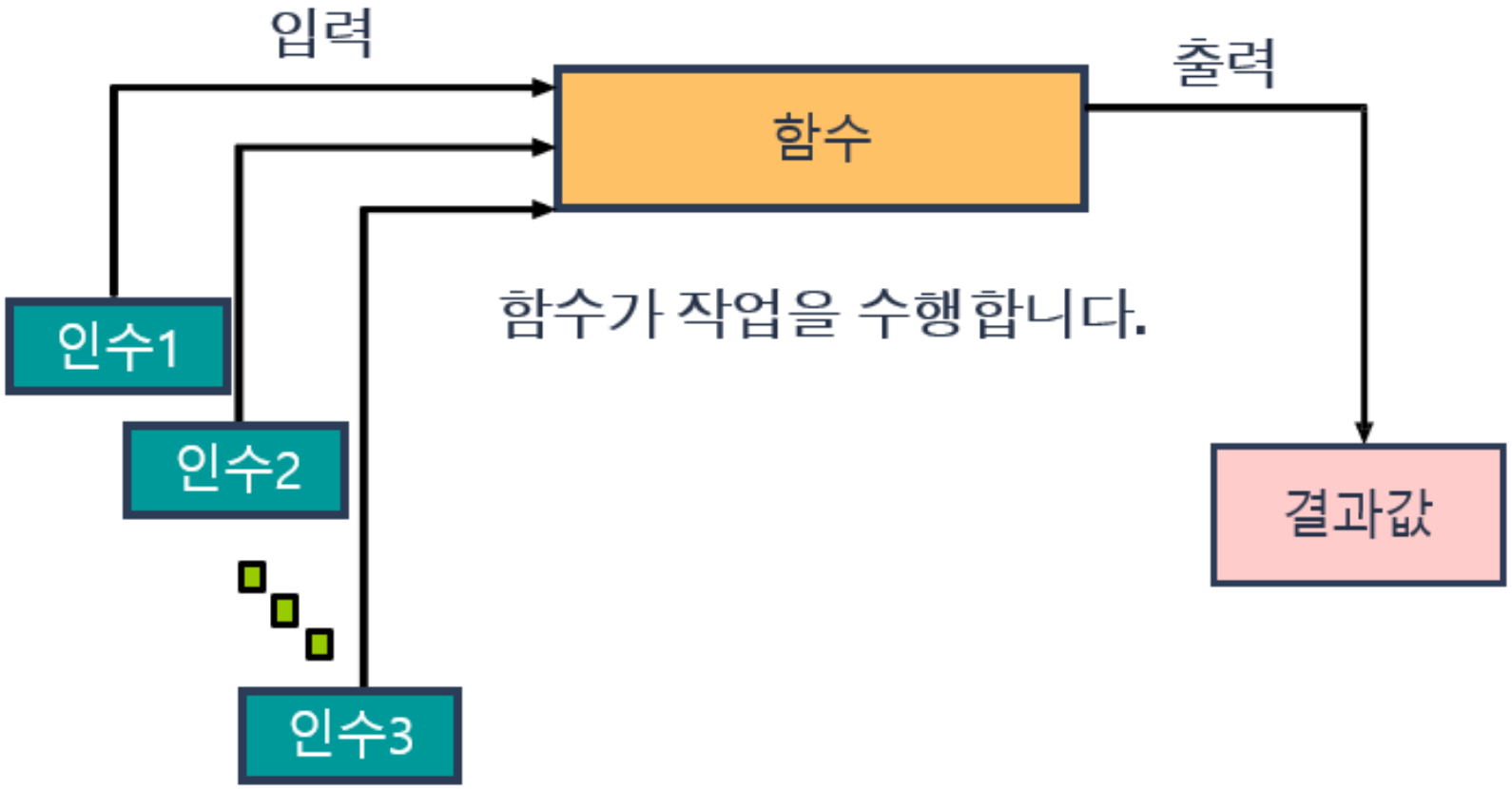
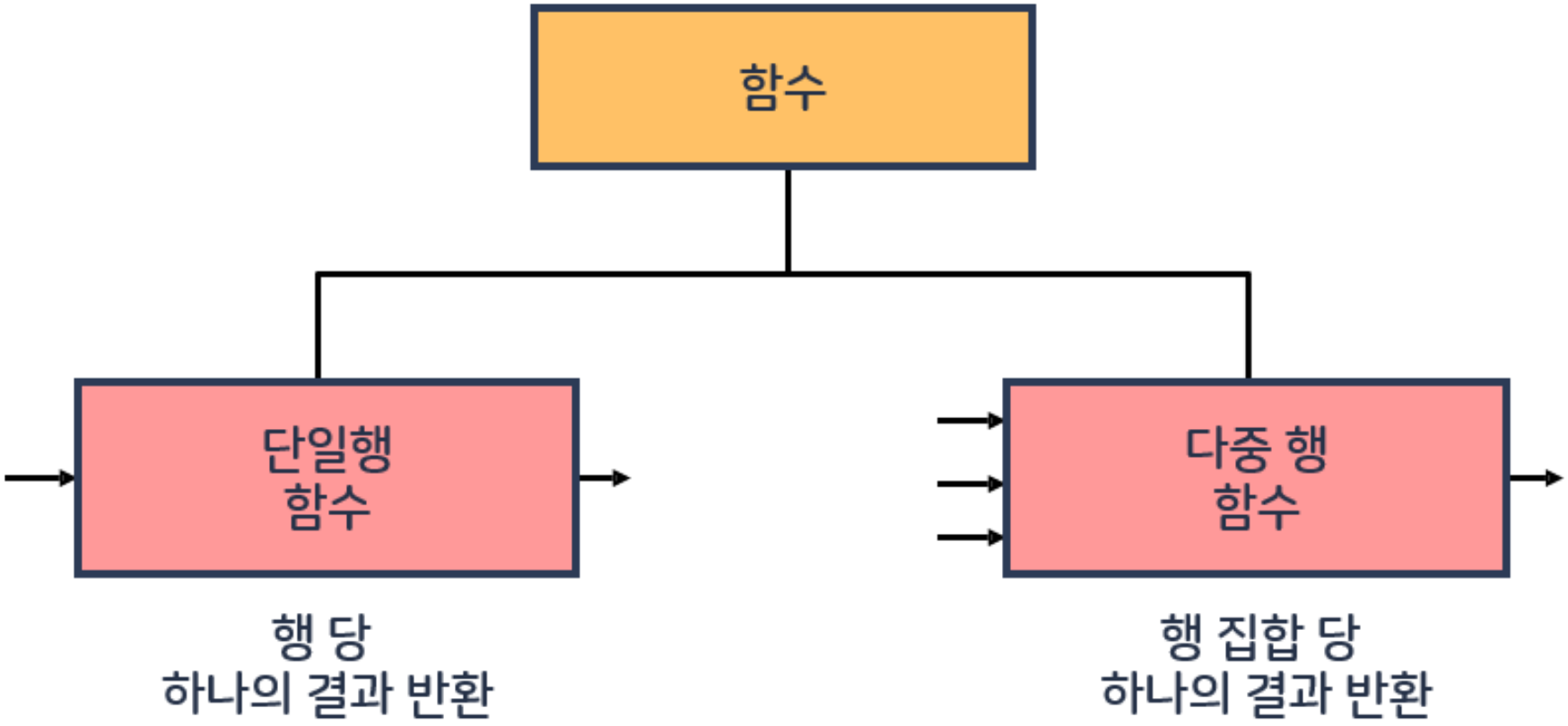
3. SQL 단일행 함수

