숫자 함수와 날짜 함수

숫자 함수와 날짜 함수

1. 숫자 함수 (Number Functions)

1.1 개요

- 숫자 함수는 숫자 입력을 사용하고 숫자 값을 반환

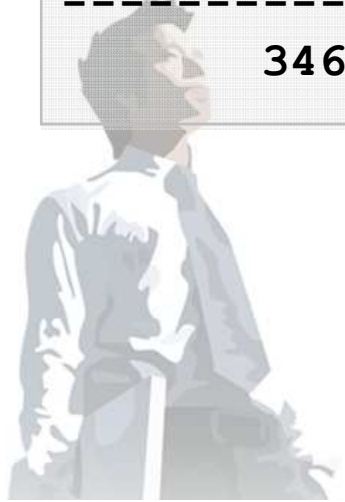
함 수	설 명
ROUND	지정한 소수점 자리로 값을 반올림
TRUNC	지정한 소수점 자리까지 남기고 값을 버림
MOD	나눗셈의 나머지를 반환

숫자 함수와 날짜 함수

1.2 ROUND

```
SQL> SELECT   ROUND(345.678) AS round1,
  2           ROUND(345.678, 0) AS round2,
  3           ROUND(345.678, 1) AS round3,
  4           ROUND(345.678, -1) AS round4
  5  FROM     dual;
```

```
    ROUND1      ROUND2      ROUND3      ROUND4
---------- ---------- ---------- ----------
       346         346       345.7         350
```

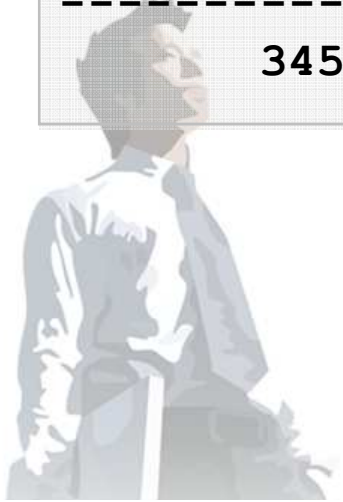


숫자 함수와 날짜 함수

1.3 TRUNC

```
SQL> SELECT   TRUNC(345.678) AS trunc1,
  2           TRUNC(345.678, 0) AS trunc2,
  3           TRUNC(345.678, 1) AS trunc3,
  4           TRUNC(345.678, -1) AS trunc4
  5  FROM     dual;
```

```
    TRUNC1      TRUNC2      TRUNC3      TRUNC4
---------- ---------- ---------- ----------
       345         345       345.6         340
```

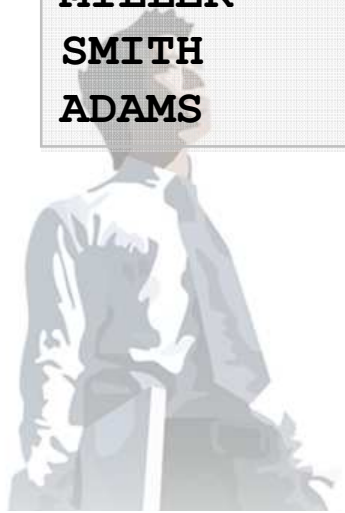


숫자 함수와 날짜 함수

1.4 MOD

```
SQL> SELECT ename, sal, MOD(sal, 1000)
  2  FROM   emp
  3  WHERE  job = 'CLERK';
```

```
ENAME                          SAL MOD(SAL,1000)
-------------------- ---------- -------------
JAMES                         950           950
MILLER                       1300           300
SMITH                         800           800
ADAMS                        1100           100
```



숫자 함수와 날짜 함수

2. 날짜 함수 (Date Functions)

2.1 오라클의 날짜

- 오라클은 세기, 연도, 월, 시, 분, 초와 같은 내부 숫자 형식으로 날짜를 저장
- 기본 날짜 형식은 **DD-MON-YY**
- 오라클의 날짜 범위는 **B.C.4712년1월1일 ~ A.D.9999년12월31일**
- **SYSDATE**는 현재의 날짜와 시간을 반환하는 함수
- **SYSDATE**는 **DUAL**이라는 **DUMMY** 테이블에서 선택

```
SQL> SELECT SYSDATE
  2  FROM   dual;
```

숫자 함수와 날짜 함수

2.2 날짜 연산

연 산	결 과	설 명
날짜 + 숫자	날짜	날짜에 일수를 더함
날짜 - 숫자	날짜	날짜에서 일수를 뺌
날짜 - 날짜	숫자(일수)	날짜에서 다른 날짜를 뺌
날짜 + 숫자/24	날짜	날짜에 시간 수를 더함

```
SQL> SELECT ename, (SYSDATE-hiredate)/7 AS WEEKS
  2  FROM   emp
  3  WHERE  deptno = 20;
```

