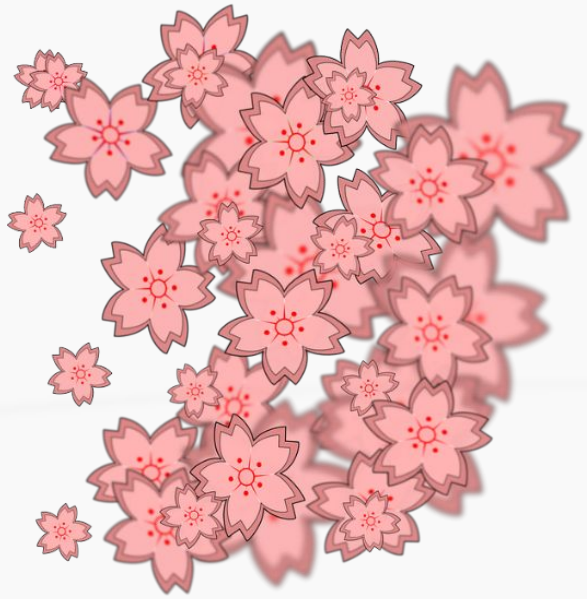
SQL for Dev

P
P
T

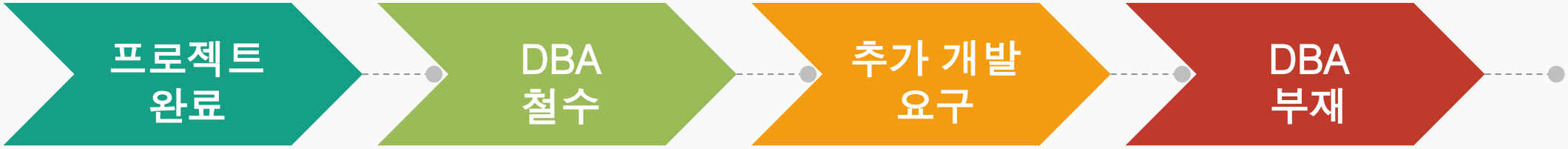
SQL CONCEPT



Sang-Taik. Jung
https://github.com/sangtaik

Contents

1. CONCEPT



1차 목적: DBA가 없는 상황에서 개발에 필요한 SQL을 정확하고,
효율적으로 작성할 수 있다.

2차 목적: 유지보수가 쉬운 SQL을 작성할 수 있다.

2. JOIN

의미

- 관계형 데이터 모델에서 관계를 실제로 구현한 명령어

기준 집합

- 결과집합의 구성에 결정적인 영향을 주는 집합
  ROW 수의 기준이 되는 집합

참조 집합

- 결과집합의 구성에 컬럼 단위로 참조되는 집합

INNER JOIN

- 가장 일반적이고 기본적인 조인
  조인 조건에 맞는 행만 결과를 출력

OUTER JOIN

- 테이블에 값이 없어도 결과를 출력

  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN

CROSS JOIN

- 조인되는 두 테이블에서 곱집합을 반환
  M행 테이블과 N행의 테이블을 조인하면 M*N행을 생성

JOIN 구조

01 SQL

```
SELECT A.사원번호
     ,A.사원명
     ,B.부서명
     ,B.지역
FROM 사원 A
INNER JOIN 부서 B ON
(A.부서코드 = B.부서코드)
WHERE
A.급여구분 = '연봉'
AND B.지역     = '수원'
;
```

02 기준집합

사원 A

03 참조집합

부서 B

04 조인조건

(A.부서코드 = B.부서코드)

05 필터조건

A.급여구분 = '연봉'
AND B.지역 = '수원'

관계에 의한 기준집합 결정

01 SQL

```
SELECT A.사원번호
     ,B.사원명
     ,A.급여월
     ,A.월급여
FROM 급여지급 A
INNER JOIN 사원 B ON
(B.사원번호 = A.사원번호
AND B.퇴사일자 IS NULL)
WHERE
A.사원번호 = 101
;
```

02 기준집합(자식 테이블)

급여지급 A

03 참조집합(부모 테이블)

사원 B

조인의 종류에 의한 기준집합 결정

01 SQL

```
SELECT A.사원번호
     ,A.사원명
     ,A.부서코드
     ,B.기본수당액
     ,B.특별수당액
FROM 사원 A
LEFT OUTER JOIN 기타급여 B ON
(A.부서코드 = B.부서코드)
;
```

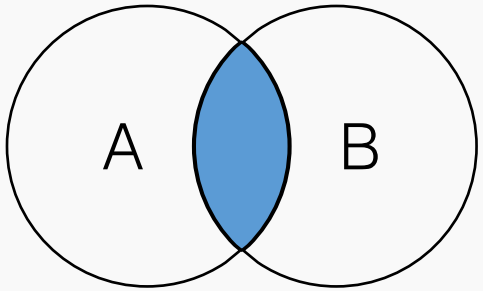
02 기준집합(조인하는 테이블)

사원 A

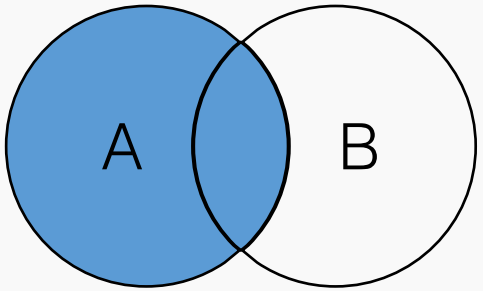
03 참조집합(조인되는 테이블)