SQL 함수



SQL 함수의 두 가지 유형

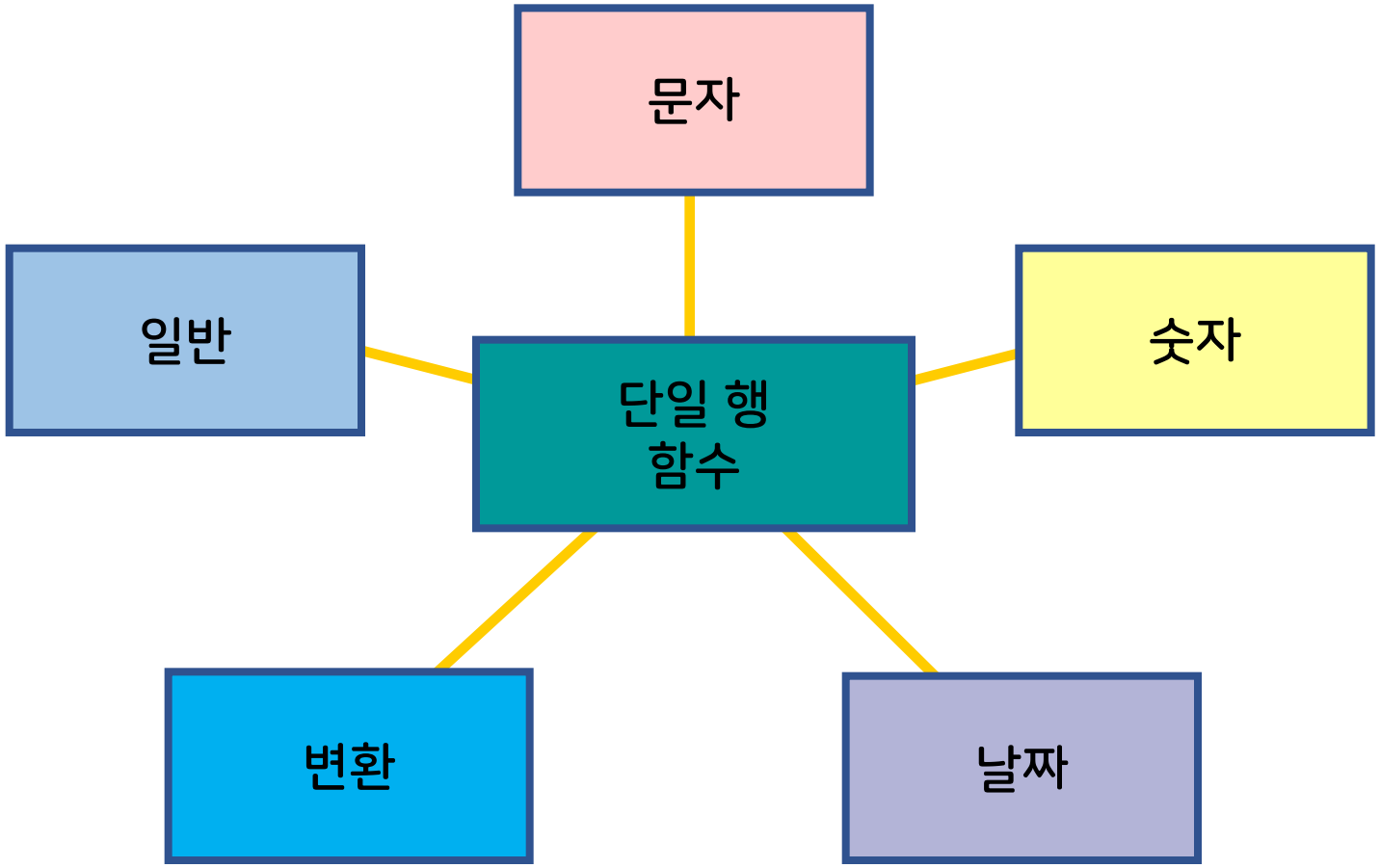


단일 행 함수

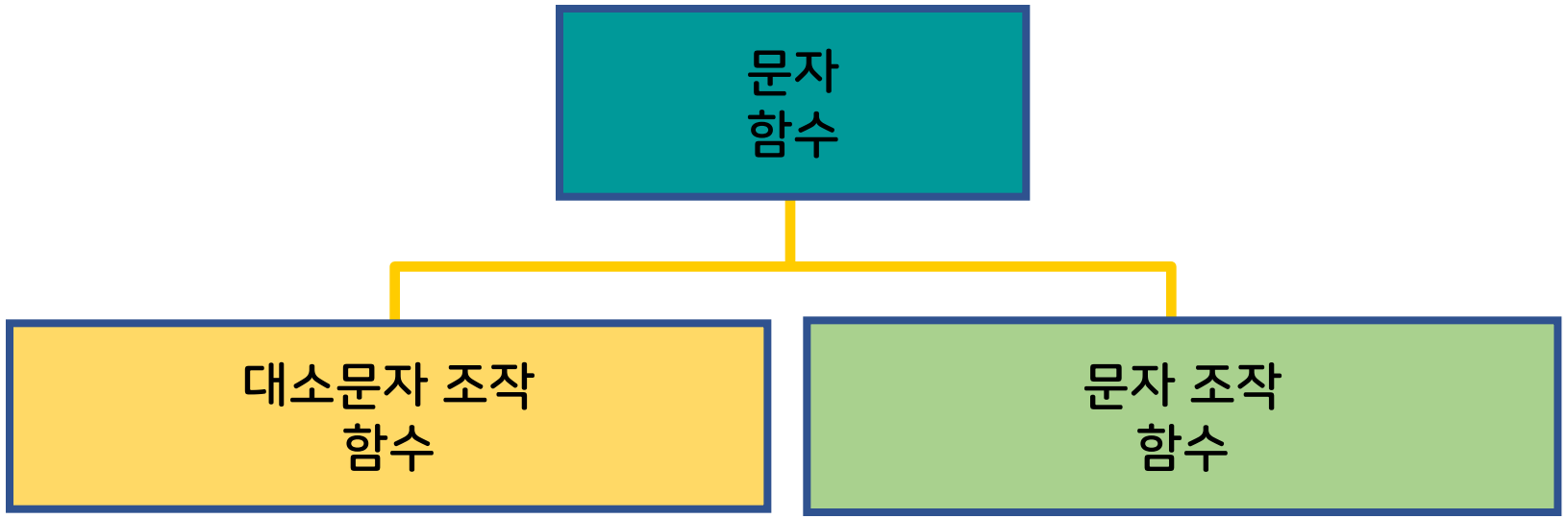
- 데이터 항목을 조작합니다.
- 인수를 사용하고 하나의 값을 반환합니다.
- 열이나 표현식을 인수로 사용할 수 있습니다.
- 반환되는 각 행에서 실행되어 행당 하나의 결과를 반환합니다.
- 중첩될 수 있습니다.
- 구문

```
function_name [(arg1, arg2,...)]
```

단일 행 함수



문자 함수



LOWER
UPPER
INITCAP

CONCAT
SUBSTR
LENGTH/LENGTHB
INSTR
LPAD/RPAD
TRIM/LTRIM/RTRIM
REPLACE

대소문자 조작 함수

- 문자열의 대소문자를 변환합니다.
  - LOWER: 대소문자 또는 대문자 문자열을 소문자로 변환합니다.
  - UPPER: 대소문자 또는 소문자 문자열을 대문자로 변환합니다.
  - INITCAP: 각 단어의 첫 문자는 대문자로, 나머지 문자는 소문자로 변환합니다.

함수	결과
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course

대소문자 조작 함수 사용

- 더미테이블인 DUAL을 이용하여 함수의 결과를 확인할 수 있습니다.

```
SELECT LOWER('SQL Course'), UPPER('SQL Course'),
       INITCAP('SQL Course')
FROM   dual;
```

- 다음 예제는 사원의 이름은 대문자로, 업무ID(JOB_ID)는 소문자로 출력됩니다.

```
SELECT 'The job id for '||UPPER(last_name)||' is '
       ||LOWER(job_id) AS "EMPLOYEE DETAILS"
FROM   employees;
```

WHERE 절에서의 대소문자 조작 함수 사용

- 다음 예제는 모두 사원 Higgins의 사원 번호, 성 및 부서 번호를 표시합니다.

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  last_name = 'higgins';
```

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  LOWER(last_name) = 'higgins';
```

```
SELECT employee_id, UPPER(last_name), department_id
FROM   employees
WHERE  INITCAP(last_name) = 'Higgins';