숫자 함수와 날짜 함수

2.3 날짜 함수

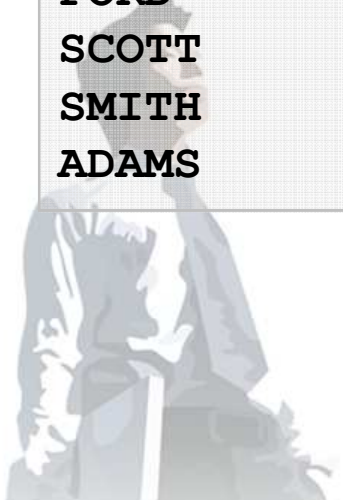
함 수	결 과
MONTHS_BETWEEN	두 날짜 사이의 월 수를 반환
ADD_MONTHS	날짜에 월을 더함
NEXT_DAY	날짜의 다음에 돌아오는 명시한 요일의 날짜를 반환
LAST_DAY	해당 월의 마지막 날짜를 반환
ROUND	날짜 반올림
TRUNC	날짜 버림

숫자 함수와 날짜 함수

2.3 날짜 함수 (계속)

```
SQL> SELECT ename, MONTHS_BETWEEN(SYSDATE, hiredate)
  2  FROM   emp
  3  WHERE  deptno = 20;
```

```
ENAME                MONTHS_BETWEEN(SYSDATE,HIREDATE)
-------------------- --------------------------------
JONES                                       347.712404
FORD                                        339.680146
SCOTT                                       327.486598
SMITH                                       351.228533
ADAMS                                       326.389823
```

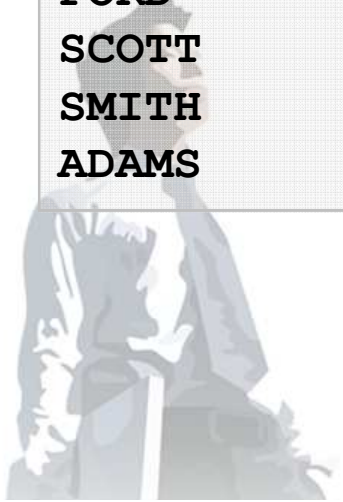


숫자 함수와 날짜 함수

2.3 날짜 함수 (계속)

```
SQL> SELECT ename, hiredate, ADD_MONTHS(hiredate, 6)
  2  FROM   emp
  3  WHERE  deptno = 20;
```

```
ENAME                ADD_MONT
-------------------- --------
JONES                81/10/02
FORD                 82/06/03
SCOTT                83/06/09
SMITH                81/06/17
ADAMS                83/07/12
```



숫자 함수와 날짜 함수

2.3 날짜 함수 (계속)

```
SQL> SELECT ename, NEXT_DAY(hiredate, '금'),
  2         LAST_DAY(hiredate)
  3  FROM   emp
  4  WHERE  deptno = 20;
```

```
ENAME                NEXT_DAY LAST_DAY
-------------------- -------- --------
JONES                81/04/03 81/04/30
FORD                 81/12/04 81/12/31
SCOTT                82/12/10 82/12/31
SMITH                80/12/19 80/12/31
ADAMS                83/01/14 83/01/31
```

숫자 함수와 날짜 함수

2.3 날짜 함수 (계속)

```
SQL> ALTER SESSION SET
  2  NLS_DATE_FORMAT = 'RRRR-MM-DD HH24:MI';
```

```
SQL> SELECT SYSDATE,
  2         ROUND(SYSDATE) ROUND1,
  3         ROUND(SYSDATE, 'DD') ROUND2,
  4         ROUND(SYSDATE, 'DAY') ROUND3,
  5         ROUND(SYSDATE, 'MON') ROUND4,
  6         ROUND(SYSDATE, 'YEAR') ROUND5
  7  FROM   dual;
```

숫자 함수와 날짜 함수

2.3 날짜 함수 (계속)

```
SQL> SELECT SYSDATE,
  2         TRUNC(SYSDATE) TRUNC1,
  3         TRUNC(SYSDATE, 'MON') TRUNC2,
  4         TRUNC(SYSDATE, 'YEAR') TRUNC3
  5  FROM   dual
```

```
SYSDATE          TRUNC1           TRUNC2           TRUNC3
---------------- ---------------- ---------------- ----------------
2010-03-10 16:44 2010-03-10 00:00 2010-03-01 00:00 2010-01-01 00:00
```

