Chapter 4. 단일행 함수

함수는 SQL 문장을 좀 더 강력하게 만들어 준다. 이 장에서는 함수 중에서 단일행 함수인 문자 함수, 숫자 함수, 날짜 함수, 데이터 타입 변환 함수에 대하여 살펴보고 이러한 함수들을 SELECT 문장에서 사용하는 방법을 알아본다.

함수의 종류

함수에는 단일행 함수(Single-row function)와 집계 함수라고 부르는 복수행 함수(Multiple-row function)의 두 종류가 있다. SQL 문장에서 단일행 함수는 모든 행에 대하여 각각 적용되어 행의 개수와 동일한 개수의 결과를 리턴해주는 반면, 집계 함수는 검색되는 모든 행에 한번만 적용되고 한건의 결과를 리턴한다. 이번 장에서는 단일행 함수에 대하여 설명하고 집계 함수는 추후에 살펴보기로 한다.

단일행 함수는 다음과 같은 기능이 있다.
· 데이터를 조작한다.
· 인자들을 입력받아 하나의 결과를 리턴한다.
· 각각의 행에 대하여 적용된다.
· 각 행별 하나의 결과를 리턴한다.
· 함수의 결과로서 데이터의 타입이 변경될 수도 있다.
· 함수의 중첩이 가능하다.
· 함수의 인자는 컬럼 또는 수식이 될 수 있다.

단일행 함수는 다음과 같은 종류가 있다.
· 문자 함수
· 숫자 함수
· 날짜 함수
· 변환 함수
· 일반 함수

문자 함수

문자 함수는 문자 타입의 컬럼 또는 값을 입력으로 받아서 그 결과로서 문자 타입의 값 또는 숫자 타입의 값을 리턴해주는 함수이다. 문자 함수는 대소문자 조작 함수와 문자열 조작 함수의 두 종류이며 간략하게 정리하면 다음과 같다.

표 4-1. 대소문자 조작 함수

함수	설명
LOWER(*column*| *expression*)	영문자를 소문자로 변환
UPPER(*column*| *expression*)	영문자를 대문자로 변환
INITCAP(*column*| *expression*)	영문자열의 첫 번째 문자를 대문자로 변환
CONCAT(*column1*| *expression1*, *column2*| *expression2*)	문자열을 결합, ||의 결과와 동등
SUBSTR(*column*| *expression*, *m* [,*n*])	입력된 문자열의 *m*번째 문자부터 *n*개의 문자열을 추출

■ LOWER

LOWER는 입력된 문자열을 소문자로 변환하여 리턴한다.

```
SQL> SELECT EMPNO, ENAME, LOWER(ENAME)
  2  FROM EMP
  3  WHERE EMPNO = 7369;

     EMPNO ENAME      LOWER(ENAM
---------- ---------- ----------
      7369 SMITH      smith
```

■ UPPER

UPPER는 입력된 문자열을 대문자로 변환하여 리턴한다.

```
SQL> SELECT EMPNO, ENAME, LOWER(ENAME), UPPER(ENAME)
  2  FROM EMP
  3  WHERE EMPNO = 7369;

     EMPNO ENAME      LOWER(ENAM UPPER(ENAM
---------- ---------- ---------- ----------
      7369 SMITH      smith      SMITH
```

단일행 함수는 SELECT 구문 뒤에만 기술할 수 있는 것이 아니라 WHERE, ORDER BY 등 구문에도 기술할 수 있다.

```
SQL> SELECT EMPNO, ENAME
  2  FROM EMP
  3  WHERE ENAME = UPPER('smith');

     EMPNO ENAME
---------- ----------
      7369 SMITH
```

■ INITCAP

INITCAP은 입력된 문자열의 첫 번째 문자를 대문자로 변환하여 리턴한다.

```
SQL> SELECT EMPNO, ENAME, INITCAP(ENAME)
  2  FROM EMP
  3  WHERE EMPNO = 7369;

     EMPNO ENAME      INITCAP(EN
---------- ---------- ----------
      7369 SMITH      Smith
```

■ CONCAT

입력된 문자열을 결합하여 리턴한다.

```
SQL> SELECT EMPNO, ENAME, JOB, CONCAT(ENAME, JOB)
  2  FROM EMP
  3  WHERE EMPNO = 7369;

     EMPNO ENAME      JOB       CONCAT(ENAME,JOB)
---------- ---------- --------- --------------------
      7369 SMITH      CLERK     SMITHCLERK
```