기타급여 B

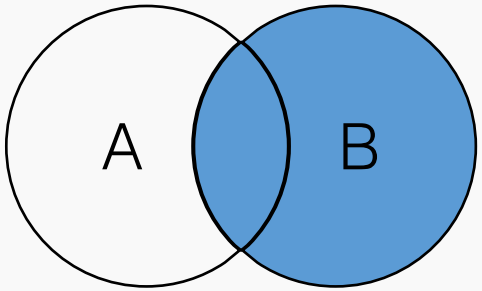
조인의 종류



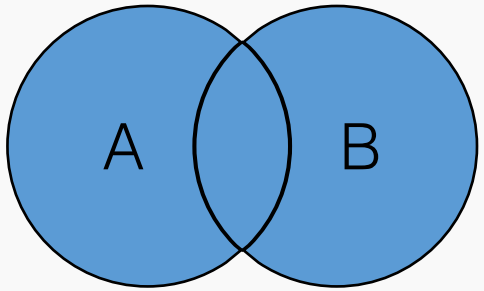
INNER JOIN



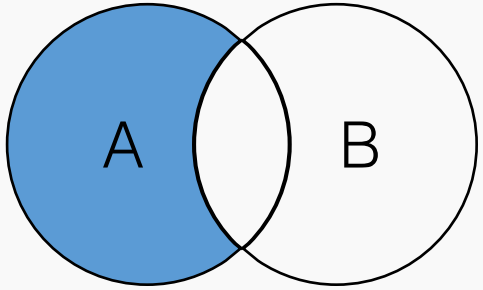
LEFT OUTER JOIN



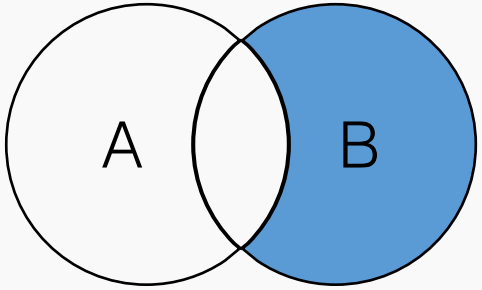
RIGHT OUTER JOIN



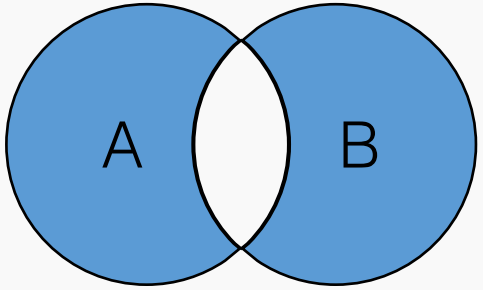
FULL OUTER JOIN



LEFT OUTER JOIN
WHERE B.KEY IS NULL



RIGHT OUTER JOIN
WHERE A.KEY IS NULL



FULL OUTER JOIN
WHERE A.KEY IS NULL
OR B.KEY IS

INNER VS OUTER

01 INNER JOIN

```
SELECT A.사원번호
     ,A.사원명
     ,B.연봉액
FROM 사원 A
INNER JOIN 연봉제급여 B ON
(B.사원번호 = A.사원번호
AND B.연봉액 > 16000000  )
WHERE
A.부서코드 = '30'
;
```

02 OUTER JOIN

```
SELECT A.사원번호
     ,A.사원명
     ,B.연봉액
FROM 사원 A
LEFT OUTER JOIN 연봉제급여 B ON
(B.사원번호 = A.사원번호
AND B.연봉액 > 16000000)
WHERE
A.부서코드 = '30'
;
```

INNER VS OUTER 결과

01 INNER JOIN

	사원번호	사원명	연봉액
1	109	전수경	16200000
2	111	김창완	18360000

기준 및 참조집합의
JOIN 조건에 일치하는 결과를
출력

02 OUTER JOIN

	사원번호	사원명	연봉액
1	101	조동진	(null)
2	102	한영애	(null)
3	107	오석준	(null)
4	109	전수경	16200000
5	111	김창완	18360000

기준집합의 조건에 일치하면
테이블에 값이 없어도 결과를
출력

CROSS JOIN

01 SQL

```
SELECT A.사원번호
     ,A.사원명
     ,B.연봉액
FROM 사원       A
CROSS JOIN 연봉제급여 B
WHERE A.부서코드 = '30'
;
```

02 특징

조인 조건이 없음
결과 행 수 = 기준테이블 행 수 x 참조 테이블 행 수

	사원번호	사원명	연봉액
1	101	조동진	15600000
2	101	조동진	15840000
3	101	조동진	17520000
4	101	조동진	16200000
5	101	조동진	18360000
6	102	한영애	15600000
7	102	한영애	15840000
8	102	한영애	17520000
9	102	한영애	16200000
10	102	한영애	18360000

	사원번호	사원명	연봉액
11	107	오석준	15600000
12	107	오석준	15840000
13	107	오석준	17520000
14	107	오석준	16200000
15	107	오석준	18360000
16	109	전수경	15600000
17	109	전수경	15840000
18	109	전수경	17520000
19	109	전수경	16200000
20	109	전수경	18360000

21	111	김창완	15600000
22	111	김창완	15840000
23	111	김창완	17520000
24	111	김창완	16200000
25	111	김창완	18360000

행렬변환

01 기준 행렬

```
SELECT 직급
     ,사원명
FROM 사원 A
WHERE 부서코드 = '30'
ORDER BY 직급
                ,사원명
;
```

행을 열로 변환하려면 최대 몇 개의 ROW가 존재하는 지 알아야 한다.

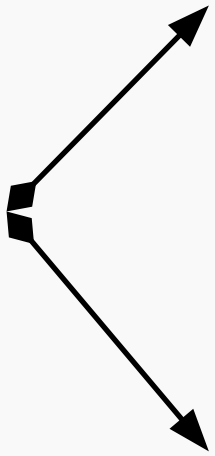
02 변환 행렬

```
SELECT 직급
     ,MAX(CASE WHEN RN = 1 THEN 사원명 END) 사원1
     ,MAX(CASE WHEN RN = 2 THEN 사원명 END) 사원2
     ,MAX(CASE WHEN RN = 3 THEN 사원명 END) 사원3
     ,MAX(CASE WHEN RN = 4 THEN 사원명 END) 사원4
     ,MAX(CASE WHEN RN = 5 THEN 사원명 END) 사원5
FROM (
    SELECT 직급
         ,사원명
         ,ROW_NUMBER() OVER (PARTITION BY 직급
    ORDER BY 사원명) RN
    FROM 사원 A
    WHERE 부서코드 = '30'
    )
GROUP BY 직급
ORDER BY 직급
;
```

ON 과 WHERE 차이

```
FROM 사원 A
LEFT OUTER JOIN 부서 B
ON
(
    A.부서코드 = B.부서코드
)
```

사원번호	사원명	부서명	지역
101	조동진	전산팀	수원
102	한영애	전산팀	수원
103	조규찬	영업팀	대전
104	이상은	인사팀	서울
105	조덕배	인사팀	서울
106	장필순	인사팀	서울
107	오석준	전산팀	수원
108	이규석	영업팀	대전
109	전수경	전산팀	수원
110	이선희	인사팀	서울
111	김창완	전산팀	수원
112	이현우	영업팀	대전



```
ON
(
    A.부서코드 = B.부서코드
    AND B.지역 = '수원'
)
```

사원번호	사원명	부서명	지역
101	조동진	전산팀	수원
102	한영애	전산팀	수원
103	조규찬	NULL	NULL
104	이상은	NULL	NULL
105	조덕배	NULL	NULL
106	장필순	NULL	NULL
107	오석준	전산팀	수원
108	이규석	NULL	NULL
109	전수경	전산팀	수원
110	이선희	NULL	NULL
111	김창완	전산팀	수원
112	이현우	NULL	NULL

```
ON
(
    A.부서코드 = B.부서코드
)
WHERE
B.지역 = '수원'
```