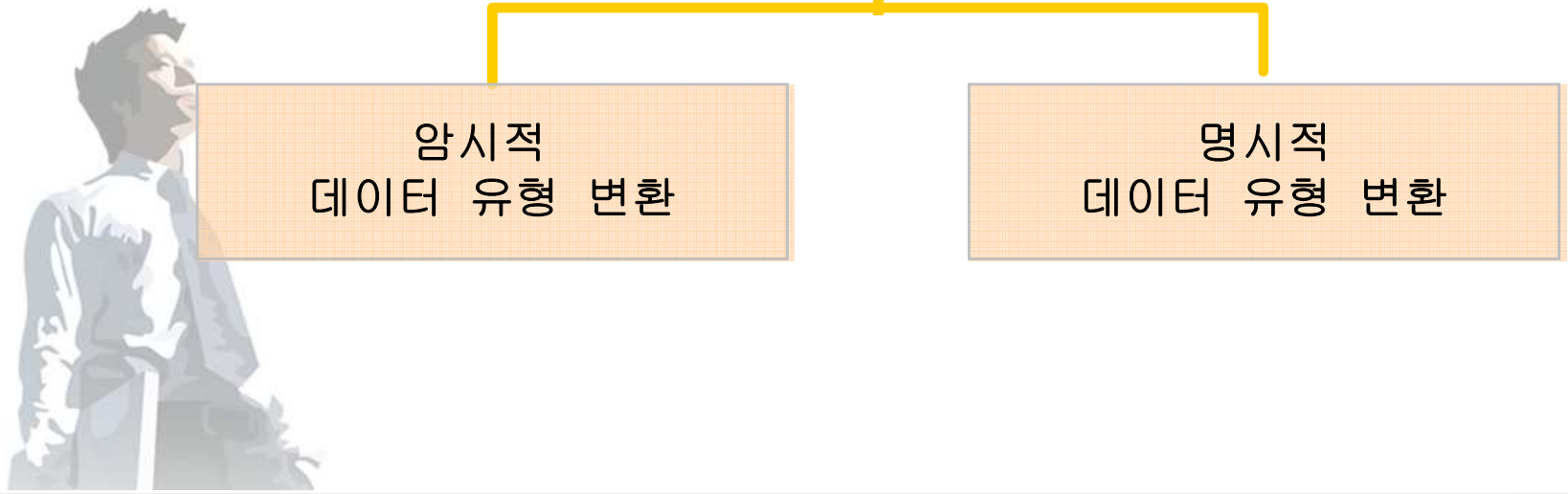
변환 함수 I

변환 함수 I

1. 데이터 유형 변환

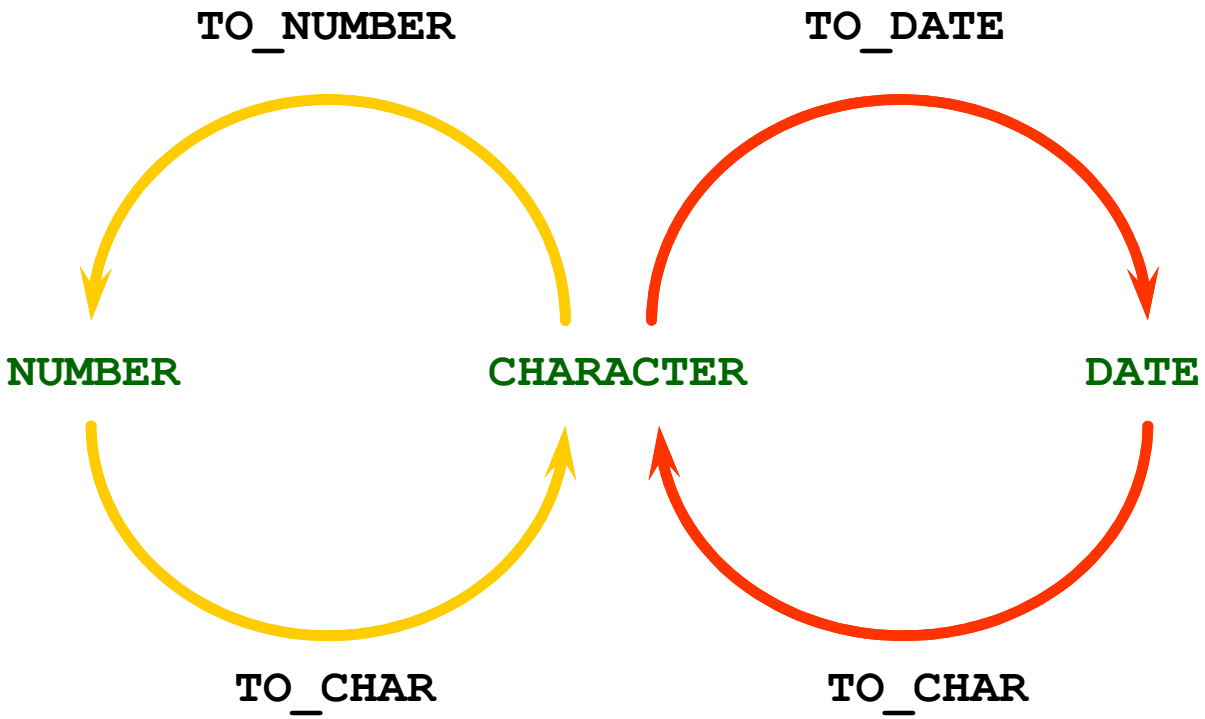
1.1 개요

- 암시적 데이터 유형 변환을 사용할 수 있지만 SQL문의 신뢰성을 높이기 위해서는 명시적 데이터 유형 변환을 사용



변환 함수 I

1.2 명시적 데이터 유형 변환



변환 함수 I

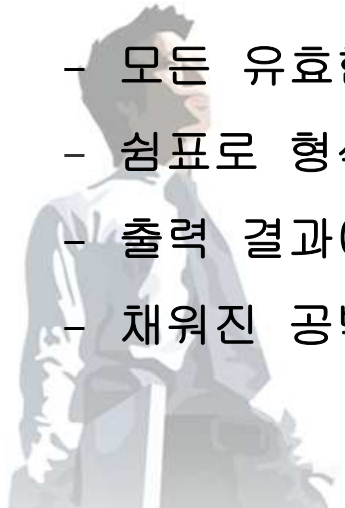
2. TO_CHAR

2.1 날짜에 TO_CHAR 함수 사용

- **TO_CHAR** 함수를 사용하면 기본 형식의 날짜를 사용자가 지정하는 문자 데이터 유형으로 변환 가능

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

- 형식 모델 **(Format Model)**
  - 작은 따옴표('')로 묶어야 하며 대소문자 구분
  - 모든 유효한 날짜 형식 요소를 포함
  - 쉼표로 형식모델과 날짜 값을 구분
  - 출력 결과에서 날짜 및 달 이름에는 공백이 자동으로 채워짐
  - 채워진 공백을 제거하거나 선행 0를 제거하는 **fm** 요소가 존재



변환 함수 I

2.1 날짜에 TO_CHAR 함수 사용 (계속)

```
SQL> SELECT empno, ename,
  2         TO_CHAR(hiredate, 'YYYY-MM') AS 입사년월
  3  FROM   emp
  4  WHERE  ename = 'SCOTT';
```

```
     EMPNO ENAME                          입사년월
---------- ------------------------------ --------------
      7788 SCOTT                          1982-12
```

변환 함수 I

2.1 날짜에 TO_CHAR 함수 사용 (계속)

- 날짜 형식 요소

요소	설명
YYYY	네자리 연도 (숫자)
YEAR	연도 (문자)
MM	두자리 값으로 나타낸 월
MONTH	월 전체 이름
MON	세자 약어로 나타낸 월
DY	세자 약어로 나타낸 요일
DAY	요일 전체 이름
DD	숫자로 나타낸 월의 일자

변환 함수 I

2.1 날짜에 TO_CHAR 함수 사용 (계속)

- 시간 형식 요소

요소	설명
AM / PM	오전 / 오후 표시자
HH / HH24	하루 중의 시간 또는 0-23으로 표시되는 시간
MI	분 (0-59)
SS	초 (0-59

- 기타 형식 요소

요소	설명