```

문자 조작 함수

- 문자 조작 함수를 사용하여 문자열을 조작합니다.
  - CONCAT: 값을 결합합니다. CONCAT에는 파라미터를 두 개만 사용할 수 있습니다.
  - SUBSTR: 지정한 길이의 문자열을 추출합니다.
  - LENGTH: 문자열의 길이를 숫자 값으로 표시합니다.
  - INSTR: 지정한 문자의 위치를 숫자로 표시합니다.
  - LPAD: 문자 값을 오른쪽 정렬하고 빈 곳을 지정한 문자열로 채웁니다.
  - RPAD: 문자 값을 왼쪽 정렬하고 빈 곳을 지정한 문자열로 채웁니다.
  - TRIM: 문자열에서 접두어나 접미어 또는 두 가지 모두를 자릅니다.

문자 조작 함수

- 문자열을 조작합니다.

함수	결과
CONCAT('Hello', 'World')	HelloWorld
SUBSTR('HelloWorld',1,5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', '1')	3
LPAD(salary,10,'*')	*****24000
RPAD(salary, 10, '*')	24000*****
TRIM('H'FROM 'HelloWorld')	elloWorld

문자 조작 함수의 사용 예제

```
SELECT CONCAT('Hello', 'World') FROM dual;
```

```
SELECT SUBSTR('HelloWorld',1,5) FROM dual;
```

```
SELECT LENGTH('HelloWorld'), LENGTHB('HelloWorld') FROM dual;
```

```
SELECT INSTR('HelloWorld', '1') FROM dual;
```

```
SELECT RPAD(last_name, 15,'*'), LPAD(salary,10,'*')FROM dual;
```

```
SELECT TRIM('H'FROM 'HelloWorld') FROM dual;
```

TRIM 함수

- 문자열 내 접두어 또는 접미어 제거 함수입니다.

```
SELECT TRIM('w' FROM 'window’),
       TRIM(LEADING 'w' FROM 'window’),
       TRIM(TRAILING 'w' FROM 'window')
FROM dual;
```

```
SELECT TRIM('0' FROM '000123456789000000')
FROM dual;
```

```
SELECT TRIM('ab' FROM 'ababxxxyyyababababab')
FROM dual;
ORA-30001: trim set should have only one character
```

