

[Database] ORACLE & ANSI표준의 JOIN 함수 사용법

노트북: [TIL-MY]

만든 날짜: 2020-07-16 오전 9:02

URL: <https://continuous-development.tistory.com/23>

나무늘보의 개발 블로그



Database

[Database] ORACLE & ANSI표준의 JOIN 함수 사용법

| 꾸까꾸 | 2020. 7. 16. 08:57 | 수정 | 삭제

JOIN (INNER JOIN)

INNER JOIN은 테이블간에 매칭되는 것들로만 묶어서 검색한다. (교집합만 가져온다고 보면 된다.)

#일단 오라클 기준과 ANSI(SQL 공통 표준)를 나눠서 설명하겠습니다.

JOIN - ORACLE

각기 다른 두테이블을 하나로 묶어서 출력을 해주는 함수

equals join - 업무적인 연관성있는 컬럼 간에 조인 (이건 부모키가 외래키랑 연관이 있는 관계)

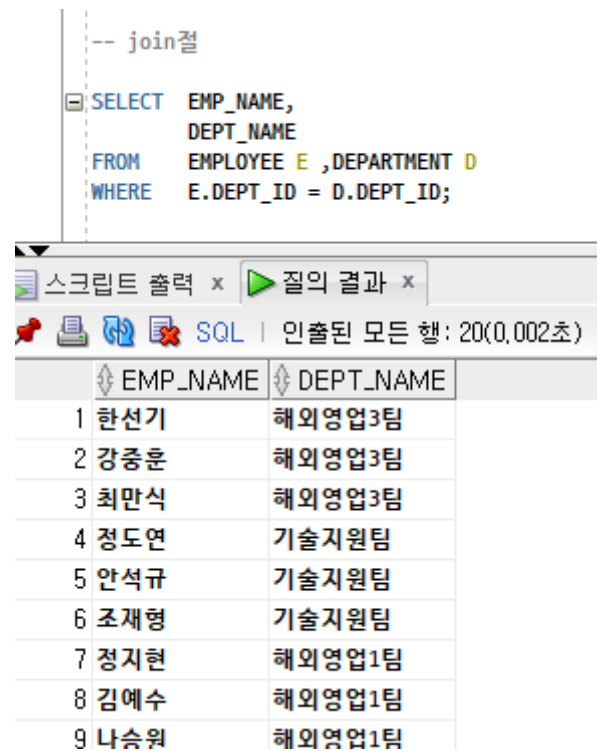
non equals join- 업무적인 연관성이 없는 조인(두 테이블 사이에 연관성이 없는 경우)

Oracle 기준

SELECT

FROM 컬럼 / 별칭 , 컬럼 / 별칭

WHERE 이 부분에서 두 컬럼사이의 연관성이 있는 걸 equals 로 묶는다.



The screenshot shows a SQL Developer window with a query editor and a results pane. The query is a join between the EMPLOYEE and DEPARTMENT tables. The results pane shows 20 rows of data, with the first 9 rows visible in the table below.

```
-- join결
SELECT EMP_NAME,
       DEPT_NAME
FROM   EMPLOYEE E ,DEPARTMENT D
WHERE  E.DEPT_ID = D.DEPT_ID;
```

	EMP_NAME	DEPT_NAME
1	한선기	해외영업3팀
2	강중훈	해외영업3팀
3	최만식	해외영업3팀
4	정도연	기술지원팀
5	안석규	기술지원팀
6	조재형	기술지원팀
7	정지현	해외영업1팀
8	김예수	해외영업1팀
9	나승위	해외영업1팀

이런식으로 별칭(EMPLOYEE E, DEPARTMENT D)을 줘야 한다. 그래야 테이블의 모호성이 없다. 여기서 모호성이란 EMPLOYEE에 있는 컬럼인지 DEPARTMENT에 있는 컬럼인지 모르는 상황에서 컬럼을 쓸수 없다. 어떤 테이블에 있는건지 모르기 때문이다.