■ SUBSTR

입력된 문자열에서 지정된 문자열을 추출한다.

```
SQL> SELECT EMPNO, JOB, SUBSTR(JOB, 6, 3)
  2  FROM EMP
  3  WHERE EMPNO = 7499;

     EMPNO JOB       SUBSTR
---------- --------- ------
      7499 SALESMAN  MAN
```

표 4-2. 문자열 조작 함수

함수	설명
LENGTH(*column*|*expression*)	입력된 문자열의 전체 문자 개수를 리턴
INSTR(*column*|*expression*, '*string*', [,*m*], [*n*])	입력된 문자열의 *m*번째 문자부터 '*string*'이 *n*번째 나오는 위치를 리턴, *m*의 디폴트 값은 1
LPAD(*column*|*expression*, *n*, '*string*') RPAD(*column*|*expression*, *n*, '*string*')	전체 문자의 개수가 *n*개가 되도록 입력된 문자열의 왼쪽에 '*string*'을 추가 전체 문자의 개수가 *n*개가 되도록 입력된 문자열의 오른쪽에 '*string*'을 추가
TRIM(LEADING|TRAILING|BOTH *trim_character* FROM *trim_source*)	*trim_source*에서 *trim_character*를 제거
REPLACE(*text*, *search_string*, *replacement_string*)	*text*에서 *search_string*을 *replacement_string*으로 교체

■ LENGTH

입력된 문자열의 전체 문자 개수를 리턴한다.

```
SQL> SELECT EMPNO, ENAME, LENGTH(ENAME)
  2  FROM EMP
  3  WHERE EMPNO = 7499;

     EMPNO ENAME      LENGTH(ENAME)
---------- ---------- -------------
      7499 ALLEN                 5
```

■ INSTR

입력된 문자열에서 특정 문자열의 위치를 리턴한다.

```
SQL> SELECT EMPNO, JOB, INSTR(JOB, 'A', 1, 2)
  2  FROM EMP
  3  WHERE EMPNO = 7844;

     EMPNO JOB       INSTR(JOB,'A',1,2)
---------- --------- ------------------
      7844 SALESMAN                   7
```

■ LPAD / RPAD

주어진 문자의 좌측 / 우측에 특정 문자를 추가한다.

```
SQL> SELECT EMPNO, SAL, LPAD(SAL, 6, '*'), RPAD(SAL, 6, '*')
  2  FROM EMP;

     EMPNO        SAL LPAD(SAL,6,' RPAD(SAL,6,'
---------- ---------- ------------ ------------
      7369        800 ***800       800***
      7499       1600 **1600       1600**
      7521       1250 **1250       1250**
      7566       2975 **2975       2975**
        ...
```

■ TRIM

주어진 문자열의 좌측 / 우측에서 특정 문자를 제거한다.

```
SQL> SELECT EMPNO, JOB,
  2   TRIM(LEADING 'S' FROM JOB) AS LEADING,
  3   TRIM(TRAILING 'N' FROM JOB) AS TRAILING
  4  FROM EMP
  5  WHERE EMPNO = 7844

     EMPNO JOB       LEADING   TRAILING
---------- --------- --------- ---------
      7844 SALESMAN  ALESMAN   SALESMA
```

■ REPLACE

주어진 문자열에서 특정 문자열을 주어진 문자열로 교체한다.

```
SQL> SELECT EMPNO, JOB, REPLACE(JOB, 'MAN', 'PERSON')
  2  FROM EMP
  3  WHERE EMPNO = 7844;

     EMPNO JOB       REPLACE(JOB,'MAN','PERSON')
---------- --------- ---------------------------------------------------
      7844 SALESMAN  SALESPERSON
```

숫자 함수

숫자 함수는 입력으로 숫자 타입의 컬럼 또는 값을 받아서 결과로 숫자 타입의 값을 리턴하는 함수이다. 숫자 함수의 종류는 다음과 같다.