사원번호	사원명	부서명	지역
101	조동진	전산팀	수원
102	한영애	전산팀	수원
107	오석준	전산팀	수원
109	전수경	전산팀	수원
111	김창완	전산팀	수원

WHERE 실행

사원번호	사원명	부서코드
101	조동진	30
102	한영애	30
103	조규찬	20
104	이상은	10
105	조덕배	10
106	장필순	10
107	오석준	30
108	이규석	20
109	전수경	30
110	이선희	10
111	김창완	30
112	이현우	20

사원 테이블 ACCESS FULL

부서코드	부서명	지역
10	인사팀	서울
20	영업팀	대전
30	전산팀	수원

부서 테이블 ACCESS FULL

```
ON
(
    A.부서코드 = B.부서코드
)
WHERE
B.지역 = '수원'
```

부서코드	부서명	지역
30	전산팀	수원

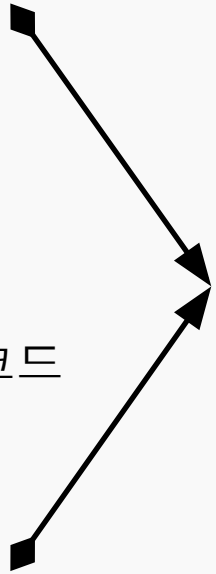
필터조건 B.지역 = ‘수원’ 적용

WHERE 실행

사원번호	사원명	부서코드
101	조동진	30
102	한영애	30
103	조규찬	20
104	이상은	10
105	조덕배	10
106	장필순	10
107	오석준	30
108	이규석	20
109	전수경	30
110	이선희	10
111	김창완	30
112	이현우	20

조인 조건
A.부서코드 = B.부서코드
발동

부서코드	부서명	지역
30	전산팀	수원



사원번호	사원명	부서코드	부서코드	부서명	지역
101	조동진	30	30	전산팀	수원
102	한영애	30	30	전산팀	수원
107	오석준	30	30	전산팀	수원
109	전수경	30	30	전산팀	수원
111	김창완	30	30	전산팀	수원

SELECT로 컬럼 선택

사원번호	사원명	부서명	지역
101	조동진	전산팀	수원
102	한영애	전산팀	수원
107	오석준	전산팀	수원
109	전수경	전산팀	수원
111	김창완	전산팀	수원

결과 추출

ON 실행

사원번호	사원명	부서코드
101	조동진	30
102	한영애	30
103	조규찬	20
104	이상은	10
105	조덕배	10
106	장필순	10
107	오석준	30
108	이규석	20
109	전수경	30
110	이선희	10
111	김창완	30
112	이현우	20

사원 테이블 ACCESS FULL

부서코드	부서명	지역
10	인사팀	서울
20	영업팀	대전
30	전산팀	수원

부서 테이블 ACCESS FULL

```
ON
(
    A.부서코드 = B.부서코드
    AND B.지역 = '수원
)
```

부서코드	부서명	지역
30	전산팀	수원

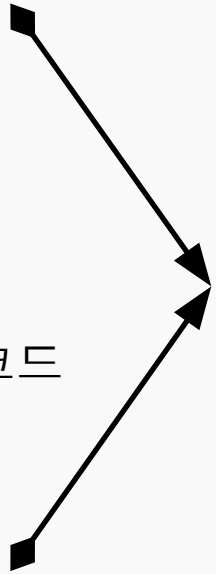
필터조건 B.지역 = ‘수원’ 적용

ON 실행

사원번호	사원명	부서코드
101	조동진	30
102	한영애	30
103	조규찬	20
104	이상은	10
105	조덕배	10
106	장필순	10
107	오석준	30
108	이규석	20
109	전수경	30
110	이선희	10
111	김창완	30
112	이현우	20

조인 조건
A.부서코드 = B.부서코드
발동

부서코드	부서명	지역
30	전산팀	수원



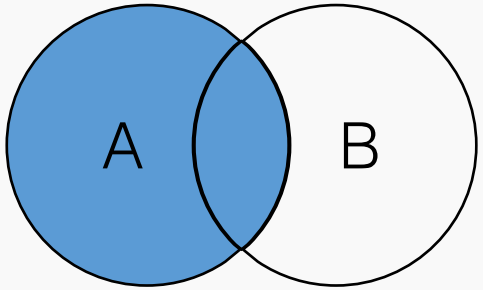
사원번호	사원명	부서코드	부서코드	부서명	지역
101	조동진	30	30	전산팀	수원
102	한영애	30	30	전산팀	수원
107	오석준	30	30	전산팀	수원
109	전수경	30	30	전산팀	수원
111	김창완	30	30	전산팀	수원

SELECT로 컬럼 선택

사원번호	사원명	부서명	지역
101	조동진	전산팀	수원
102	한영애	전산팀	수원
107	오석준	전산팀	수원
109	전수경	전산팀	수원
111	김창완	전산팀	수원

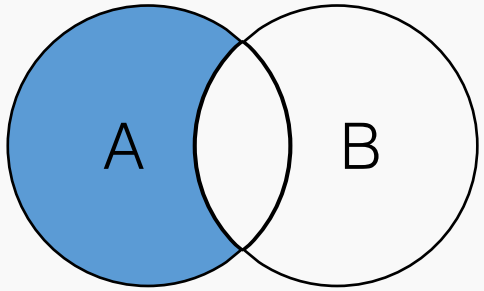
결과 추출

ON 과 WHERE 차이



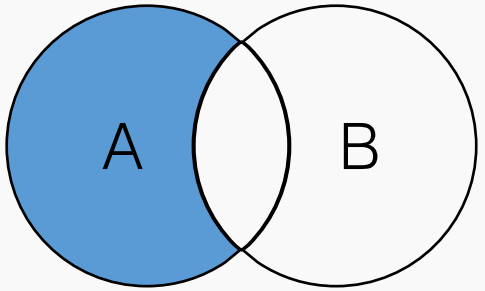

```
SELECT A.
사원번호
      ,A.사원명
      ,B.부서명,
      ,B.지역,
FROM 사원 A
LEFT OUTER
JOIN 부서 B
ON (A.부서코드 =
B.부서코드)
```

```
SELECT A.사원번호
      ,A.사원명
      ,B.부서명,
      ,B.지역,
FROM 사원 A
LEFT OUTER JOIN 부서 B
ON(A.부서코드 = B.
부서코드
    AND B.지역 = '수원')
```