non equals join join - 업무적인 연관성이 없는 조인

<pre> SELECT * FROM SAL_GRADE; -- job을 조인해서 직급이름을 가져와라잉 </pre>																											
<div> <div>질의 결과 x</div> <div> <div>SQL 인출된 모든 행: 5(0,006초)</div> <table> <tr> <th></th><th>SLEVEL</th><th>LOWEST</th><th>HIGHEST</th></tr> <tr> <td>1</td><td>A</td><td>3000000</td><td>9000000</td></tr> <tr> <td>2</td><td>B</td><td>2500000</td><td>2999999</td></tr> <tr> <td>3</td><td>C</td><td>2000000</td><td>2499999</td></tr> <tr> <td>4</td><td>D</td><td>1500000</td><td>1999999</td></tr> <tr> <td>5</td><td>E</td><td>1000000</td><td>1499999</td></tr> </table> </div> </div>					SLEVEL	LOWEST	HIGHEST	1	A	3000000	9000000	2	B	2500000	2999999	3	C	2000000	2499999	4	D	1500000	1999999	5	E	1000000	1499999
	SLEVEL	LOWEST	HIGHEST																								
1	A	3000000	9000000																								
2	B	2500000	2999999																								
3	C	2000000	2499999																								
4	D	1500000	1999999																								
5	E	1000000	1499999																								

SLEVEL에 따른 연봉 값

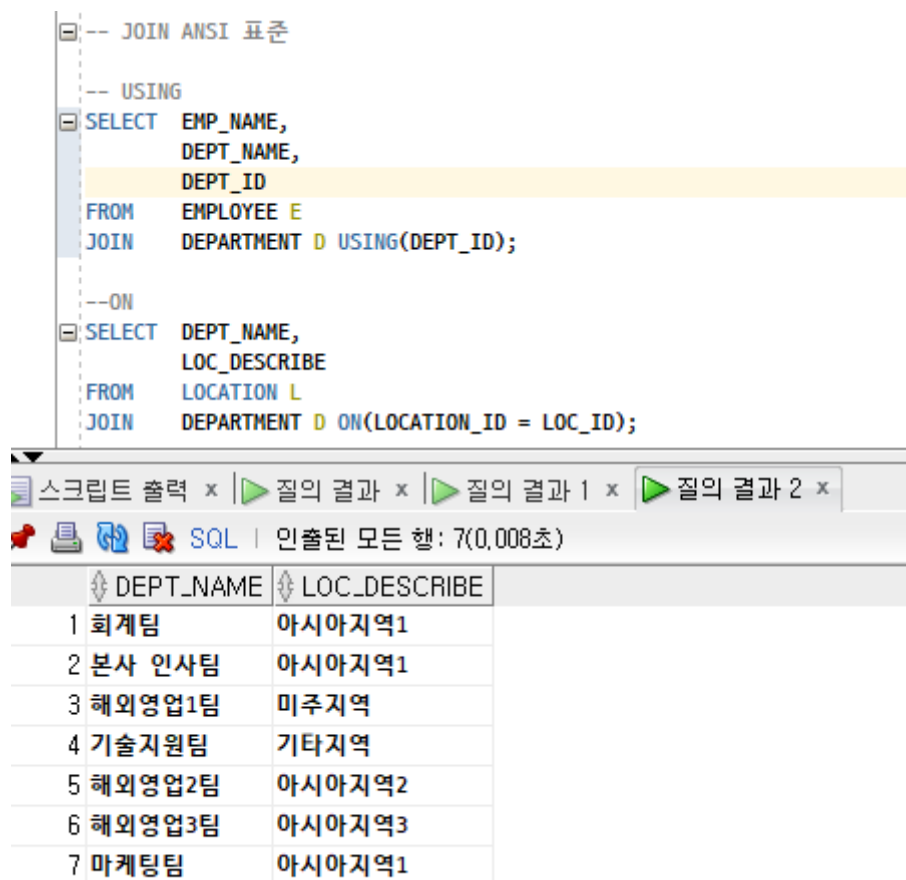
<pre> SELECT EMP_NAME, SALARY, SLEVEL FROM EMPLOYEE, SAL_GRADE WHERE SALARY BETWEEN LOWEST AND HIGHEST ; </pre>																																															
<div> <div>스크립트 출력 x</div> <div> <div>질의 결과 x</div> <div>SQL 인출된 모든 행: 22(0,002초)</div> <table> <tr> <th></th><th>EMP_NAME</th><th>SALARY</th><th>SLEVEL</th></tr> <tr> <td>1</td><td>엄정하</td><td>1500000</td><td>D</td></tr> <tr> <td>2</td><td>고승우</td><td>1500000</td><td>D</td></tr> <tr> <td>3</td><td>정지현</td><td>1500000</td><td>D</td></tr> <tr> <td>4</td><td>성해교</td><td>1900000</td><td>D</td></tr> <tr> <td>5</td><td>전우성</td><td>2090000</td><td>C</td></tr> <tr> <td>6</td><td>김예수</td><td>2100000</td><td>C</td></tr> <tr> <td>7</td><td>심하균</td><td>2300000</td><td>C</td></tr> <tr> <td>8</td><td>나승원</td><td>2300000</td><td>C</td></tr> <tr> <td>9</td><td>엄정하</td><td>2420000</td><td>C</td></tr> <tr> <td>10</td><td>이즈기</td><td>2500000</td><td>B</td></tr> </table> </div> </div>					EMP_NAME	SALARY	SLEVEL	1	엄정하	1500000	D	2	고승우	1500000	D	3	정지현	1500000	D	4	성해교	1900000	D	5	전우성	2090000	C	6	김예수	2100000	C	7	심하균	2300000	C	8	나승원	2300000	C	9	엄정하	2420000	C	10	이즈기	2500000	B
	EMP_NAME	SALARY	SLEVEL																																												
1	엄정하	1500000	D																																												
2	고승우	1500000	D																																												
3	정지현	1500000	D																																												
4	성해교	1900000	D																																												
5	전우성	2090000	C																																												
6	김예수	2100000	C																																												
7	심하균	2300000	C																																												
8	나승원	2300000	C																																												
9	엄정하	2420000	C																																												
10	이즈기	2500000	B																																												