표 4-3. 숫자 함수의 종류

함수	설명
ROUND(column|expression, n)	입력된 숫자를 반올림하여 소수점 n자리로 리턴, n이 음수인 경우 정수 자리에서 반올림
TRUNC(column|expression, n)	입력된 숫자를 내림하여 소수점 n자리로 리턴
MOD(m, n)	m을 n으로 나눈 나머지 리턴

■ ROUND

ROUND는 입력 값을 지정된 자리에서 반올림하는 함수이다.

```
SQL> SELECT ROUND(45.923, 2), ROUND(45.923, 0),
  2  ROUND(45.923, -1)
  3  FROM DUAL;

ROUND(45.923,2) ROUND(45.923,0) ROUND(45.923,-1)
--------------- --------------- ----------------
          45.92              46               50
```

위에서 보면 SELECT 구문 뒤에 기술된 컬럼들을 연산하기 위해서는 FROM 구문이 불필요한 것처럼 보이지만 Oracle에서는 SELECT 문장에서 FROM이 누락되면 에러를 발생시키기 때문에 DUAL 테이블이라는 하나의 행과 하나의 컬럼을 갖는 더미(Dummy) 테이블을 사용한다.

■ TRUNC

TRUNC는 입력 값을 지정된 자리에서 내림하는 함수이다.

```
SQL> SELECT TRUNC(45.923, 2), TRUNC(45.923),
  2  TRUNC(45.923, -2)
  3  FROM DUAL;

TRUNC(45.923,2) TRUNC(45.923) TRUNC(45.923,-2)
--------------- ------------- ----------------
          45.92            45                0
```

■ MOD

MOD는 입력 값을 지정된 숫자로 나눈 나머지를 구하는 함수이다.

```
SQL> SELECT MOD(100, 30) FROM DUAL;

MOD(100,30)
-----------
         10
```

날짜 함수

날짜 함수를 설명하기에 앞서 날짜 타입에 대하여 먼저 알아보자. Oracle 데이터베이스는 날짜 타입을 저장할 때, 내부적으로 세기, 년도, 월, 일, 시간, 분, 초를 모두 저장한다. 그러나 화면에 표시되는 날짜는 디폴트로 RR/MM/DD로 표시되므로 20세기 년도(1900년대)와 21세기 년도(2000년대)의 처리에 의문이 생길 수 있지만, Oracle 데이터베이스는 날짜 타입에 내부적으로 세기 정보를 저장하고 있고, 데이터를 저장하고자 할 때도 연도를 4자리로 입력할 수 있기 때문에 큰 문제가 되지 않는다.

현재, 데이터베이스 서버에 설정된 날짜를 살펴보면 다음과 같다.

```
SQL> SELECT SYSDATE FROM DUAL;

SYSDATE
--------
04/06/16
```

참고로 현재 세션에서 날짜의 표시형식을 변경하는 방법은 다음과 같다. 이 설정은 현재 세션에서만 유효하므로 SQL*Plus를 종료하고 다시 시작하면 원래 설정으로 돌아간다.

```
SQL> ALTER SESSION
  2  SET NLS_DATE_FORMAT = 'YYYY/MM/DD HH24:MI:SS';

세션이 변경되었습니다.

SQL> SELECT SYSDATE FROM DUAL;

SYSDATE
-------------------
2004/06/16 14:14:22
```

날짜 타입의 연산에는 주의할 필요가 있다. 다음은 날짜 타입의 연산에 대한 의미와 결과를 정리한 것이다.

표 4-4. 날짜 연산

연산	결과	설명
날짜 + 숫자	날짜	날짜에 일수를 더한다
날짜 - 숫자	날짜	날짜에서 일수를 뺀다
날짜 - 날짜	숫자(일수)	두 날짜의 차이(일수)를 계산한다
날짜 + 숫자/24	날짜	날짜에 시간을 더한다

사원 테이블에서 사원들이 입사후 몇주가 경과되었는지를 살펴보면 다음과 같다.

```
SQL> SELECT EMPNO, ENAME, (SYSDATE-HIREDATE)/7 "주"
  2  FROM EMP;

     EMPNO ENAME                주
---------- ---------- ----------
      7369 SMITH      1226.08576
      7499 ALLEN      1216.80004
      7521 WARD       1216.51433
        ...
      7900 JAMES       1175.9429
      7902 FORD        1175.9429
      7934 MILLER     1168.65719

14 개의 행이 선택되었습니다.
```

날짜 함수의 종류는 다음과 같다.