LTRIM/RTRIM 함수

- 반복적인 문자 제거에 사용합니다.
- 다음 예제는 각각 오른쪽 또는 왼쪽의 'ab' 문자를 제거합니다.

```
SELECT RTRIM('ababxxxababyyyababababab', 'ab'),
       LTRIM('ababxxxababyyyababababab', 'ab')
FROM dual;
```

- 다음 예제는 함수를 중첩하여 양쪽 끝의 'ab'문자를 동시에 제거합니다.

```
SELECT RTRIM(LTRIM('ababxxxababyyyababababab', 'ab'), 'ab')
FROM dual;
```

문자 조작 함수 사용 예제 1

- 다음 예제는 업무 ID의 네번째 문자부터 REP라는 문자열이 포함된 모든 사원에 대해 성과 이름을 합친 전체 이름, 성의 길이, 성에서 문자 a의 위치를 표시합니다

```
SELECT employee_id, CONCAT(first_name, last_name) NAME,
       job_id, LENGTH (last_name),
       INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(job_id, 4) = 'REP';
```

문자 조작 함수 사용 예제 2

- 다음 예제는 성이 n으로 끝나는 사원에 대한 데이터를 표시합니다.

```
SELECT employee_id, CONCAT(first_name, last_name) NAME,
          LENGTH (last_name), INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(last_name, -1, 1) = 'n';
```

- 다음 예제는 사원번호, 성 및 이메일주소를 표시합니다. 이메일주소는 저장된 이메일 계정명과 회사 URL을 연결하여 출력합니다.

```
SELECT employee_id, last_name,
       CONCAT(email,'@oracle.com') AS "Email Address"
FROM employees;
```

숫자함수

함수	설명	예시	결과
ROUND	지정된 숫자의 특정 위치에서 반올림한 값 반환	ROUND(45.926, 2)	45.93
TRUNC	지정된 숫자의 특정 위치에서 버림한 값을 반환	TRUNC(45.926, 2)	45.92
MOD	지정된 숫자를 인수로 나눈 나머지 값 반환	MOD(1600, 300)	100
CEIL	지정된 숫자보다 큰 정수 중 가장 작은 정수 반환	CEIL(3.14)	4
FLOOR	지정된 숫자보다 작은 정수 중 가장 큰 정수 반환	FLOOR(3.14)	3

ROUND와 TRUNC 함수 사용

- ROUND 함수는 열, 표현식 또는 값을 소수점 n째 자리로 반올림합니다
  - DUAL은 함수 및 계산 결과를 보는 데 사용할 수 있는 더미 테이블(dummy table)입니다.

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

- TRUNC 함수는 열, 표현식 또는 값을 소수점 n째 자리까지 남기고 버립니다.

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

MOD 함수 사용

- 업무가 영업 사원인 모든 사원에 대해 급여를 5000으로 나눈 나머지를 계산합니다.

```
SELECT last_name, salary, MOD(salary, 5000)
FROM   employees
WHERE  job_id = 'SA_REP';
```

  - MOD 함수는 값이 홀수인지 짝수인지를 확인하는 경우에도 많이 사용됩니다.

CEIL과 FLOOR 함수 사용

- 다음 실습을 통해서 양수와 음수의 경우 CEIL과 FLOOR 함수의 결과를 비교해 봅니다.

```
SELECT CEIL(4.457), CEIL(-4.457), FLOOR(4.457), FLOOR(-4.457)
FROM dual;
```

  - 음수와 양수의 경우 각각의 결과를 비교해 봅니다.

날짜 사용

- 오라클 데이터베이스는 내부 숫자 형식(세기, 연도, 월, 일, 시, 분, 초)으로 날짜를 저장합니다.
- 국가별 기본 날짜 표시 형식이 있습니다.
- 연도의 마지막 두 자리만 지정하여 20세기에 21세기 날짜를 저장할 수 있습니다.
  - 동일한 방식으로 21세기에 20세기 날짜를 저장할 수 있습니다.
- 날짜 실습에 EMPLOYEES 테이블의 HIRE_DATE 열을 사용합니다.

```
SELECT last_name, hire_date
FROM   employees
WHERE  last_name like 'G%';
```

날짜 관련 시스템 함수

- SYSDATE와 CURRENT_DATE 데이터유형이 DATE인 함수로 다음을 반환합니다.
  - 날짜
  - 시간(데이터유형 변환 필요)

서버 시스템의 날짜를 기본으로 반환

```
SELECT sysdate
FROM dual;
```

사용자 시스템의 날짜를 기본으로 반환

```
SELECT current_date
FROM dual;
```

  - 데이터 유형 변환에 대해서는 이후 변환함수에서 다루어 집니다.

날짜 관련 시스템 함수

- SYSTIMESTAMP와 CURRENT_TIMESTAMP는 데이터유형이 DATETIME인 함수로 다음을 반환합니다.
  - 날짜
  - 소수 표시 초 단위 시간 및 시간대(Time Zone)

서버 시스템의 날짜와 초단위 시간 및 시간대를 기본으로 반환

```
SELECT systimestamp
FROM dual;
```

사용자 시스템의 날짜와 초단위 시간 및 시간대를 기본으로 반환

```
SELECT current_timestamp
FROM dual;
```

날짜 계산

- 날짜에 숫자를 더하거나 빼서 날짜 값을 계산합니다.
- 한 날짜에서 다른 날짜를 빼서 날짜 간의 일 수를 알 수 있습니다.
- 예제는 부서 90에 있는 모든 사원의 이름과 근무한 주 수를 표시합니다.
  - 현재 날짜(SYSDATE)에서 입사일을 뺀 후 결과를 7로 나누어 사원이 근무한 주 수를 계산합니다

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS
FROM   employees
WHERE  department_id = 90;
```

날짜 함수

함수	설명
MONTHS_BETWEEN(date1, date2)	두 날짜간의 달수
ADD_MONTHS(date, n)	날짜에 달 수 더하기
NEXT_DAY(date, 'char’)	이후 날짜며 지정한 요일 ('char')에 해당하는 날짜
LAST_DAY(date)	해당 달의 마지막 달
EXTRACT([YEAR¦MONTH¦DAY] FROM date)	날짜로 부터 지정된 필드의 값을 추출하여 반환

날짜 함수 사용

- 다음은 모든 사원의 사원 번호, 입사일, 근무한 개월 수, 입사 6개월 후는 직무능력 검사일, 교육시작일 및 첫 급여일을 표시합니다.
- 근무기간은 버림을 적용하여 정수로 나타내고, 교육시작일은 입사 후 처음 돌아오는 월요일 이며, 첫 급여일은 입사한 다음 달 10일입니다.