이 두 테이블 사이에 연관 관계가 없다. 테이블을 붙이기 위해 FROM절에서 테이블을 넣고 조건절로 두 테이블을 이어준다. SALARY는 연봉인데 연봉에 따른 SLEVEL을 붙여줬다.

JOIN -ANSI 표준구문(ANSI 표준은 모든 SQL에서 호환되는 SQL 이다.)

JOIN table2 ON (condition1) - 조건식을 만들수 있다. 관계를 가지고 있지 않은 테이블간의 조인(non equals join)

JOIN table2 USING (column1) - 부모의 기본키를 자식의 외래키를 이용하는게 using 이다. 관계를 가지고 있는 사이에 쓴다.(equals join)



```
-- JOIN ANSI 표준
-- USING
SELECT EMP_NAME,
       DEPT_NAME,
       DEPT_ID
FROM   EMPLOYEE E
JOIN   DEPARTMENT D USING(DEPT_ID);

--ON
SELECT DEPT_NAME,
       LOC_DESCRIBE
FROM   LOCATION L
JOIN   DEPARTMENT D ON(LOCATION_ID = LOC_ID);
```

스크립트 출력 x | 질의 결과 x | 질의 결과 1 x | 질의 결과 2 x

SQL | 인출된 모든 행: 7(0.008초)

	DEPT_NAME	LOC_DESCRIBE
1	회계팀	아시아지역1
2	본사 인사팀	아시아지역1
3	해외영업1팀	미주지역
4	기술지원팀	기타지역
5	해외영업2팀	아시아지역2
6	해외영업3팀	아시아지역3
7	마케팅팀	아시아지역1

아까도 언급했듯이 ANSI 표준에서 USING 과 ON을 쓸수 있는데

USING 은 서로가 부모키와 자식키로 연관이 있을 때 사용하고
ON은 서로 연관관계가 없을때 조건식을 사용해 JOIN 한다.

```
--NATURAL JOIN
SELECT EMP_NAME,
       DEPT_NAME,
       DEPT_ID
FROM   EMPLOYEE E
NATURAL JOIN DEPARTMENT D ;
```