표 4-5. 날짜 함수

함수	설명
MONTHS_BETWEEN(*date1*, *date2*)	*date1*과 *date2*의 차이를 월 단위로 계산한다
ADD_MONTHS(*date*, *n*)	*date*에 *n*개월을 더한다
NEXT_DAY(*date*, '*char*')	*date*이후 '*char*'로 지정된 요일의 날짜를 계산한다
LAST_DAY(*date*)	*date*의 해당 월에서 마지막 일을 계산한다
ROUND(*date*[, '*fmt*'])	*date*에서 '*fmt*'로 지정된 자리수로 반올림한다
TRUNC(*date*[, '*fmt*'])	*date*에서 '*fmt*'로 지정된 자리수로 내림한다

■ MONTHS_BETWEEN

두 날짜의 차이를 월 단위로 계산한다.

```
SQL> SELECT EMPNO, ENAME, MONTHS_BETWEEN(SYSDATE, HIREDATE)
  2  FROM EMP
  3  WHERE EMPNO = 7839;

     EMPNO ENAME      MONTHS_BETWEEN(SYSDATE,HIREDATE)
---------- ---------- --------------------------------
      7839 KING                            270.987419
```

■ ADD_MONTHS

날짜에 개월을 더한다.

```
SQL> SELECT EMPNO, ENAME, HIREDATE, ADD_MONTHS(HIREDATE, 6)
  2  FROM EMP
  3  WHERE EMPNO = 7839;

     EMPNO ENAME      HIREDATE ADD_MONT
---------- ---------- -------- --------
      7839 KING       81/11/17 82/05/17
```

■ NEXT_DAY

입력된 날짜로부터 지정된 다음 요일의 날짜를 계산한다.

```
SQL> SELECT EMPNO, ENAME, HIREDATE, NEXT_DAY(HIREDATE, '수')
  2  FROM EMP
  3  WHERE EMPNO = 7839;

     EMPNO ENAME      HIREDATE NEXT_DAY
---------- ---------- -------- --------
      7839 KING       81/11/17 81/11/18
```

■ LAST_DAY

입력된 날짜의 월에서 마지막 일을 계산한다.

```
SQL> SELECT EMPNO, ENAME, HIREDATE, LAST_DAY(HIREDATE)
  2  FROM EMP
  3  WHERE EMPNO = 7839;

     EMPNO ENAME      HIREDATE LAST_DAY
---------- ---------- -------- --------
      7839 KING       81/11/17 81/11/30
```

■ ROUND

입력된 날짜를 반올림한다.

```
SQL> SELECT EMPNO, ENAME, HIREDATE,
  2  ROUND(HIREDATE, 'MONTH'),
  3  ROUND(HIREDATE, 'YEAR')
  4  FROM EMP
  5  WHERE EMPNO = 7839;

     EMPNO ENAME      HIREDATE ROUND(HI ROUND(HI
---------- ---------- -------- -------- --------
      7839 KING       81/11/17 81/12/01 82/01/01
```

■ TRUNC

입력된 날짜를 내림한다.

```
SQL> SELECT EMPNO, ENAME, HIREDATE,
  2  TRUNC(HIREDATE, 'MONTH'),
  3  TRUNC(HIREDATE, 'YEAR')
  4  FROM EMP
  5  WHERE EMPNO = 7839;

     EMPNO ENAME      HIREDATE TRUNC(HI TRUNC(HI
---------- ---------- -------- -------- --------
      7839 KING       81/11/17 81/11/01 81/01/01
```

변환 함수

변환 함수는 데이터 타입을 다른 데이터 타입으로 변환하는 함수이다. 데이터 타입의 변환은 Oracle 데이터베이스 내부에서 자동으로 진행되는 암시적인 데이터 타입 변환과 변환 함수를 사용하는 명시적 데이터 타입 변환으로 구분되며, 다시 암시적 데이터 타입 변환은 값을 저장할 때와 수식을 전개할 때의 두 가지 경우에 따라 변환 방식이 다르다.

먼저, 테이블 또는 변수에 값을 입력하거나 지정하는 경우 변환 방식은 다음과 같다. 예를 들어, 테이블내 문자 타입 컬럼에 숫자를 저장하면 숫자는 내부적으로 문자로 변환되어 저장된다.

표 4-6. 암시적 데이터 타입 변환 1

변환 전	**변환 후**
VARCHAR2 또는 CHAR	NUMBER
VARCHAR2 또는 CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

다음은 수식을 전개하거나 조건식에서 값을 비교할 때 변환 방식이다. 예를 들어, 테이블내 문자 타입 컬럼을 숫자와 비교하면 컬럼내 저장된 값은 숫자로 변환되어 비교된다.

