, Inapplicable의 의미를 갖는다.

5. 사원 테이블을 이용하여 다음과 같은 결과를 얻을 수 있는 SQL 문장을 작성하라.

```
사원정보
------------------------------------------------------------------------
SMITH의 업무는 CLERK이고 급여는 800만원입니다
ALLEN의 업무는 SALESMAN이고 급여는 1600만원입니다
WARD의 업무는 SALESMAN이고 급여는 1250만원입니다
JONES의 업무는 MANAGER이고 급여는 2975만원입니다
MARTIN의 업무는 SALESMAN이고 급여는 1250만원입니다
BLAKE의 업무는 MANAGER이고 급여는 2850만원입니다
CLARK의 업무는 MANAGER이고 급여는 2450만원입니다
SCOTT의 업무는 ANALYST이고 급여는 3000만원입니다
KING의 업무는 PRESIDENT이고 급여는 5000만원입니다
TURNER의 업무는 SALESMAN이고 급여는 1500만원입니다
ADAMS의 업무는 CLERK이고 급여는 1100만원입니다
JAMES의 업무는 CLERK이고 급여는 950만원입니다
FORD의 업무는 ANALYST이고 급여는 3000만원입니다
MILLER의 업무는 CLERK이고 급여는 1300만원입니다

14 개의 행이 선택되었습니다.
```