```
SELECT employee_id, hire_date,
         TRUNC(MONTHS_BETWEEN (SYSDATE, hire_date)) 근무기간,
         ADD_MONTHS (hire_date, 6) 직무능력검사일,
         NEXT_DAY (hire_date, '월요일') 교육시작일,
         LAST_DAY(hire_date)+10 첫급여일
  FROM   employees;
```

EXTRACT 함수 사용

- SYSDATE에서 YEAR를 추출합니다.

```
SELECT EXTRACT(year FROM sysdate)
FROM dual;
```

- 다음은 사원들의 입사일로 부터 MONTH를 표시합니다.

```
SELECT last_name, hire_date,
       EXTRACT (MONTH FROM hire_date) AS 입사일
FROM employees;
```

날짜데이터에 ROUND와 TRUNC 함수 사용

- 지정한 형식모델에 따라 함수가 반올림되거나 버려지므로 날짜를 가장 가까운 연도 또는 달로 반올림 또는 버림할 수 있습니다.
- 형식모델

형식모델	기준
YEAR/YYYY	한 해의 7월 1일을 기준으로 지정된 날짜를 당해 혹은 다음해의 1월1일로 초기화
MONTH/MM	각 달의 중간(예 : 15일) 지정된 날짜를 당월 혹은 다음 월의 1일로 초기화
DD	정오를 기준으로지정된 날짜를 특정일의 자정으로 초기화
D	수요일을 기준으로 지정된 날짜를 전후 일요일로 초기화

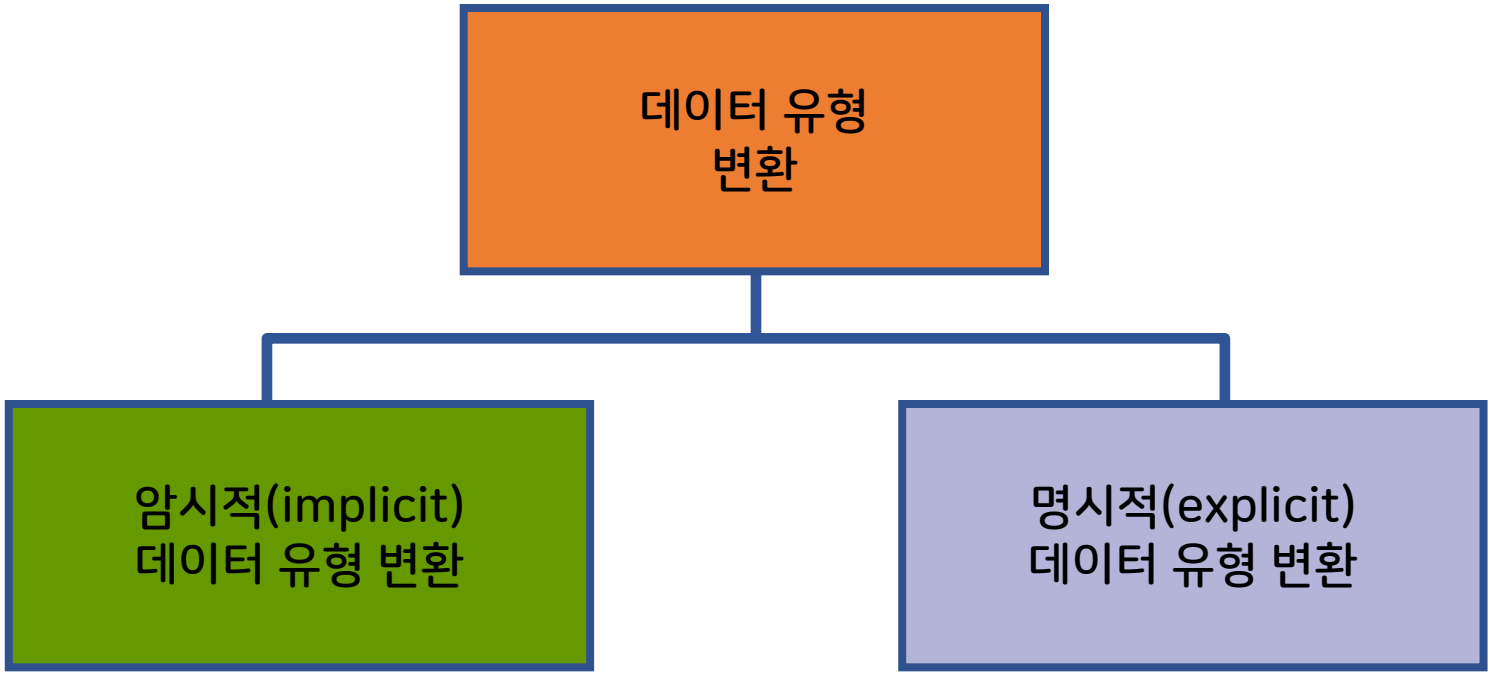
날짜데이터에 ROUND와 TRUNC 함수 사용

- SYSDATE를 기준으로 다음을 실행하여 비교합니다.

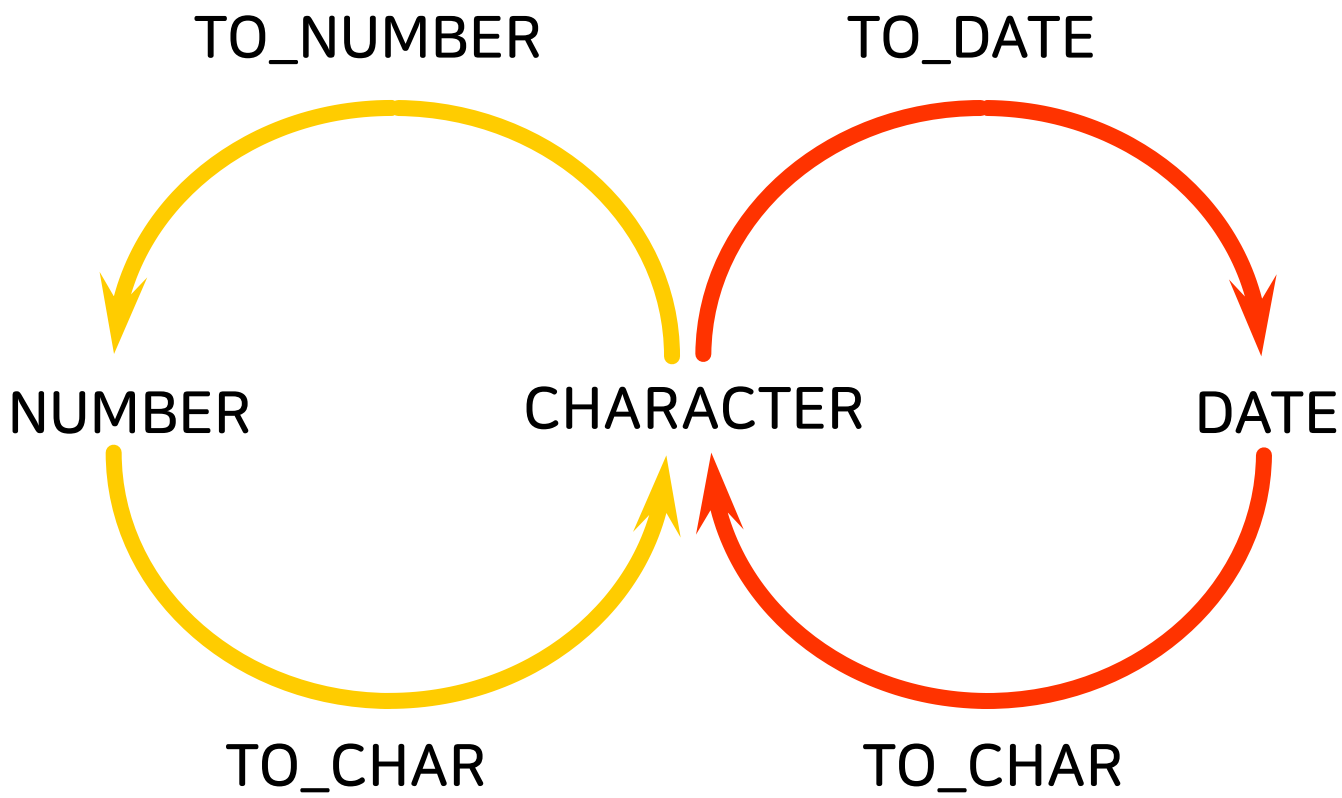
```
SELECT ROUND(sysdate, 'YEAR'), ROUND(sysdate, 'MONTH’),
       ROUND(sysdate, 'DD'), ROUND(sysdate, 'D')
FROM dual;
```