표 4-7. 암시적 데이터 타입 변환 2

변환 전	**변환 후**
VARCHAR2 또는 CHAR	NUMBER
VARCHAR2 또는 CHAR	DATE

특히, 두 번째 암시적 데이터 타입 변환이 SELECT 문장의 WHERE 구문에서 발생하는 경우를 INTERNAL SUPPRESSING이라고 부르며 검색 효율을 저하시키는 원인이 될 수가 있다. 예를 들어, SELECT 문장에 WHERE ID = 4567 이라고 기술되어 있고, ID 컬럼에 인덱스가 생성되어 있다고 가정하자. 여기서 ID 컬럼의 데이터 타입이 VARCHAR2 타입이라면 조건문의 4567이 VARCHAR2 타입으로 변환되는 것이 아니라 ID 컬럼의 데이터 값이 NUMBER 타입으로 변환되기 때문에 ID 컬럼에 생성되어 있는 인덱스를 사용할 수 없으므

로 검색 효율은 나빠지게 된다. 인덱스는 책의 찾아보기와 같은 것으로 데이터 검색의 효율을 향상시켜 주는 역할을 하는 것인데, 추후에 다루기로 한다.

명시적 데이터 타입 변환은 아래와 같은 전용 변환 함수를 사용하여 데이터 타입을 변경하는 방법이다.

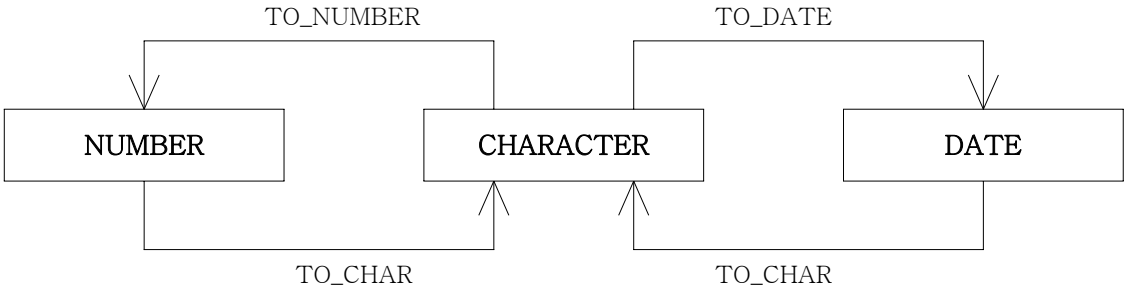

그림 4-1. 데이터 타입간 변환

■ TO_CHAR

TO_CHAR 함수는 NUMBER 또는 DATE 타입을 CHAR 타입으로 변환시켜 준다. 먼저, DATE 타입을 CHAR 타입으로 변환하기 위한 TO_CHAR 함수의 사용 방법은 다음과 같다.

```
TO_CHAR(date, 'format_model')
```

여기서 *format_model*은 아래 표와 같이 날짜, 시간, 기타 형식이 있으며 유효한 형식을 사용하여야 한다. 사용된 형식 앞에 *fm*을 붙이면 결과 값의 좌측부분에 있는 공백이나 0을 제거하여 표시할 수 있다.

표 4-8. *format_model* - 날짜 형식

형식	설명
YYYY	년도를 숫자로 표시
YEAR	년도를 영문으로 표시
MM	월을 숫자로 표시
MONTH	월을 영문으로 표시
MON	월을 영문 3자리로 축약해서 표시
DY	요일을 영문 3자리로 축약해서 표시, 한글 환경에서는 한글 1자로 표시
DAY	요일을 영문으로 표시, 한글 환경에서는 한글 3자로 표시
DD	일을 숫자로 표시

표 4-9. *format_model* - 시간 형식

형식	설명
AM 또는 PM	오전 또는 오후를 표시
A.M. 또는 P.M.	오전 또는 오후를 표시
HH 또는 HH12 또는 HH24	시간 또는 1~12시 또는 1~24시
MI	분(0~59)
SS	초(0~59)
SSSSS	자정 이후 초(0~86399)

표 4-10. *format_model* - 기타 형식

형식	설명
/ . ,	구분자 표시
"of the"	결과에 추가할 문자열
TH	서수 표시 (예, DDTH인 경우 4TH)
SP	영문 기수 표시 (예, DDSP인 경우 FOUR)
SPTH 또는 THSP	영문 서수 표시 (예, DDSPTH인 경우 FOURTH)

사용 예는 다음과 같다.

```
SQL> SELECT EMPNO, ENAME,
  2  TO_CHAR(HIREDATE, 'fmDD MONTH YYYY') "입사일"
  3  FROM EMP;

     EMPNO ENAME      입사일
---------- ---------- -----------------
      7369 SMITH      17 DECEMBER 1980
      7499 ALLEN      20 FEBRUARY 1981
      7521 WARD       22 FEBRUARY 1981
      7566 JONES      2 APRIL 1981
        ...
      7902 FORD       3 DECEMBER 1981
      7934 MILLER     23 JANUARY 1982
```

NUMBER 타입을 CHAR 타입으로 변환하기 위한 TO_CHAR 함수의 사용 방법은 다음과 같다.

```
TO_CHAR(number, 'format_model')
```

마찬가지로 적용가능한 *format_model*이 있으며 다음과 같다.