LEFT OUTER
JOIN
WHERE B.KEY IS
NULL

```
SELECT A.사원번호
      ,A.사원명
      ,B.부서명,
      ,B.지역,
FROM 사원 A
LEFT OUTER JOIN 부서 B
ON (A.부서코드 = B.
부서코드)
WHERE
B.지역 = '수원'
```



LEFT OUTER
JOIN
WHERE B.KEY IS

행렬변환 결과

01 기준행렬

	직급	사원명
1	대리	전수경
2	대리	조동진
3	대리	한영애
4	차장	김창완
5	차장	오석준

02 변환행렬

	직급	사원1	사원2	사원3	사원4	사원5
1	대리	전수경	조동진	한영애	(null)	(null)
2	차장	김창완	오석준	(null)	(null)	(null)

LOT Table을 이용한 행렬변환

01 기준 행렬

```
SELECT USRDEF.USER_DESC  사원명
        ,LOTDEF.LOT_ID LOT
        ,ROW_NUMBER() OVER (PARTITION BY USRDEF.USER_ID
         ORDER BY LOTDEF.LOT_ID) LOT순번
   FROM FAB3_USRDEF USRDEF
   INNER JOIN FAB3_LOTDEF LOTDEF
   ON (USRDEF.USER_ID = LOTDEF.CREATE_USER_ID)
   WHERE USRDEF.USER_GRP_2 = '조립 사업부'
;
```

한 작업자가 최대 5개의 LOT을 생성

현재까지 생성된 LOT을 작업자별로 모니터링하는 기능

LOT Table을 이용한 행렬변환

02 데이터 입력

LOT_ID	LOT_DESC
LOT_130281A_220427_01	실내조명용 LED 모듈을 위한 전원 장치 / 130281A
LOT_60101U_220427_01	실내조명용 LED 모듈을 위한 전원 장치 / 60101U
LOT_80101A_220427_01	실외조명용 LED 모듈을 위한 전원 장치 / 80101A
LOT_3718_220427_01	Capacitor는 전하를 충/방전하고 Noise를 제거하는 Capacitor / PTC3718
LOT_3226_220427_01	Capacitor는 전하를 충/방전하고 Noise를 제거하는 Capacitor / PTC3226
LOT_ENHASU80_220427_01	감성전자 은하수 80 LTE version
LOT_ENHASU80_PRO_5G_220427_01	감성전자 은하수 80 PRO 5G-A

LOT Table을 이용한 행렬변환

03 변환 행렬

```
SELECT 사원명
     ,MAX(CASE WHEN LOT순번 = 1 THEN LOT END) 담당_LOT_01
     ,MAX(CASE WHEN LOT순번 = 2 THEN LOT END) 담당_LOT_02
     ,MAX(CASE WHEN LOT순번 = 3 THEN LOT END) 담당_LOT_03
     ,MAX(CASE WHEN LOT순번 = 4 THEN LOT END) 담당_LOT_04
     ,MAX(CASE WHEN LOT순번 = 5 THEN LOT END) 담당_LOT_05
FROM (
    SELECT USRDEF.USER_DESC  사원명
         ,LOTDEF.LOT_ID LOT
         ,ROW_NUMBER() OVER (PARTITION BY USRDEF.USER_ID ORDER BY LOTDEF.LOT_ID)
LOT순번
    FROM FAB3_USRDEF USRDEF
    INNER JOIN FAB3_LOTDEF LOTDEF
    ON (USRDEF.USER_ID = LOTDEF.CREATE_USER_ID)
    WHERE USRDEF.USER_GRP_2 = '조립 사업부')
GROUP BY 사원명
ORDER BY 사원명;
```

LOT Table을 이용한 행렬변환 결과

01 기준행렬

사원명	LOT	LOT순번
정상택	LOT_30281A_220427_01	1
홍길동	LOT_ENHASU80_220427_01	2
정상택	LOT_30281A_220427_01	2
정상택	LOT_ENHASU80_220427_01	3

02 변환행렬

사원명	담당_LOT_01	담당_LOT_02	담당_LOT_03
정상택	LOT_30281A_220427_01	LOT_30281A_220427_01	LOT_ENHASU80_220427_01
홍길동		LOT_ENHASU80_220427_01	

3. SUB QUERY

의미

- 하나의 SQL문 안에 포함되어 있는 또 다른 SQL문
  집합관계에 영향을 주지 않으면서 조건을 이용할 때 사용

INLINE VIEW

- FROM 절에 기술하여 테이블처럼 사용

NESTED SUB QUERY

- WHERE 절에 기술하여 메인쿼리의 조건으로 사용

SCALAR SUB QUERY EXPRESSION

- 1 ROW, 1 COLUMN을 반환하는 목적으로 사용
  주로 SELECT, WHERE 절에 사용

SUB QUERY 종류

01 INLINE VIEW

```
SELECT A.사원번호
     ,A.사원명
FROM 사원 A
    ,(
     SELECT DISTINCT
          B.사원번호
     FROM 급여지급 B
     WHERE B.급여월 BETWEEN
     '201101' AND '201112'
     ) B
WHERE A.부서코드 = '30'
AND B.사원번호 = A.사원번호
;
```

02 NESTED SUB QUERY

```
SELECT A.사원번호
     ,A.사원명
FROM 사원 A
WHERE A.부서코드 = '30'
AND A.사원번호 IN
(
SELECT B.사원번호
FROM 급여지급 B
WHERE B.급여월
BETWEEN '201101'
AND '201112'
)
;
```

03 SCALAR SUB QUERY

```
SELECT A.사원번호
     ,A.사원명
     ,NVL(
      (
      SELECT B.기본수당액
      FROM 기타급여 B
      WHERE B.사원번호 = A.사원번호
      ), 0)          기본수당액
FROM 사원 A
;
```

SCALAR SUB QUERY

01 SQL

```
SELECT A.사원번호
      ,A.사원명
      ,(
       SELECT B.기본수당액 + B.특별수당액
       FROM 기타급여 B
       WHERE B.사원번호 = A.사원번호
       )              지급예정수당
      ,(
       SELECT MAX(B.수당)
       FROM 급여지급 B
       WHERE B.사원번호 = A.사원번호
       )              지급수당
FROM 사원 A
WHERE A.부서코드 = '30'
;
```

02 가이드

1 ROW, 1 COLUMN로 값이 추출되는 원칙

01427. 00000 - "single-row subquery returns more than one row

	사원번호	사원명	지급예정수당	지급수당
1	101	조동진	(null)	100000
2	102	한영애	220000	220000
3	107	오석준	220000	220000
4	109	전수경	(null)	120000
5	111	김황완	(null)	(null)