/ . , -	사용 문자가 결과에 그대로 나타남
“문자열”	인용 문자열을 결과에 재사용

변환 함수 I

2.1 날짜에 TO_CHAR 함수 사용 (계속)

- 숫자 표시에 영향을 주는 접미사

요소	설명
TH	서수 (예:DDTH -> 4TH)
SP	문자로 표시한 수 (예:DDSP -> FOUR)
SPTH/THSP	문자로 명시한 서수 (예:DDSPTH -> FOURTH)

변환 함수 I

2.1 날짜에 TO_CHAR 함수 사용 (계속)

```
SQL> ALTER SESSION SET
  2  NLS_DATE_LANGUAGE = AMERICAN;
```

```
SQL> SELECT TO_CHAR
  2           (hiredate, 'yyyy "년" Ddspth Month hh:mi:ss pm')
  3  FROM    emp
  4  WHERE   deptno = 30;
```

```
TO_CHAR(HIREDATE,'YYYY"년"DDSPTHMONTHHH:MI:SSPM')
------------------------------------------------
1981 년 First May        12:00:00 am
1981 년 Twentieth February  12:00:00 am
1981 년 Twenty-Second February  12:00:00 am
1981 년 Twenty-Eighth September 12:00:00 am
1981 년 Eighth September 12:00:00 am
1981 년 Third December  12:00:00 am
```

변환 함수 I

2.1 날짜에 TO_CHAR 함수 사용 (계속)

```
SQL> SELECT TO_CHAR
  2         (hiredate, 'fm yyyy "년" Ddspth Month hh:mi:ss pm')
  3  FROM    emp
  4  WHERE   deptno = 30;
```

```
TO_CHAR(HIREDATE,'FMYYYY"년"DDSPTHMONTHHH:MI:SSPM')
--------------------------------------------------
 1981 년 First May 12:0:0 am
 1981 년 Twentieth February 12:0:0 am
 1981 년 Twenty-Second February 12:0:0 am
 1981 년 Twenty-Eighth September 12:0:0 am
 1981 년 Eighth September 12:0:0 am
 1981 년 Third December 12:0:0 am
```