표 4-11. *format_model* - 숫자 형식

형식	설명
9	숫자로 표시
0	숫자의 앞부분을 0으로 표시
$	달러 표시
L	지역 화폐 단위 표시
.	소수점 표시
,	1000 단위 구분자 표시

사용 예는 다음과 같다.

```
SQL> SELECT EMPNO, ENAME, TO_CHAR(SAL, 'L99,999.00') "급여"
  2  FROM EMP;

     EMPNO ENAME      급여
---------- ---------- --------------------
      7369 SMITH                ￦800.00
      7499 ALLEN              ￦1,600.00
      7521 WARD               ￦1,250.00
        ...
      7934 MILLER             ￦1,300.00
```

■ TO_NUMBER

TO_NUMBER 함수는 CHAR 타입을 NUMBER 타입으로 변환시켜 준다. TO_NUMBER 함수의 사용 방법은 다음과 같다.

```
TO_NUMBER(char[, 'format_model'])
```

사용 예는 다음과 같다.

```
SQL> SELECT TO_NUMBER('100')+10 FROM DUAL;

TO_NUMBER('100')+10
-------------------
                110
```

■ TO_DATE

TO_DATE 함수는 CHAR 타입을 DATE 타입으로 변환시켜 준다. TO_DATE 함수의 사용 방법은 다음과 같다.

```
TO_DATE(char[, 'format_model'])
```

사용 예는 다음과 같다.

```
SQL> SELECT EMPNO, ENAME, HIREDATE
  2  FROM EMP
  3  WHERE HIREDATE=TO_DATE('DECEMBER 17, 1980', 'MONTH DD, YYYY');

     EMPNO ENAME      HIREDATE
---------- ---------- --------
      7369 SMITH      80/12/17
```

특히, TO_DATE 함수의 *format_model*에 fx 식별자를 사용하면 문자열과 *format_model*의 형식이 정확히 일치해야 되며 그렇지 않은 경우 에러가 발생된다.

```
SQL> SELECT EMPNO, ENAME, HIREDATE
  2  FROM EMP
  3  WHERE HIREDATE=TO_DATE('DECEMBER 17,  1980', 'fxMONTH DD, YYYY');
WHERE HIREDATE=TO_DATE('DECEMBER 17,  1980', 'fxMONTH DD, YYYY')
                       *
3행에 오류:
ORA-01841: 년은 영이 아닌 -4713 과 +4713 사이의 값으로 지정해야 합니다
```

또한, *format_model*에 추가적으로 RR 날짜 포맷이 있으며, 이 기능은 연도를 두자리로 입력하는 경우 적절히 1900년대 또는 2000년대의 4자리 연도로 적절히 변환되도록 해 준다. 변환 규칙은 다음과 같다.

표 4-12. RR 날짜 포맷에 의한 연도 변환 규칙

		지정하고자 하는 두자리 연도	
		0~49	50~99
시스템의 두자리 연도	0~49	현재 세기의 연도로 변환	전 세기의 연도로 변환
	50~99	다음 세기의 연도로 변환	현재 세기의 연도로 변환

예를 들면 다음과 같다.

```
SQL> ALTER SESSION
  2  SET NLS_DATE_FORMAT = 'YYYY/MM/DD';

세션이 변경되었습니다.

SQL> SELECT SYSDATE FROM DUAL;

SYSDATE
----------
2004/06/17

SQL> SELECT TO_DATE('99', 'YY'), TO_DATE('01', 'YY') FROM DUAL;

TO_DATE('9 TO_DATE('0
---------- ----------
2099/06/01 2001/06/01

SQL> SELECT TO_DATE('99', 'RR'), TO_DATE('01', 'RR') FROM DUAL;

TO_DATE('9 TO_DATE('0
---------- ----------
1999/06/01 2001/06/01
```

일반 함수

일반 함수 중에 NULL 값 처리와 관련된 함수는 다음과 같다.