SCALAR SUB QUERY

01 SQL

```
SELECT A.사원번호
      ,A.사원명
      ,NVL((
       SELECT B.기본수당액 + B.특별수당액
       FROM 기타급여 B
       WHERE B.사원번호 = A.사원번호
       ), 0)              지급예정수당
      ,NVL(
       (SELECT MAX(B.수당)
       FROM 급여지급 B
       WHERE B.사원번호 = A.사원번호
       ), 0)              지급수당
FROM 사원 A
WHERE A.부서코드 = '30'
;
```

02 가이드

NULL 처리에 유의할 것

	사원번호	사원명	지급예정수당	지급수당
1	101	조동진	0	100000
2	102	한영애	220000	220000
3	107	오석준	220000	220000
4	109	전수경	0	120000
5	111	김창완	0	0

SCALAR SUB QUERY

01 SQL

```
SELECT C.사원명, C.지급예정수당, C.지급수당
FROM (    SELECT A.사원명
         ,(SELECT B.기본수당액 + B.특별수당액
          FROM 기타급여 B
          WHERE B.사원번호 = A.사원번호
          )              지급예정수당
         ,(SELECT MAX(B.수당)
          FROM 급여지급 B
          WHERE B.사원번호 = A.사원번호
         )              지급수당
    FROM 사원 A
    WHERE A.부서코드 = '30' ) C
WHERE C.지급예정수당 > -10000
;
```

02 가이드

필터 조건 범위가 마이너스가 될 경우,
NULL 처리가 안 되어 있는 경우에는
검색 결과가 달라질 수 있음

	사원명	지급예정수당	지급수당
1	한영애	220000	220000
2	오석준	220000	220000

SCALAR SUB QUERY

01 SQL

```
SELECT C.사원명, C.지급예정수당, C.지급수당
FROM (    SELECT A.사원명
          ,NVL((
           SELECT B.기본수당액 + B.특별수당액
           FROM 기타급여 B
           WHERE B.사원번호 = A.사원번호), 0
           )                지급예정수당
          ,NVL(
           (SELECT MAX(B.수당)
           FROM 급여지급 B
           WHERE B.사원번호 = A.사원번호), 0
            )                지급수당
    FROM 사원 A
    WHERE A.부서코드 = '30' ) C
WHERE C.지급예정수당 > -10000
;
```

02 가이드

NULL을 입력 값으로 받으면, 에러 처리가 복잡해지는 경우가 발생

	사원명	지급예정수당	지급수당
1	조동진	0	100000
2	한영애	220000	220000
3	오석준	220000	220000
4	전수경	0	120000
5	김창완	0	0

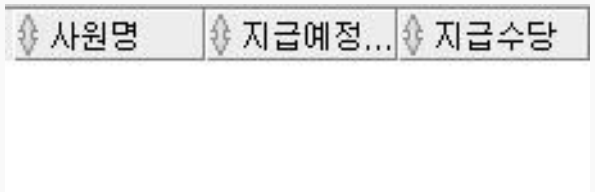
SCALAR SUB QUERY

01 SQL

```
SELECT C.사원명, C.지급예정수당, C.지급수당
FROM (    SELECT A.사원명
         ,(SELECT B.기본수당액 + B.특별수당액
           FROM 기타급여 B
           WHERE B.사원번호 = A.사원번호
           AND B.사원번호 < 0) 지급예정수당
         ,(SELECT MAX(B.수당)
           FROM 급여지급 B
           WHERE B.사원번호 = A.사원번호
           AND B.사원번호 < 0) 지급수당
    FROM 사원 A
    WHERE A.부서코드 = '30' ) C
WHERE C.지급예정수당 > -10000
;
```

02 가이드

스칼라 서브 쿼리를 사용한 메인쿼리의 결과 값이 NULL이 되는 문제 발생
- 필터 조건 사용



SCALAR SUB QUERY

01 SQL

```
SELECT C.사원명, C.지급예정수당, C.지급수당
FROM (    SELECT A.사원명
          ,NVL((
           SELECT B.기본수당액 + B.특별수당액
           FROM 기타급여 B
           WHERE B.사원번호 = A.사원번호
           AND B.사원번호 < 0 ), 0) 지급예정수당
          ,NVL(
           (SELECT MAX(B.수당)
           FROM 급여지급 B
           WHERE B.사원번호 = A.사원번호
           AND B.사원번호 < 0 ), 0) 지급수당
     FROM 사원 A
     WHERE A.부서코드 = '30' ) C
WHERE C.지급예정수당 > -10000
;
```

02 가이드

스칼라 서브 쿼리 밖에서 NVL로 NULL처리

	사원명	지급예정수당	지급수당
1	조동진	0	0
2	한영애	0	0
3	오석준	0	0
4	전수경	0	0
5	김창완	0	0

4. NULL 처리

의미

- NULL은 3가지 의미를 가짐
  - 아직 정해지지 않은 값
  - 확인되지 않는 값
  - 적용할 수 있는 값이 없음

IS NULL / IS NOT NULL

- NULL 여부를 판단하여 참 / 거짓을 반환하는 함수

COUNT(*)

- NULL 값도 집계하는 함수

NVL 함수

- NULL 값이 있는 경우, 다른 값으로 대체하는 함수

IS NULL / IS NOT NULL

01 SQL

```
SELECT 사원번호, 사원명, 퇴사일자
FROM 사원
WHERE (
    퇴사일자 <>
    TO_DATE('20101201', 'YYYYMMDD')
    OR 퇴사일자 IS NULL
    )
;

SELECT 사원번호, 사원명, 퇴사일자
FROM 사원
WHERE (
    퇴사일자 <>
    TO_DATE('20101201', 'YYYYMMDD')
    OR 퇴사일자 = NULL
    )
;
```

02 NULL을 제외하는 연산자

=

	사원번호	사원명	퇴사일자
1	110	이선희	07/12/01

03 NULL을 비교하는 연산자

IS NULL
IS NOT NULL

	사원번호	사원명	퇴사일자
1	101	조동건	(null)
2	103	조규찬	(null)
3	104	이상은	(null)
4	105	조덕배	(null)
5	106	장필순	(null)
6	107	오석준	(null)
7	108	이규석	(null)
8	109	전수경	(null)
9	110	이선희	07/12/01
10	111	김창완	(null)
11	112	이현우	(null)

집계함수의 NULL 처리

01 SQL

```
SELECT COUNT(*)
     ,COUNT(금액)
     ,SUM(금액)
     ,AVG(금액)
FROM (
    SELECT 10   금액 FROM DUAL UNION ALL
    SELECT NULL 금액 FROM DUAL UNION
ALL
    SELECT 30   금액 FROM DUAL
    ) A
;
```