변환 함수 I

2.2 숫자에 TO_CHAR 함수 사용

- TO_CHAR 함수를 사용하면 기본 형식의 숫자를 사용자가 지정하는 문자 데이터 유형으로 변환 가능

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

```
SQL> SELECT TO_CHAR(777.777, '$99999.9999')
  2  FROM   dual;
```

```
TO_CHAR(777.
------------
   $777.7770
```

변환 함수 I

2.2 숫자에 TO_CHAR 함수 사용 (계속)

- 숫자 형식 요소

요소	설명
9	숫자를 표시
0	0을 강제로 표시
$	$ 기호 표시
L	지역 화폐 기호 표시
.	명시한 위치에 소수점 표시
,	명시한 위치에 구분자 표시

변환 함수 I

2.2 숫자에 TO_CHAR 함수 사용 (계속)

```
SQL> SELECT TO_CHAR(1234.5678, '9999.999')
  2  FROM   dual;
```

```
TO_CHAR(1
---------
 1234.568
```

```
SQL> SELECT ename, TO_CHAR(sal, '$99999') salary
  2  FROM   emp
  3  WHERE  job = 'SALESMAN';
```

```
ENAME                SALARY
-------------------- --------------
ALLEN                  $1600
WARD                   $1250
MARTIN                 $1250
TURNER                 $1500
```

변환 함수 II

변환 함수 II

1. TO_NUMBER

1.1 개요

- 문자열을 숫자로 변환해야 하는 경우 **TO_NUMBER** 변환 함수 사용
- 형식 모델은 숫자 형식 요소 기반

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

```
SQL> SELECT TO_NUMBER('$3,400', '$99,999')
  2  FROM   dual;
```

```
TO_NUMBER('$3,400','$99,999')
-----------------------------
                         3400
```

변환 함수 II

1.2 실습 예제

```
SQL> SELECT TO_NUMBER('$1200', '$9999')
  2  FROM   dual;
```

```
TO_NUMBER('$1200','$9999')
--------------------------
                      1200
```

```
SQL> SELECT TO_NUMBER('1200')
  2  FROM   dual;
```

```
TO_NUMBER('1200')
-----------------
             1200
```

변환 함수 II

2. TO_DATE

2.1 개요

- 문자열을 날짜로 변환해야 하는 경우 **TO_DATE** 변환 함수 사용
- 형식 모델은 날짜 및 시간 형식 요소 기반

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

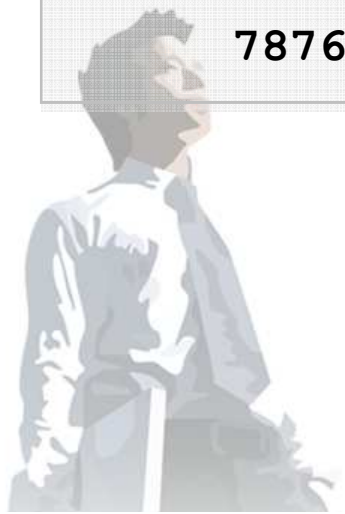
```
SQL> SELECT TO_DATE('2010년, 02월', 'YYYY"년", MM"월"')
  2  FROM   dual;

TO_DATE(
--------
10/02/01
```

변환 함수 II

2.2 실습 예제

```
SQL> SELECT empno, ename, sal, hiredate
  2  FROM   emp
  3  WHERE  hiredate =
                TO_DATE('January 12, 1983', 'Month DD, YYYY');
```

```
     EMPNO ENAME                          SAL HIREDATE
---------- -------------------- ---------- ----------------
      7876 ADAMS                      1100 1983-01-12 00:00
```

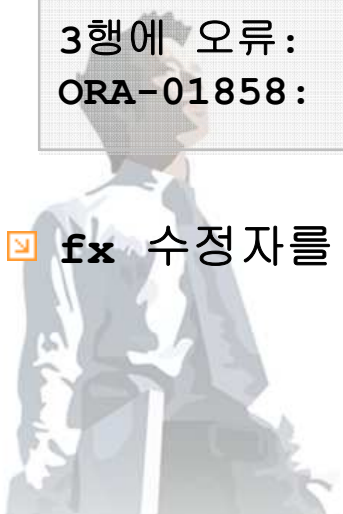


변환 함수 II

2.2 실습 예제 (계속)

```
SQL> SELECT empno, ename, sal, hiredate
  2  FROM   emp
  3  WHERE  hiredate =
            TO_DATE('January  12, 1983', 'fxMonth DD, YYYY');
```

```
WHERE  hiredate = TO_DATE('January  12, 1983', 'fxMonth DD, YYYY')
                          *
3행에 오류:
ORA-01858: 수치를 지정해야 할 위치에 비수치 문자가 지정되었습니다
```