표 4-13. 일반 함수

함수	**설명**
NVL(*expr1*, *expr2*)	*expr1*이 NULL이면 *expr2*를 리턴
NVL2(*expr1*, *expr2*, *expr3*)	*expr1*이 NULL이 아니면 *expr2*, NULL이면 *expr3*를 리턴
NULLIF(*expr1*, *expr2*)	*expr1*가 *expr2*가 같으면 NULL, 다르면 *expr1*을 리턴
COALESCE(*expr1*, *expr2*, ... , *exprn*)	인자들 중에서 NULL이 아닌 첫 번째 인자를 리턴

■ NVL

함수의 첫 번째 인자가 NULL 이면 두 번째 인자를 리턴한다. NVL 함수는 NULL 값이

포함된 컬럼을 연산에 포함시키고자 할 때, 아주 유용한 함수이다.

```
SQL> SELECT EMPNO, ENAME, SAL, COMM, SAL*12+COMM, SAL*12+NVL(COMM, 0)
  2  FROM EMP;

     EMPNO ENAME             SAL       COMM SAL*12+COMM SAL*12+NVL(COMM,0)
---------- ---------- ---------- ---------- ----------- -------------------
      7369 SMITH             800                                      9600
      7499 ALLEN            1600        300       19500               19500
      7521 WARD             1250        500       15500               15500
      7566 JONES            2975                                     35700
        ...
      7934 MILLER           1300                                     15600
```

■ NVL2

함수의 첫 번째 인자가 NULL이 아니면 두 번째 인자로, NULL이면 세 번째 인자를 리턴한다.

```
SQL> SELECT EMPNO, ENAME, SAL, COMM, SAL+COMM, NVL2(COMM, SAL+COMM, SAL)
  2  FROM EMP;

     EMPNO ENAME             SAL       COMM   SAL+COMM NVL2(COMM,SAL+COMM,SAL)
---------- ---------- ---------- ---------- ---------- ----------------------
      7369 SMITH             800                                          800
      7499 ALLEN            1600        300       1900                  1900
      7521 WARD             1250        500       1750                  1750
      7566 JONES            2975                                       2975
        ...
      7934 MILLER           1300                                       1300
```

■ NULLIF

함수의 첫 번째 인자와 두 번째 인자가 같으면 NULL을, 다르면 첫 번째 인자를 리턴한다.

```
SQL> SELECT EMPNO, ENAME, JOB, NULLIF(LENGTH(ENAME), LENGTH(JOB))
  2  FROM EMP;

     EMPNO ENAME      JOB       NULLIF(LENGTH(ENAME),LENGTH(JOB))
---------- ---------- --------- ---------------------------------
      7369 SMITH      CLERK
      7499 ALLEN      SALESMAN                                   5
      7521 WARD       SALESMAN                                   4
        ...
      7900 JAMES      CLERK
      7902 FORD       ANALYST                                    4
      7934 MILLER     CLERK                                      6
```

■ COALESCE

함수의 인자들 중에 NULL이 아닌 최초의 인자를 리턴한다.

```
SQL> SELECT EMPNO, ENAME, SAL, COMM, COALESCE(SAL, COMM, 10)
  2  FROM EMP;

     EMPNO ENAME             SAL       COMM COALESCE(SAL,COMM,10)
---------- ---------- ---------- ---------- --------------------
      7369 SMITH             800                            800
      7499 ALLEN            1600        300                1600
      7521 WARD             1250        500                1250
        ...
      7934 MILLER           1300                           1300
```

■ CASE

CASE 함수는 SQL 문장내에서 IF-THEN-ELSE와 같은 흐름제어문의 역할을 한다. CASE 함수의 문법은 다음과 같다.

```
CASE expr WHEN comparison_expr1 THEN return_expr1
         [WHEN comparison_expr2 THEN return_expr2
          WHEN comparison_exprn THEN return_exprn
          ELSE else_expr]
END
```

*expr*이 *comparison_expr1*과 같으면 *return_expr1*, *comparison_expr2*와 같으면 *return_expr2*, *comparison_exprn*과 같으면 *return_exprn*을 리턴하고 그렇지 않으면 *else_expr*을 리턴한다. 사용방법은 다음과 같다.

```
SQL> SELECT EMPNO, ENAME, SAL, JOB,
  2   CASE JOB WHEN 'ANALYST'    THEN SAL*1.1
  3            WHEN 'CLERK'      THEN SAL*1.2
  4            WHEN 'MANAGER'    THEN SAL*1.3
  5            WHEN 'PRESIDENT'  THEN SAL*1.4
  6            WHEN 'SALESMAN'   THEN SAL*1.5
  7            ELSE SAL
  8   END "급여"
  9  FROM EMP;

     EMPNO ENAME             SAL JOB              급여
---------- ---------- ---------- --------- ----------
      7369 SMITH             800 CLERK            960
      7499 ALLEN            1600 SALESMAN        2400
      7521 WARD             1250 SALESMAN        1875
      7566 JONES            2975 MANAGER       3867.5
      7654 MARTIN           1250 SALESMAN        1875
        ...
      7900 JAMES             950 CLERK           1140
      7902 FORD             3000 ANALYST         3300
      7934 MILLER           1300 CLERK           1560
```