```
SELECT TRUNC(sysdate, 'YEAR'), TRUNC(sysdate, 'MONTH'),
       TRUNC(sysdate, 'DD'), TRUNC(sysdate, 'D')
FROM dual;
```

변환 함수



명시적(explicit) 데이터 유형 변환



TO_CHAR 함수의 날짜관련 Format Model

- Format_model은 작은 따옴표로 묶어야 하며 대소문자를 구분합니다.
- Format_model은 모든 유효한 날짜 형식 요소를 포함할 수 있습니다.
- Format_model은 채워진 공백을 제거하거나 선행 제로를 제거하는 fm 요소를 사용할 수 있습니다.
- Format_model은 쉼표로 날짜 값과 구분합니다.

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

날짜 형식 모델 요소

Format_model	의미
YYYY / RRRR	연도
MM	두자리 숫자 값으로 나타낸 달
MONTH	월 전체 이름
MON	세 자의 약어로 나타낸 월 이름
DAY	요일 전체 이름
DY	세 자의 약어로 나타낸 요일 이름
DD	숫자로 나타낸 달의 일
Q	숫자로 나타낸 분기

시간 형식 모델 요소

Format_model	의미
HH	시간 (1 ~ 12)
HH24	시간 (0 ~23)
MI	분(0 ~ 59)
SS	초(0 ~ 59)
AM/PM	오전/오후 표시

날짜데이터에 TO_CHAR 함수 사용

- 다음 SQL 문은 모든 사원의 이름과 입사일을 표시합니다.

```
SELECT last_name,
       TO_CHAR(hire_date, 'DD MM YYYY')
       AS HIREDATE
FROM   employees;
```

- 오늘 날짜를 2023 07 12 15:21:20 형식으로 표시합니다.

```
SELECT TO_CHAR(sysdate, 'YYYY MM DD HH24:MI:SS')
FROM   dual;
```

숫자데이터에 TO_CHAR 함수 사용

- TO_CHAR 함수에는 숫자 값을 문자로 표시할 수 있는 다음과 같은 몇 가지 형식 요소가 있습니다.

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

Format_model	의미
9	숫자를 표시
0	0을 강제로 표시
$	부동통화기호 표시
l	지역통화기호 표시
.	소수점 출력
,	천 단위 구분기호 출력

숫자데이터에 TO_CHAR 함수 사용

- 다음 예제는 모든 사원들의 사원번호, 이름, 부동통화기호 $표시와 천단위 구분기호를 포함한 급여를 출력합니다.

```
SELECT employee_id, last_name,
       TO_CHAR(salary, '$999,999') SALARY
FROM   employees;
```

- 다음 예제는 모든 사원들의 사원번호, 이름, 지역통화기호와 천단위 구분기호를 포함한 급여를 출력합니다.

```
SELECT employee_id, last_name,
       TO_CHAR(salary, 'L999,999') SALARY
FROM   employees;
```

숫자데이터에 TO_CHAR 함수 사용

- 다음 예제는 형식모델 9와 0을 비교 합니다.