- **fx** 수정자를 사용하여 완벽한 문장 형식 검증

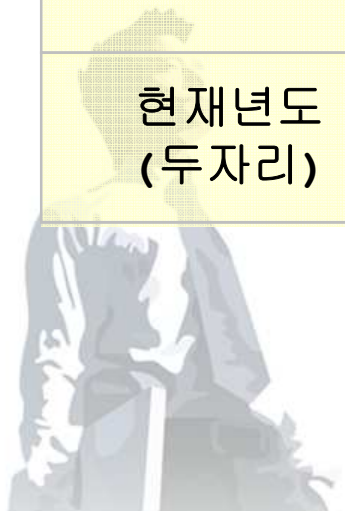
변환 함수 II

3. RR 날짜 형식

3.1 개요

- RR 날짜 형식은 YY 날짜 형식 요소와 유사하지만 여러 세기 지정 가능
- YY 대신 RR 날짜 형식 요소를 사용하면 지정 두자리 연도 및 현재 연도의 마지막 두자리에 따라 세기 값이 달라짐
- RR 날짜 형식

RR 날짜 형식		지정한 연도 (두자리)	
		0-49	50-99
현재년도 (두자리)	0-49	반환일이 현재 세기	반환일이 이전 세기
	50-99	반환일이 다음 세기	반환일이 현재 세기



변환 함수 II

3.2 실습 예제

현재 연도	지정한 날짜	RR 형식	YY 형식
1995	95/03/01	1995	1995
1995	10/03/01	2010	1910
2010	10/03/01	2010	2010
2010	95/03/01	1995	2095

변환 함수 II

3.2 실습 예제 (계속)

```
SQL> CREATE TABLE test (
  2     id          NUMBER(3),
  3     name        VARCHAR2(10),
  4     hiredate    DATE);
```

```
SQL> INSERT INTO test
  2  VALUES (1, 'SKJ', '95/03/01');
```

```
SQL> INSERT INTO test
  2  VALUES (2, 'HKD', '10/03/01');
```

변환 함수 II

3.2 실습 예제 (계속)

```
SQL> SELECT *
  2  FROM   test;
```

```
        ID NAME       HIREDATE
---------- ---------- --------
         1 SKJ        95/03/01
         2 HKD        10/03/01
```

변환 함수 II

3.2 실습 예제 (계속)

```
SQL> SELECT *
  2  FROM   test
  3  WHERE  hiredate = '95/03/01';
```

```
        ID NAME       HIREDATE
---------- ---------- --------
         1 SKJ        95/03/01
```

```
SQL> SELECT *
  2  FROM   test
  3  WHERE  hiredate = '1995/03/01';
```

```
SQL> SELECT *
  2  FROM   test
  3  WHERE  hiredate = '2095/03/01';
```

변환 함수 II

3.2 실습 예제 (계속)

```
SQL> SELECT *
  2  FROM   test
  3  WHERE  hiredate = '10/03/01';
```

```
        ID NAME       HIREDATE
---------- ---------- --------
         2 HKD        10/03/01
```

```
SQL> SELECT *
  2  FROM   test
  3  WHERE  hiredate = '2010/03/01';
```

```
SQL> SELECT *
  2  FROM   test
  3  WHERE  hiredate = '1910/03/01';
```

변환 함수 II

3.2 실습 예제 (계속)

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

```
SQL> INSERT INTO test
  2  VALUES (3, 'KKK', '95/03/01');
```

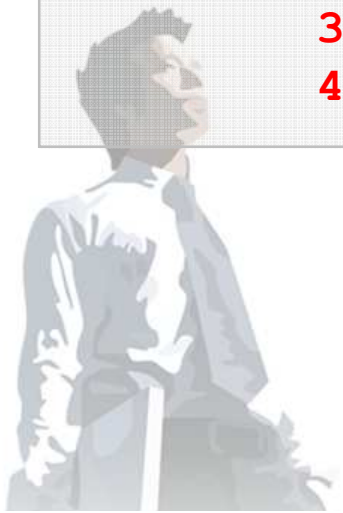
```
SQL> INSERT INTO test
  2  VALUES (4, 'SSS', '10/03/01');
```

변환 함수 II

3.2 실습 예제 (계속)

```
SQL> SELECT *
  2  FROM   test;
```

```
        ID NAME       HIREDATE
---------- ---------- --------
         1 SKJ        95/03/01
         2 HKD        10/03/01
         3 KKK        95/03/01
         4 SSS        10/03/01
```

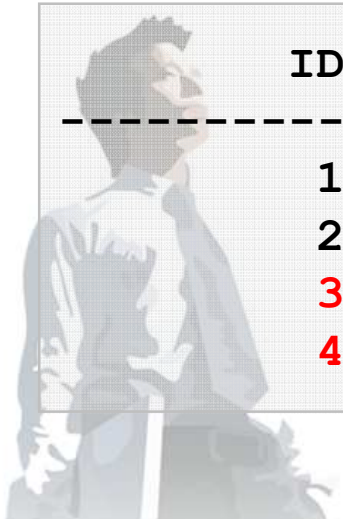


변환 함수 II

3.2 실습 예제 (계속)