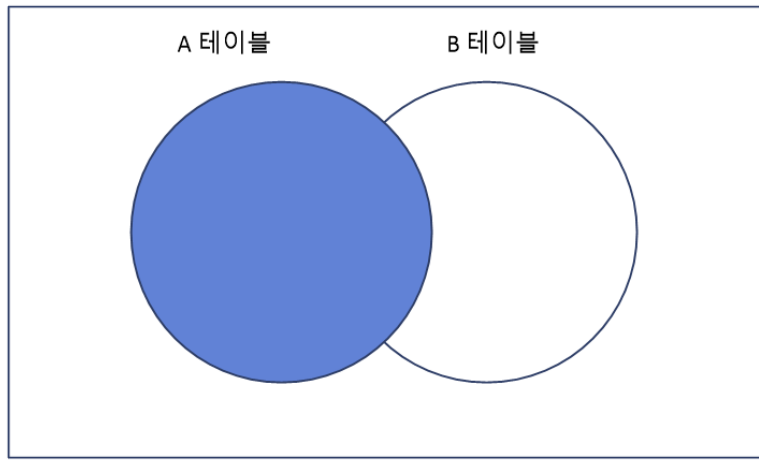
스크립트 출력 x	질의 결과 x	질의 결과 1 x
SQL 인출된 모든 행: 20(0,002초)		
EMP_NAME	DEPT_NAME	DEPT_ID
1 한선기	해외영업3팀	90
2 강중훈	해외영업3팀	90
3 최만식	해외영업3팀	90
4 정도연	기술지원팀	60
5 안석규	기술지원팀	60
6 조재형	기술지원팀	60
7 정지현	해외영업1팀	50
8 김예수	해외영업1팀	50
9 나승원	해외영업1팀	50
10 김순이	해외영업1팀	50
11 성해교	해외영업1팀	50
12 전우성	해외영업2팀	80

이렇게 보면 INNER JOIN은 테이블간에 서로 매칭이 되는 것들로만 엮어서 검색을 하기 때문에 NULL 값이 나오지 않는다. 한 테이블을 기준으로 그 테이블에 맞춰서 뒤에 테이블을 앞 테이블에 붙이는 형태이기 때문이다.

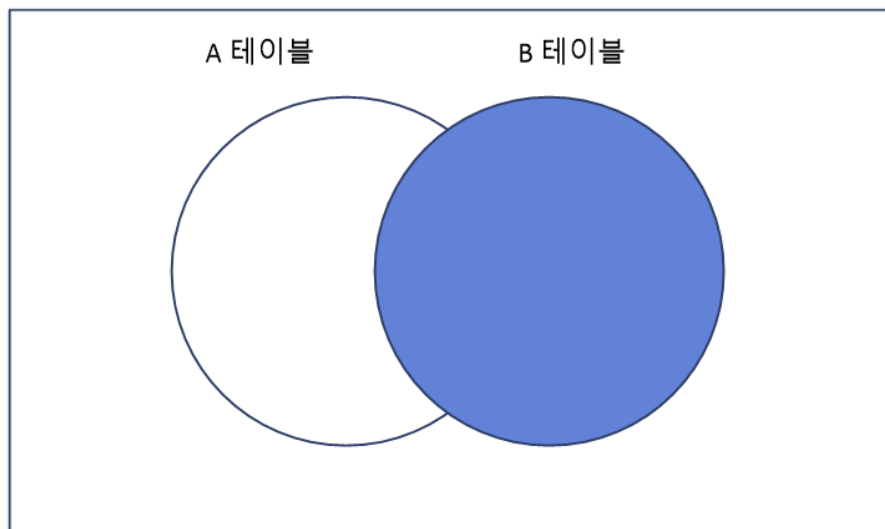
JOIN (OUTER JOIN)

OUTER JOIN은 테이블간에 매칭되는 것 이외에도 옵션에 따라 매칭이 되지 않는 부분들도 엮어서 검색한다.
(합집합이라고 보면 된다.)

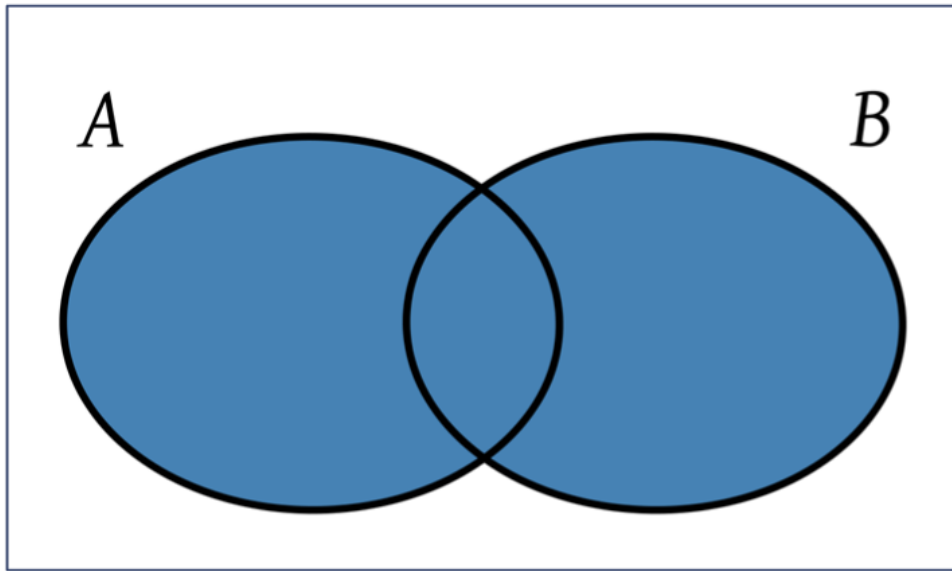
OUTER JOIN은 LEFT / RIGHT / FULL 이 있다. 이걸 방향이라고 생각하면 된다.