```
SELECT employee_id, last_name,
       TO_CHAR(salary, '$999,999.00') SALARY1,
       TO_CHAR(salary, '$099,999.00') SALARY2
FROM   employees;
```

- 다음 예제는 세션에서 지역을 변경한 후 모든 사원들의 사원번호, 이름, 해당 지역의 통화기호와 천단위 구분기호를 포함한 급여를 출력합니다.

```
ALTER SESSION SET nls_territory='Japan';
SELECT employee_id, last_name, TO_CHAR(salary, 'L99,999') SALARY
FROM   employees;
ALTER SESSION SET nls_territory='Korea';
```

TO_NUMBER 및 TO_DATE 함수

- TO_NUMBER 함수를 사용하여 문자열을 숫자 형식으로 변환합니다.

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

- TO_DATE 함수를 사용하여 문자열을 날짜 형식으로 변환합니다.

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

TO_NUMBER 및 TO_DATE 함수의 사용

- TO_NUMBER 함수를 사용하여 급여가 $8,000보다 많은 사원을 검색합니다.

```
SELECT employee_id, last_name, salary, job_id
FROM   employees
WHERE salary > TO_NUMBER('$8,000','$9,999');
```

- TO_DATE 함수를 사용하여 2017년 1월 1일 이후에 입사한 사원을 검색합니다.

```
SELECT employee_id, last_name, hire_date, department_id
FROM   employees
WHERE hire_date >= TO_DATE('01-01-2017','dd-mm-yyyy');
```

RR 날짜 형식

구분		지정한 연도(두자리)	
		0 - 49	50 -99
현재 연도 (두자리)	0 -49	반환일이 현재 세기	반환일이 이전세기
	50 -99	반환일이 다음 세기	반환일이 현재 세기

RR 날짜 형식 예제

- 1999년 12월 31일 후에 고용된 사원을 찾는 경우 RR 형식을 사용하면 명령을 1999년에 실행하든 지금 실행하든 동일한 결과를 얻습니다.

```
SELECT last_name, TO_CHAR(hire_date, 'yyyy/mm/dd')
FROM  employees
WHERE hire_date > TO_DATE('31-12-99', 'DD-MM-RR');
```

- 다음 명령은 YY 형식이 날짜의 연도 부분을 현재 세기(2099)로 해석하므로, 행이 선택되지 않습니다.

```
SELECT last_name, TO_CHAR(hire_date, 'yyyy/mm/dd')
FROM  employees
WHERE hire_date > TO_DATE('31-12-99', 'DD-MM-YY');
```

일반 함수

- 다음 함수에는 모든 데이터 유형을 사용할 수 있으며 널도 사용할 수 있습니다.

함수	설명
NVL (expr1, expr2)	널값을 실제값으로 변환
NVL2 (expr1, expr2, expr3)	expr1이 null이 아닌 경우 NVL2는 expr2를 반환 expr1이 null인 경우 NVL2는 expr3을 반환
COALESCE (expr1, expr2, ..., exprn)	표현식 리스트에서 null이 아닌 첫번째 표현식을 반환

NVL 함수

- 널을 실제 값으로 변환합니다.
- 사용되는 데이터 유형은 날짜, 문자 및 숫자입니다.
- 데이터 유형은 서로 일치해야 합니다.
  - NVL(commission_pct,0)
  - NVL(hire_date,'01-JAN-07')
  - NVL(job_id,'No Job Yet')

NVL 함수 사용

- 모든 사원의 연간 총수입을 계산하기위해 월급에 12를 곱한 후 커미션 비율을 더합니다.

```
SELECT last_name, salary, NVL(commission_pct, 0),
   (salary*12) + (salary*12*commission_pct) AN_SAL
FROM employees;
```

커미션이 널인 사원은 연간 총수입도 널값 출력

```
SELECT last_name, salary, NVL(commission_pct, 0),
   (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL
FROM employees;
```

커미션이 널인 사원의 연간 총수입도 출력

NVL2 함수 사용

- 다음 예제의 비고 열은 COMMISSION_PCT 열을 검사합니다.
  - 값이 발견되면 두번째 표현식인 COMM이 반환됩니다.
  - COMMISSION_PCT 열에 널 값이 있으면 세번째 표현식인 NOCOMM이 반환됩니다.

```
SELECT  last_name, salary, commission_pct,
        (salary*12) + (salary*12*NVL(commission_pct, 0)) AS 연봉,
        NVL2(commission_pct,'COMM', 'NOCOMM') AS 비고
FROM   employees WHERE department_id IN (50, 80);
```

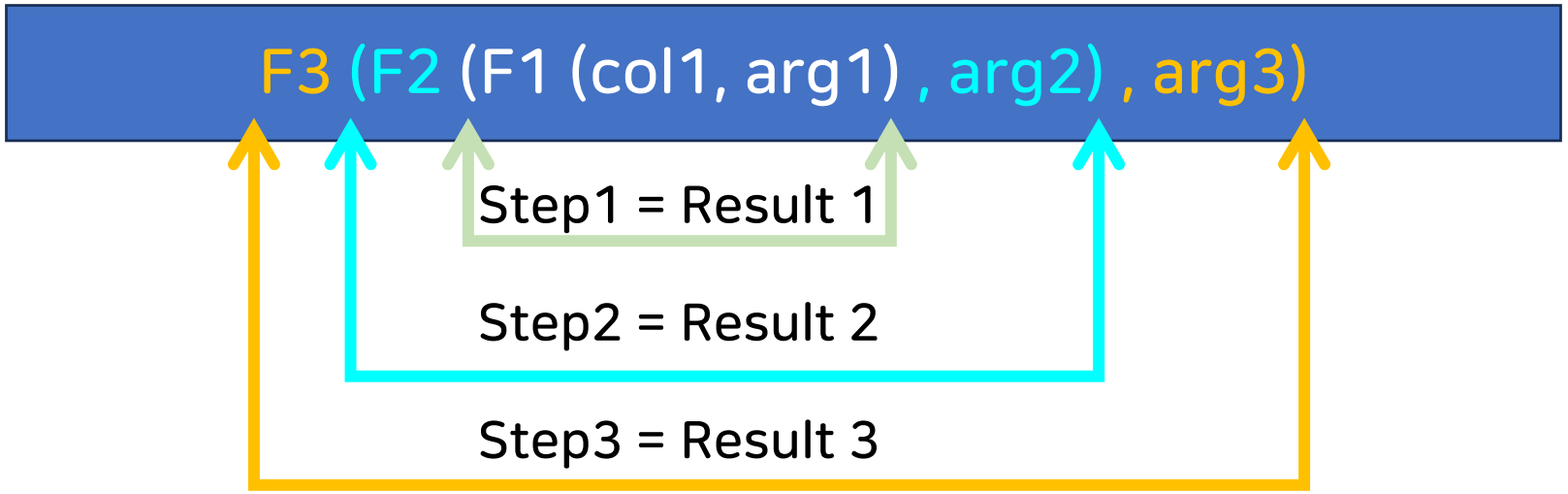
COALESCE 함수 사용

- 예제에서, COMMISSION_PCT 값이 널이 아닌 경우, 이 값이 표시됩니다.
  - COMMISSION_PCT 값이 널인 경우, SALARY가 표시됩니다.
  - COMMISSION_PCT와 SALARY 값이 모두 널인 경우, 값 10이 표시됩니다.