02 NULL 포함

COUNT(*)

03 NULL 비포함

COUNT(금액)
SUM(금액)
AVG(금액)

	COUNT(*)	COUNT(금액)	SUM(금액)	AVG(금액)
1	3	2	40	20

NVL함수로 NULL 처리

01 NULL 처리를 하지 않음

```
SELECT 사원번호
     ,사원명
     ,퇴사일자
FROM 사원
WHERE 부서코드 = '30'
;
```

	사원번호	사원명	퇴사일자
1	101	조동진	(null)
2	102	한영애	10/12/01
3	107	오석준	(null)
4	109	전수경	(null)
5	111	김창완	(null)

02 NULL 처리

```
SELECT 사원번호
     ,사원명
     ,NVL(퇴사일자, TO_DATE('99991231', 'YYYYMMDD'))  퇴사일자
FROM 사원
WHERE 부서코드 = '30‘
;
```

	사...	사원명	퇴사일자
1	101	조동진	99/12/31
2	102	한영애	10/12/01
3	107	오석준	99/12/31
4	109	전수경	99/12/31
5	111	김창완	99/12/31

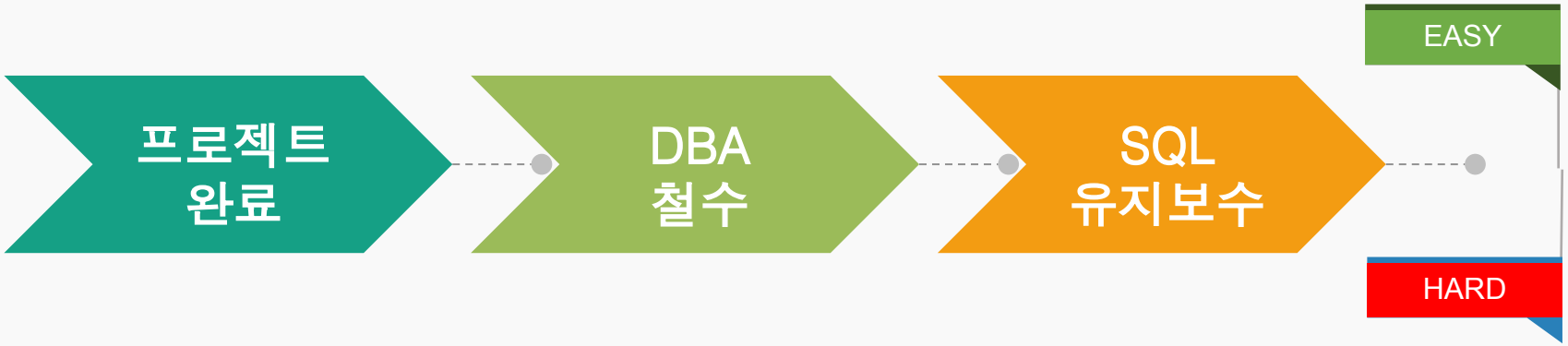
SCALAR SUB QUERY의 NULL 처리

01 SQL

```
SELECT A.사원번호
      ,A.사원명
      ,NVL(
       (
       SELECT B.기본수당액
       FROM 기타급여 B
       WHERE B.사원번호 = A.사원번호
       ), 0)              기본수당액
FROM 사원 A
;
```

	사원번호	사원명	기본수당액
1	101	조동진	100000
2	102	한영애	100000
3	103	조규찬	100000
4	104	이상은	100000
5	105	조덕배	100000
6	106	장필순	100000
7	107	오석준	100000
8	108	이규석	100000
9	109	전수경	0
10	110	이선희	0
11	111	김창완	0
12	112	이현우	100000

5. SQL 유지보수



동일한 결과를 얻을 수 있는 SQL은 많으나, 최적의 SQL은 얻기 힘들다.

생각없이 만든 SQL은 재앙으로 다가온다.

유지보수가 쉬울려면

ANSI SQL Global

괄호 사용

규칙 통일

내려쓰기

들여쓰기

별칭 사용

대소문자 통일

주석 사용

확장성

사례

01 별칭(ALIAS)을 잘 사용한 예

```
SELECT A.사원번호  사원번호
    ,A.사원명          사원명
    ,A.부서코드        부서코드
    ,(
        SELECT D.부서명
        FROM 부서 D
        WHERE D.부서코드 = A.부서코드
    )                  부서명
    ,B.기본급급여차이
    ,B.수당            수당
FROM 사원    A
        ,급여지급 B
WHERE A.사원번호 = B.사원번호
```

02 ANSI SQL 사용

```
SELECT A.사원번호  사원번호
    ,A.사원명          사원명
    ,A.부서코드        부서코드
    ,(
        SELECT D.부서명
        FROM 부서 D
        WHERE D.부서코드 = A.부서코드
    )                  부서명
    ,B.기본급급여차이
    ,B.수당            수당
FROM 사원    A
INNER JOIN 급여지급 B ON
(A.사원번호 = B.사원번호)
```

VARCHAR2 VS CHAR

개발 Data Type의 사례

- 한 업체에서는 Flag를 제외한 모든 문자열을 VARCHAR2로 사용한다.