■ DECODE

DECODE 함수도 CASE 함수와 마찬가지로 SQL 문장내에서 IF-THEN-ELSE와 같은

흐름제어문의 역할을 한다. DECODE 함수의 문법은 다음과 같다.

```
DECODE(col|expression, search1, result1
                    [, search2, result2, ... ,]
                    [, default])
```

*col1|expression*이 *search1*과 같으면 *result1*, *search2*와 같으면 *result2*를 리턴하고 그렇지 않으면 *default*를 리턴한다. 사용방법은 다음과 같다.

```
SQL> SELECT EMPNO, ENAME, SAL, JOB,
  2   DECODE(JOB, 'ANALYST'   , SAL*1.1,
  3               'CLERK'     , SAL*1.2,
  4               'MANAGER'   , SAL*1.3,
  5               'PRESIDENT', SAL*1.4,
  6               'SALESMAN' , SAL*1.5, SAL) "급여"
  7  FROM EMP;

     EMPNO ENAME             SAL JOB              급여
---------- ---------- ---------- --------- ----------
      7369 SMITH             800 CLERK            960
      7499 ALLEN            1600 SALESMAN        2400
      7521 WARD             1250 SALESMAN        1875
      7566 JONES            2975 MANAGER       3867.5
      7654 MARTIN           1250 SALESMAN        1875
        ...
      7900 JAMES             950 CLERK           1140
      7902 FORD             3000 ANALYST         3300
      7934 MILLER           1300 CLERK           1560

14 개의 행이 선택되었습니다.
```

위에서 기술한 단일행 함수들은 함수의 중첩이 가능하며, 함수 중첩이 되어 있는 경우 연산의 순서는 가장 안쪽에 있는 함수부터 바깥 쪽 함수로 연산된다.

```
F3(F2(F1(col1, arg1), arg2), arg3)
```

```
SQL> SELECT EMPNO, ENAME, MGR,
  2  NVL(TO_CHAR(MGR), '상위 관리자 없음')
  3  FROM EMP
  4  WHERE MGR IS NULL;

     EMPNO ENAME             MGR NVL(TO_CHAR(MGR),'상위관리자없음')
---------- ---------- ---------- ----------------------------------------
      7839 KING                  상위 관리자 없음
```

복습

1. 사원 테이블의 사원명에서 2번째 문자부터 3개의 문자를 추출하시오.

2. 사원 테이블에서 입사일이 12월인 사원의 사번, 사원명, 입사일을 검색하시오.

3. 다음과 같은 결과를 검색할 수 있는 SQL 문장을 작성하시오.

```
     EMPNO ENAME      급여
---------- ---------- --------------------
      7369 SMITH      *******800
      7499 ALLEN      ******1600
      7521 WARD       ******1250
        ...
      7934 MILLER     ******1300

14 개의 행이 선택되었습니다.
```

4. 다음과 같은 결과를 검색할 수 있는 SQL 문장을 작성하시오.

```
     EMPNO ENAME      입사일
---------- ---------- ----------
      7369 SMITH      1980-12-17
      7499 ALLEN      1981-02-20
      7521 WARD       1981-02-22
        ...
      7934 MILLER     1982-01-23

14 개의 행이 선택되었습니다.
```

5. 사원 테이블에서 급여에 따라 사번, 이름, 급여, 등급을 검색하는 SQL 문장을 작성하시오. (Hint : CASE 함수 사용)

급여	등급
0~1000	E
1001~2000	D
2001~3000	C
3001~4000	B
4001~5000	A

```
     EMPNO ENAME             SAL
---------- ---------- ---------- -
      7369 SMITH             800 E
      7499 ALLEN            1600 D
      7521 WARD             1250 D
      7566 JONES            2975 C
        ...
      7934 MILLER           1300 D

14 개의 행이 선택되었습니다.
```