```
SELECT   last_name,
         COALESCE(commission_pct, salary, 10) comm
FROM     employees
ORDER BY commission_pct;
```

함수의 중첩

- 단일 행(row) 함수는 여러 번 중첩될 수 있습니다.
- 중첩 함수는 가장 안쪽부터 바깥쪽 순으로 계산됩니다.



함수의 중첩 (1)

- 다음 예제는 부서 60에 있는 사원의 성 4문자와 '_US'를 함께 표시합니다.

```
SELECT last_name,
UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 4), '_US'))
FROM employees
WHERE department_id = 60;
```

- 다음은 관리자가 없는 사원을 'No Manager'로 표시합니다

```
SELECT last_name,
       NVL(TO_CHAR(manager_id), 'No Manager')
FROM   employees
WHERE  manager_id IS NULL;
```

함수의 중첩 (2)

- 다음은 채용날짜로 부터 6개월이 경과한 날로부터 돌아오는 첫번째 금요일을 표시합니다.
- 결과날짜는 2023-01-03 금요일'과 같은 형식으로 나타내고 결과는 채용날짜별로 정렬합니다.

```
SELECT TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), '금요일'),
               'YYYY-MM-DD DAY') AS "Next 6 Month to Review"
FROM employees
ORDER BY hire_date;
```

조건 표현식

- SQL 문 안에서 IF-THEN-ELSE 논리를 사용할 수 있도록 해 줍니다.
- 다음 두 방법을 사용합니다.
  - CASE 표현식
  - DECODE 함수
- CASE 표현식은 ANSI SQL을 준수합니다.
- DECODE는 오라클 구문에만 있습니다.

CASE 표현식

- IF-THEN-ELSE 문의 역할을 수행하여 조건부 조회를 손쉽게 수행할 수 있습니다.
- 구문

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

  - expr이 comparison_expr과 동일한 첫번째 WHEN ... THEN 쌍을 찾아서 return_expr을 반환합니다.
  - 이 조건을 만족하는 WHEN ... THEN 쌍이 없고 ELSE 절이 존재하는 경우, else_expr을 반환하고 ELSE 절이 존재하지 않는 경우 널을 반환합니다

CASE 표현식 사용 예제

- 다음 예제는 JOB_ID에 따라 다음과 같이 인상된 급여를 출력합니다.
  - IT_PROG면 급여가 10% 인상
  - JOB_ID가 ST_CLERK이면 급여가 15% 인상
  - JOB_ID가 SA_REP면 급여가 20% 인상
  - 다른 업무에 대한 급여 인상은 없습니다.

```
SELECT last_name, job_id, salary,
       CASE job_id WHEN 'IT_PROG'  THEN  1.10*salary
                   WHEN 'ST_CLERK' THEN  1.15*salary
                   WHEN 'SA_REP'   THEN  1.20*salary
       ELSE      salary END     "REVISED_SALARY"
FROM   employees;
```

CASE 표현식의 활용

- WHEN 절 다음에 expr과 비교연산자를 사용하여 보다 다양한 값 비교를 수행할 수 있습니다.
- 구문

```
CASE WHEN expr operator expr1 THEN return_expr1
          [WHEN expr operator expr2 THEN return_expr2
           WHEN expr operator exprn THEN return_exprn
           ELSE else_expr]
END
```

  - Operator에는 비교연산자(=, <>, >, >= 등)와 BETWEEN…AND…, IN, LIKE, IS NULL 등이 올 수 있습니다.

CASE 표현식 활용 예제

- 다음 예제는 이름, 직급, 급여 및 다음과 같이 급여등급을 표시합니다.
- 급여등급 표시 기준
  - 급여가 9000보다 많으면 'A'
  - 4000이상 9000이하이면 'B'
  - 4000미만이면 'C'

```
SELECT last_name, job_id, salary,
       CASE WHEN salary > 9000 THEN  'A'
            WHEN salary BETWEEN 4000 AND 9000 THEN  'B'
            ELSE 'C' END AS "급여등급"
FROM   employees;
```

DECODE 함수

- DECODE 함수로 CASE 식 IF-THEN-ELSE 문의 역할을 수행하여 조건부 조회를 손쉽게 수행할 수 있습니다.

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

  - expression을 각 search 값과 비교한 후 해독하여 expression이 search 값과 동일하면 result를 반환합니다.
  - 기본값이 생략되면 검색 값에 결과 값과 일치하는 값이 없으므로 널 값이 반환됩니다.

DECODE 함수 사용 예제

- 다음 예제는 JOB_ID에 따라 다음과 같이 인상된 급여를 출력합니다.
  - IT_PROG면 급여가 10% 인상
  - JOB_ID가 ST_CLERK이면 급여가 15% 인상
  - JOB_ID가 SA_REP면 급여가 20% 인상
  - 다른 업무에 대한 급여 인상은 없습니다.

```
SELECT last_name, job_id, salary,
       DECODE(job_id, 'IT_PROG',  1.10*salary,
                      'ST_CLERK', 1.15*salary,
                      'SA_REP',   1.20*salary,
              salary) AS REVISED_SALARY
FROM   employees;
```


Thank you