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

```
SQL> SELECT *
  2  FROM   test;
```



```
        ID NAME       HIREDATE
---------- ---------- ----------
         1 SKJ        1995-03-01
         2 HKD        2010-03-01
         3 KKK        2095-03-01
         4 SSS        2010-03-01
```

일반 함수와 중첩 함수

일반 함수와 중첩 함수

1. NVL 함수

- 널(NULL)을 실제 값으로 변환
- 사용되는 데이터 유형은 날짜, 문자 및 숫자
- 데이터 유형은 서로 일치
- 구문

```
NVL (expr1, expr2)
```

- ***expr1*** : 널을 포함할 수 있는 소스 값 또는 표현식
- ***expr2*** : 널을 변환할 대상 값

일반 함수와 중첩 함수

1. NVL 함수 (계속)

```
SQL> SELECT ename, NVL(comm, '보너스 없음')
  2  FROM   emp;
```

```
SQL> SELECT ename, NVL(comm, 0)
  2  FROM   emp;
```

```
ENAME                NVL(COMM,0)
-------------------- -----------
KING                           0
JONES                          0
BLAKE                          0
CLARK                          0
FORD                           0
SCOTT                          0
ALLEN                        300
WARD                         500
MARTIN                      1400
...
```

일반 함수와 중첩 함수

2. NVL2 함수

- 첫번째 표현식(***expr1***)이 널이 아닌 경우, 두번째 표현식(***expr2***) 반환
- 첫번째 표현식(***expr1***)이 널인 경우, 세번째 표현식(***expr3***) 반환
- 구문

```
NVL2 (expr1, expr2, expr3)
```

- ***expr1*** : 널을 포함할 수 있는 소스 값 또는 표현식
- ***expr2*** : ***expr1***이 널이 아닌 경우 반환되는 값
- ***expr3*** : ***expr1***이 널이 경우 반환되는 값

일반 함수와 중첩 함수

2. NVL2 함수 (계속)

```
SQL> SELECT NVL2(TO_CHAR(comm),TO_CHAR(comm),'보너스 없음')
  2  FROM   emp;
```

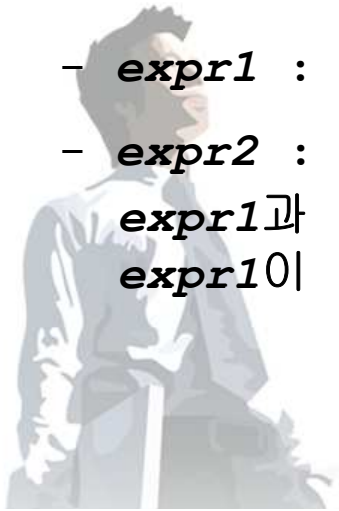
```
NVL2(TO_CHAR(COMM),TO_CHAR(COMM),'보너스없음')
----------------------------------------------
보너스 없음
보너스 없음
보너스 없음
보너스 없음
보너스 없음
보너스 없음
보너스 없음
300
500
1400
0
보너스 없음
보너스 없음
...
```

일반 함수와 중첩 함수

3. NULLIF 함수

- 두 표현식 비교
  - 동일한 경우 널(null)을 반환
  - 동일하지 않은 경우 첫번째 표현식(*expr1*)을 반환
- 첫번째 표현식(*expr1*)에 널을 지정할 수 없음
- 구문

```
NULLIF (expr1, expr2)
```



- *expr1* : *expr2*와 비교되는 소스 값
- *expr2* :
  *expr1*과 비교되는 소스 값이고, 이 값이 *expr1*과 동일하지 않은 경우 *expr1*이 반환

일반 함수와 중첩 함수

3. NULLIF 함수 (계속)

```
SQL> SELECT ename, LENGTH(ename) "expr1",
  2         job, LENGTH(job) "expr2",
  3         NULLIF(LENGTH(ename), LENGTH(job)) result
  4  FROM   emp;
```

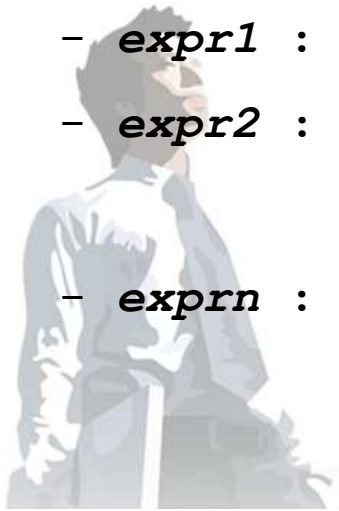
DNAME	expr1	LOC	expr2	RESULT
ACCOUNTING	10	NEW YORK	8	10
RESEARCH	8	DALLAS	6	8
SALES	5	CHICAGO	7	5
OPERATIONS	10	BOSTON	6	10