LEFT는 왼쪽 방향의 있는 테이블의 기준의 합집합



RIGHT는 오른쪽 방향에 있는 테이블 기준의 합집합



FULL은 두 테이블 기준의 합집합

이런식으로 생각하면 된다.

LEFT | RIGHT | FULL [OUTER]

OUTER는 조건에 만족되지 않는 누락되는 데이터 까지 결과 집합에 포함시킨다.

이 조건이 왼쪽에 있냐 오른쪽에 있냐 양쪽에 있냐를 찾는다.

JOIN - ORACLE

OUTER JOIN - 매칭 되지 않는 데이터도 엮어서 조회한다.

지금 현재는 ORACLE 기준이다.

```
--OUTER JOIN
SELECT EMP_NAME,
       DEPT_NAME,
       D.DEPT_ID
FROM   DEPARTMENT D, EMPLOYEE E
WHERE  D.DEPT_ID(+) = E.DEPT_ID;
```

스크립트 출력 x	질의 결과 x	질의 결과 1 x	▶
SQL 인출된 모든 행: 22(0.005초)			
EMP_NAME	DEPT_NAME	DEPT_ID	
13 조재성	기술지원팀	60	
14 안석규	기술지원팀	60	
15 정도연	기술지원팀	60	
16 엄정하	해외영업2팀	80	
17 전우성	해외영업2팀	80	
18 최만식	해외영업3팀	90	
19 강중훈	해외영업3팀	90	
20 한선기	해외영업3팀	90	
21 엄정하	(null)	(null)	
22 심하균	(null)	(null)	

ORACLE(RIGHT)

<ORACLE 조인 방법>

이렇게 매칭 되지 않는 것도 나온다. + 부분의 맞은편에 있는 애들이 출력된다.

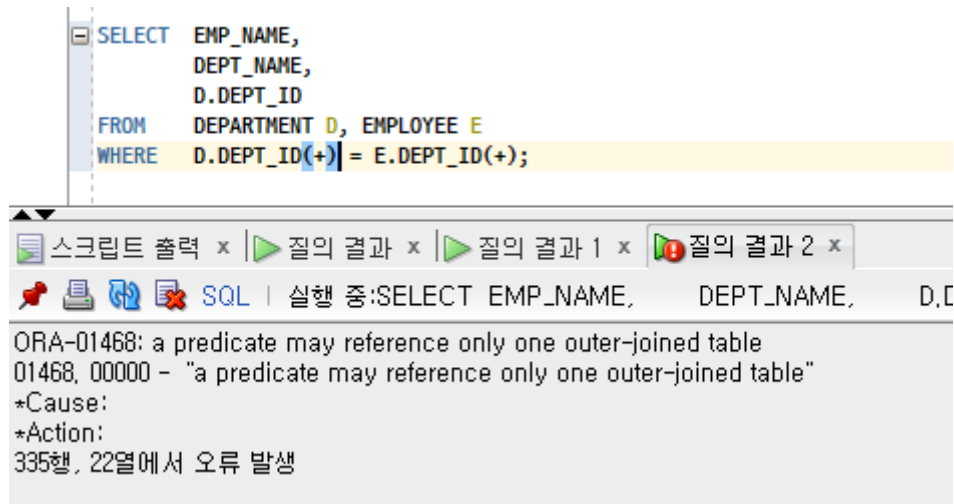
이렇게 WHERE 절에서 두 컬럼사이를 묶어주고 (+)을 통해서 어떤 조인인지 정해 준다.

```
SELECT EMP_NAME,
       DEPT_NAME,
       D.DEPT_ID
FROM   DEPARTMENT D, EMPLOYEE E
WHERE  D.DEPT_ID = E.DEPT_ID(+);
```