CHAR를 쓰는 경우, 얻는 성능 이득보다 관리 비용이 더 많이 발생

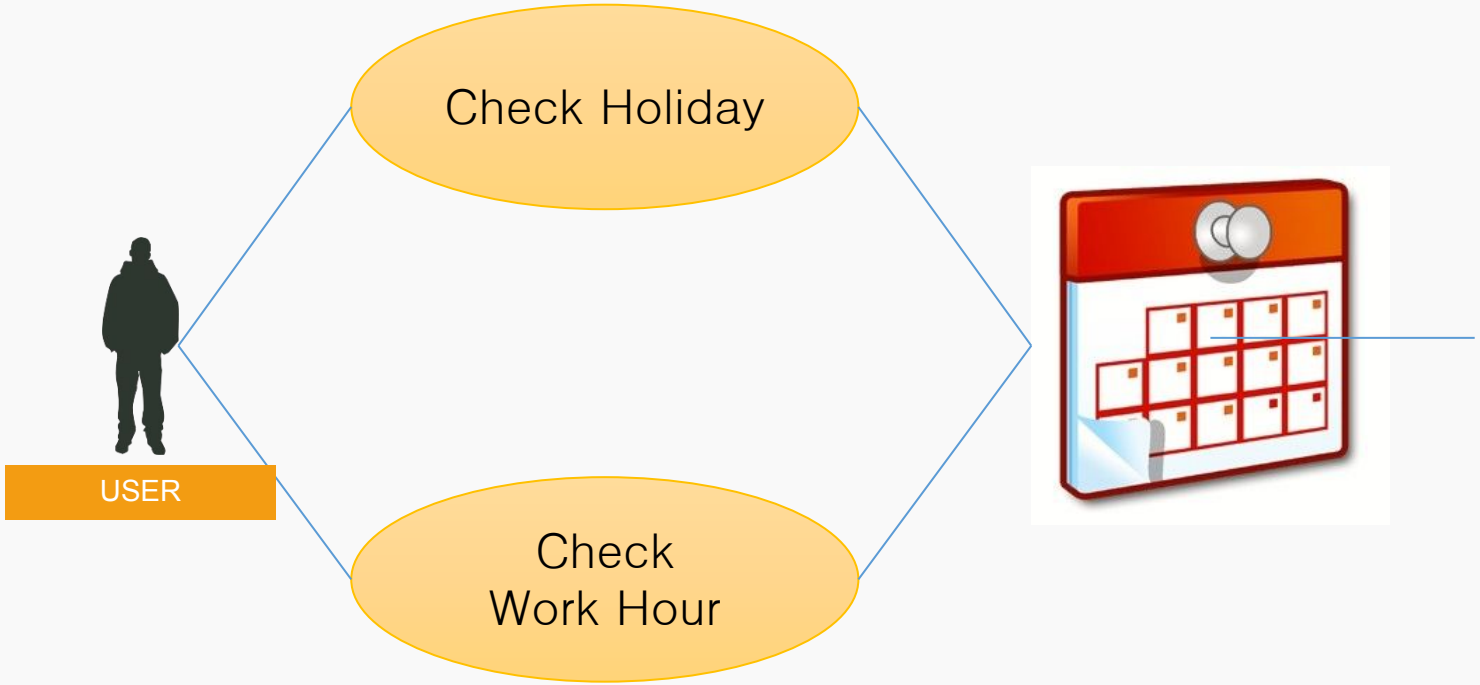
MES, ERP와 같은 대형 시스템은 유지보수 및 관리를 더 중시 여긴다.

관리의 복잡도를 줄이는 이점이 있어 문자열은 VARCHAR2로 통일

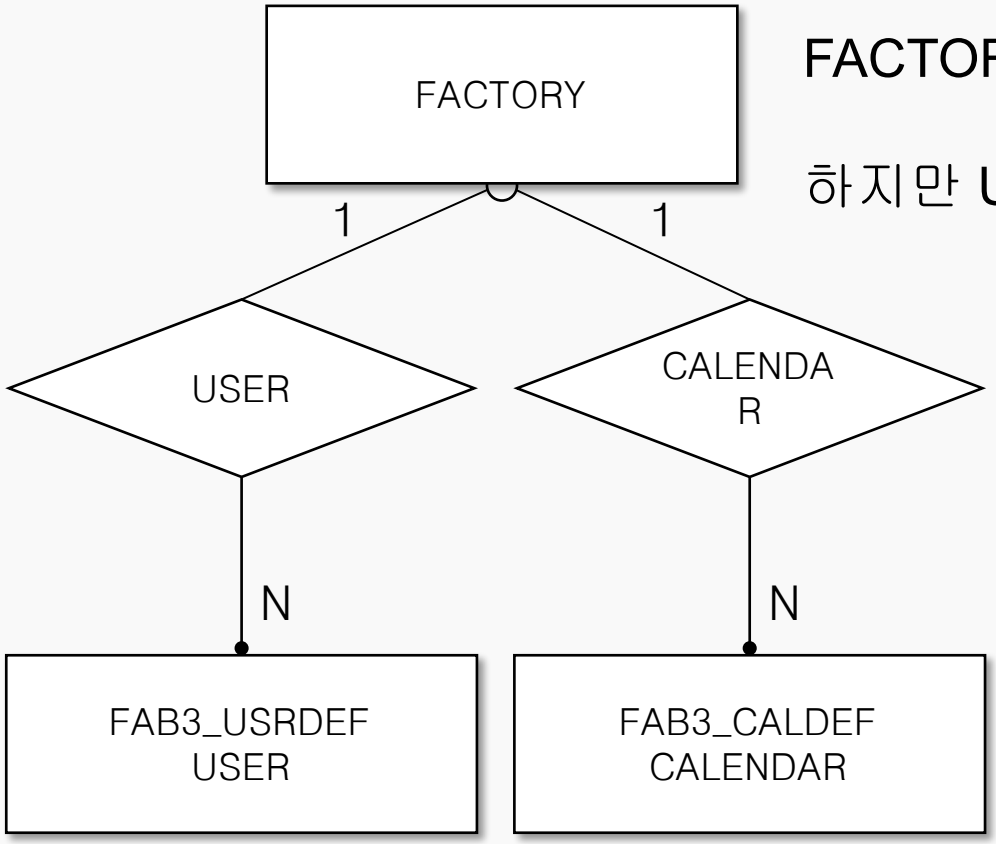
- 대부분은 관리의 비용을 줄이는 쪽이 더 이득이다.
- DB는 여러 소프트웨어와 연결되어 있기 때문에, 개발과 유지보수의 편의성이 중요하다.
- 어느정도 개발이 되어야 다음에 성능 및 속도를 위한 튜닝에 들어갈 수 있다.