일반 함수와 중첩 함수

4. COALESCE 함수

- 여러 대체 값을 사용할 수 있다는 점에서 **NVL** 함수보다 좋음
- 첫번째 표현식(***expr1***)이 널이 아닌 경우, 해당 표현식을 반환
- 첫번째 표현식(***expr1***)이 널인 경우, 나머지 표현식(***exprn***)에 대해 **COALESCE** 함수 적용
- 구문

```
COALESCE (expr1, expr2, ..., exprn)
```



- ***expr1*** : 이 표현식이 널이 아닌 경우, 이 표현식 반환
- ***expr2*** : ***expr1***이 널이고 이 표현식이 널이 아닌 경우, 이 표현식 반환
- ***exprn*** : 앞의 표현식이 모두 널인 경우, 이 표현식 반환

일반 함수와 중첩 함수

4. COALESCE 함수 (계속)

```
SQL> SELECT COALESCE(NULL, NULL, 10, 100, NULL)
  2  FROM   dual;
```

```
COALESCE(NULL,NULL,10,100,NULL)
-------------------------------
                             10
```

일반 함수와 중첩 함수

5. 조건 표현식

5.1 개요

- **SQL**문 안에서 **IF-THEN-ELSE** 논리를 사용할 수 있도록 함
- 두 가지 방법
  - **CASE** 표현식 **(ANSI SQL** 준수**)**
  - **DECODE** 함수 **(**오라클 구문에만 존재**)**

일반 함수와 중첩 함수

5.2 CASE 표현식

- **IF-THEN-ELSE** 문의 역할을 수행하여 조건부 조회를 수행

```
CASE expr  WHEN comparison_expr1 THEN returen_expr1
          [WHEN comparison_expr2 THEN returen_expr2
           WHEN comparison_exprn THEN returen_exprn
           ELSE else_expr]
END
```

일반 함수와 중첩 함수

5.2 CASE 표현식 (계속)

```
SQL> SELECT ename, job, sal,
  2         CASE job WHEN 'MANAGER' THEN sal*1.1
  3                  WHEN 'ANALYST' THEN sal*1.2
  4                  WHEN 'CLERK'   THEN sal*1.3
  5                  ELSE sal
  6         END
  7         AS "인상된 급여"
  8  FROM   emp;
```

```
ENAME                JOB                       SAL   인상된 급여
-------------------- ------------------ ---------- -----------
KING                 PRESIDENT                5000        5000
JONES                MANAGER                  2975      3272.5
BLAKE                MANAGER                  2850        3135
CLARK                MANAGER                  2450        2695
FORD                 ANALYST                  3000        3600
...
```

일반 함수와 중첩 함수

5.2 CASE 표현식 (계속)

```
SQL> SELECT ename, job, sal,
  2         CASE WHEN sal < 1000 THEN sal*1.1
  3              WHEN sal < 2000 THEN sal*1.2
  4              WHEN sal < 3000 THEN sal*1.3
  5              ELSE sal
  6         END
  7         AS "인상된 급여"
  8  FROM   emp;
```

일반 함수와 중첩 함수

5.3 DECODE 함수

- **CASE** 또는 **IF-THEN-ELSE** 문의 역할을 수행하여 조건부 조회를 수행

```
DECODE(col | comparison, search1, result1
                        [, search2, result2
                         , searchn, resultn]
                        [, default ] )
```

일반 함수와 중첩 함수

5.3 DECODE 함수 (계속)

```
SQL> SELECT ename, job, sal,
 2          DECODE(job, 'MANAGER', sal*1.1,
 3                      'ANALYST', sal*1.2,
 4                      'CLERK',   sal*1.3,
 5                                 sal)
 6          AS "인상된 급여"
 7   FROM   emp;
```

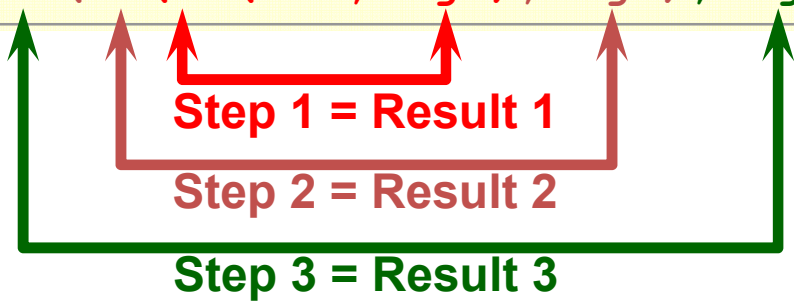
ENAME	JOB	SAL	인상된 급여
KING	PRESIDENT	5000	5000
JONES	MANAGER	2975	3272.5
BLAKE	MANAGER	2850	3135
CLARK	MANAGER	2450	2695
FORD	ANALYST	3000	3600
•••			

일반 함수와 중첩 함수

6. 중첩 함수

- 단일 행 함수는 여러 번 중첩될 수 있음
- 중첩 함수는 가장 안쪽부터 바깥쪽 순으로 계산

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




```
SQL> SELECT   ename,
  2           NVL(TO_CHAR(mgr),'No Manager')
  3  FROM     emp;
```