스크립트 출력 x	질의 결과 x	질의 결과 1 x	▶	질의 결과 2 x
SQL 인출된 모든 행: 21(0.005초)				
EMP_NAME	DEPT_NAME	DEPT_ID		
13 엄정하	해외영업2팀	80		
14 고승우	본사 인사팀	10		
15 박하일	해외영업1팀	50		
16 권상후	본사 인사팀	10		
17 임영애	본사 인사팀	10		
18 김술오	회계팀	20		
19 이중기	회계팀	20		
20 감우섭	회계팀	20		
21 (null)	마케팅팀	30		

ORACLE (LEFT)

이건 (+)의 위치가 오른쪽에 있기 때문에 LEFT 조인이다.



ORACLE 형태에서 FULL 조인은 안된다.

JOIN - ANSI

<ANSI OUTER 조인 방법>

SELECT

FROM

LEFT | RIGHT | FULL JOIN 테이블 ON|USING (조건)

이런 구문을 가진다.

```

--ANSI RIGHT OUTER JOIN
SELECT EMP_NAME,
       DEPT_NAME
FROM   DEPARTMENT
RIGHT JOIN   EMPLOYEE E USING(DEPT_ID);

--ANSI FULL OUTER JOIN

```

EMP_NAME	DEPT_NAME
13 조계영	기술지원팀
14 안석규	기술지원팀
15 정도연	기술지원팀
16 엄정하	해외영업2팀
17 전우성	해외영업2팀
18 최만식	해외영업3팀
19 강중훈	해외영업3팀
20 한선기	해외영업3팀
21 엄정하	(null)
22 심하균	(null)

ANSI(RIGHT)

ANSI에서는 방향대로 가면 된다. RIGHT 일 경우 오른쪽에 있는 EMPLOYEE 테이블을 기준으로

```

--ANSI LEFT OUTER JOIN
SELECT EMP_NAME,
       DEPT_NAME
FROM   DEPARTMENT
LEFT JOIN   EMPLOYEE E USING(DEPT_ID);

--ANSI RIGHT OUTER JOIN

```

EMP_NAME	DEPT_NAME
13 엄정하	해외영업2팀
14 고승우	본사 인사팀
15 박하일	해외영업1팀
16 권상후	본사 인사팀
17 임영애	본사 인사팀
18 김술오	회계팀
19 이중기	회계팀
20 감우섭	회계팀
21 (null)	마케팅팀

ANSI(LEFT)

LEFT에서는 왼쪽에 있는 테이블 기준이어서 DEPARTMENT 기준으로 한다.

<ANSI 표준 조인>

.

```
--ANSI FULL OUTER JOIN
SELECT EMP_NAME,
       DEPT_NAME
FROM   DEPARTMENT
FULL JOIN EMPLOYEE E USING(DEPT_ID);
```

스크립트 출력 x | 질의 결과 x | 질의 결과 1 x

SQL | 인출된 모든 행: 23(0.002초)

EMP_NAME	DEPT_NAME
14 김이관	(null)
15 고승우	본사 인사팀
16 박하일	해외영업1팀
17 권상후	본사 인사팀
18 임영애	본사 인사팀
19 염정하	(null)
20 김술오	회계팀
21 이중기	회계팀
22 감우섭	회계팀
23 (null)	마케팅팀

ANSI 형태의 조인에서는 FULL 조인도 가능하다.

—

'Database' 카테고리의 다른 글

[Database] ORACLE & ANSI표준의 JOIN 함수 사용법 (0)	08:57:15
[Database]Oracle SQL ORDER BY 절,GROUP BY 절 함수 사용법 (0)	00:01:13
[Database] Oracle SQL DECODE & CASE 함수 사용법 (0)	2020.07.15
[Database]Oracle SQL NVL,NVL2 함수 사용방법 (0)	2020.07.15
[Database] 데이터 타입 변환 TO_DATE,TO_NUMBER,TO_CHAR 사용법 (0)	2020.07.15
[Database] 날짜함수 SYSDATE,ADD_MONTHS,MONTHS_BETWEEN 함수 사용법 (0)	2020.07.14

'Database' Related Articles

[Database]Oracle SQL ORDER BY...	[Database] Oracle SQL...	[Database]Oracle SQL NVL,NVL2 ...	[Database] 데이 터 타입 변환...
-------------------------------------	-----------------------------	--------------------------------------	-----------------------------