6. 실습

01 요구사항 분석



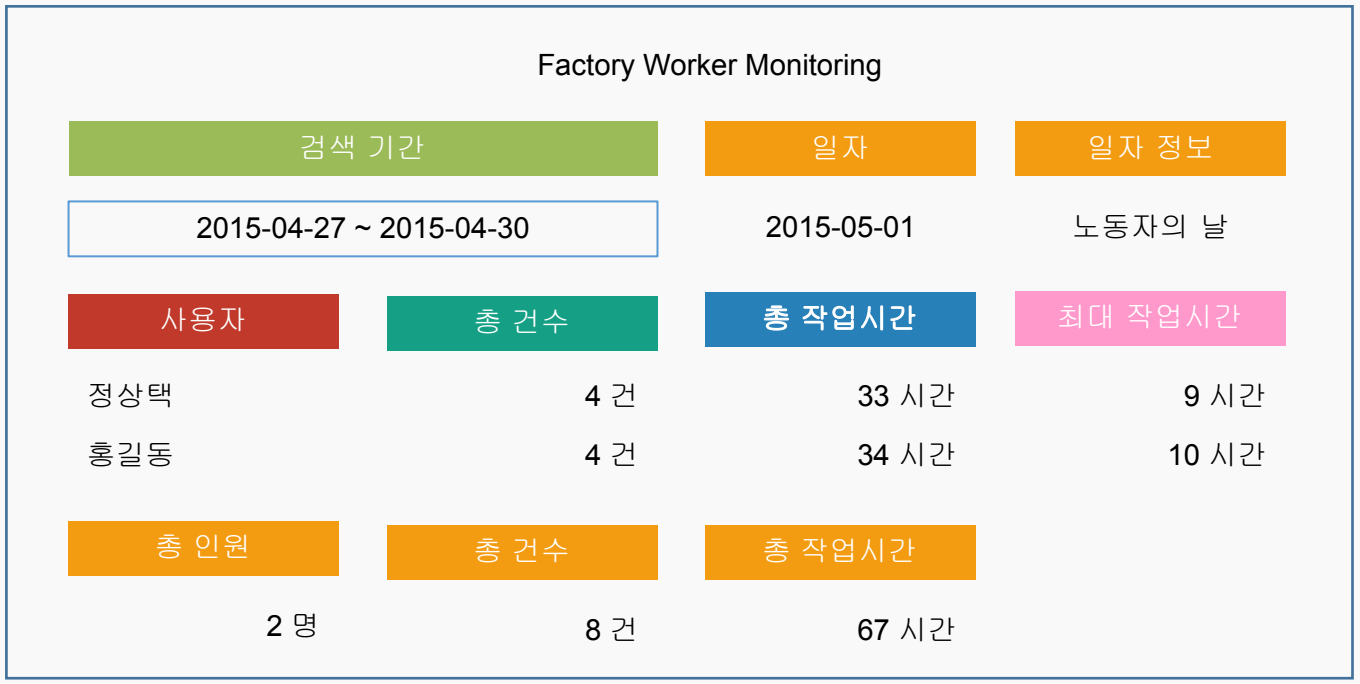
02 관계 분석



FACTORY는 USER와 CALENDAR을 소유 할 수 있고, 그렇지 않을 수 있

하지만 USER와 CALENDAR은 반드시 FACTORY에 속해 있어야 한다.

03 UI 확인



TEXT BOX 입력

TITLE LABEL 별칭

DATA LABEL 데이터 출력

04 Data Structure 분석

```
struct FAB3_CALDEF_TAG
{
    char        CALENDAR_ID[20];
    int         SYS_YEAR;
    int         SYS_MONTH;
    int         SYS_DAY;
    char        SYS_DATE[8];
    int         WORK_HOUR;
    char        WORK_HOUR_FLAG;
    char        HOLIDAY_FLAG;
    char        HOLIDAY_DESC[200];
    char        CREATE_USER_ID[20];
    char        CREATE_TIME[14];
    char        UPDATE_USER_ID[20];
    char        UPDATE_TIME[14];
};
```

05 테이블로 변환

```
CREATE TABLE FAB3_CALDEF
(
    CALENDAR_ID          VARCHAR2(20)        DEFAULT(' ')     NOT NULL,
    SYS_YEAR             NUMBER(6)           DEFAULT(0) NOT NULL,
    SYS_MONTH            NUMBER(6)           DEFAULT(0) NOT NULL,
    SYS_DAY              NUMBER(6)           DEFAULT(0) NOT NULL,
    SYS_DATE             VARCHAR2(8)         DEFAULT(' ')     NOT NULL,
    WORK_HOUR            NUMBER(6)           DEFAULT(0) NOT NULL,
    HOLIDAY_FLAG         CHAR(1)         DEFAULT(' ')     NOT NULL,
    HOLIDAY_DESC         VARCHAR2(200)   DEFAULT(' ')     NOT NULL,
    CREATE_USER_ID       VARCHAR2(20)        DEFAULT(' ')     NOT NULL,
    CREATE_TIME          VARCHAR2(14)        DEFAULT(' ')     NOT NULL,
    UPDATE_USER_ID       VARCHAR2(20)        DEFAULT(' ')     NOT NULL,
    UPDATE_TIME          VARCHAR2(14)     DEFAULT(' ')     NOT NULL
) TABLESPACE FAB3;
```

06 데이터 입력

CALANDAR_ID	SYS_YEAR	SYS_MONTH	SYS_DAY	SYS_DATE	WORK_HOUR	HOLIDAY_FLAG	HOLIDAY_DESC	CREATE_USER
TAIK_150427_01	2015	4	27	20150427	8	N		taik
TAIK_150428_01	2015	4	28	20150428	8	N		taik
TAIK_150429_01	2015	4	29	20150429	8	N		taik
TAIK_150430_01	2015	4	30	20150430	9	N		taik
TAIK_150501_01	2015	5	1	20150501	0	Y	노동자의 날	taik
HONG150427_01	2015	4	27	20150427	8	N		hong
HONG_150428_01	2015	4	28	20150428	8	N		hong
HONG_150429_01	2015	4	29	20150429	8	N		hong
HONG_150430_01	2015	4	30	20150430	10	N		hong
HONG_150501_01	2015	5	1	20150501	0	Y	노동자의 날	hong

07 사용자 정보 SQL

```
SELECT USRDEF.USER_ID
      ,USRDEF.USER_DESC
      ,COUNT(USRDEF.USER_ID)
      ,SUM(CALDEF.WORK_HOUR)
      ,MAX(CALDEF.WORK_HOUR)
FROM FAB3_USRDEF USRDEF
INNER JOIN FAB3_CALDEF CALDEF ON
(
      USRDEF.USER_ID = CALDEF.CREATE_USER_ID
      AND (   CALDEF.SYS_DATE >= '20150427'
            AND CALDEF.SYS_DATE <= '20150430'
      )
)
GROUP BY USRDEF.USER_ID, USRDEF.USER_DESC
ORDER BY USRDEF.USER_ID DESC
;
```

사용자	총 건수	총 작업시간	최대 작업시간
정상택	4 건	33 시간	9
홍길동	4 건	34 시간	10

08 합계 SQL

```
SELECT COUNT(DISTINCT CALDEF.CREATE_USER_ID)
       ,COUNT(*)
       ,SUM(CALDEF.WORK_HOUR)
FROM FAB3_USRDEF USRDEF
INNER JOIN FAB3_CALDEF CALDEF ON
(
       USRDEF.USER_ID = CALDEF.CREATE_USER_ID
       AND (   CALDEF.SYS_DATE >= '20150427'
              AND CALDEF.SYS_DATE <= '20150430'
       )
)
;
```

총 인원	총 건수	총 작업시간
2 명	8 건	67 시간

Q&A

감사합니다.

https://github.com/sangtaik