Chapter 03. SQLD 이론

# 주요개념 Wrap up 4 – SQL활용

# INNER JOIN

TAB1

| COL1 | COL2 | JKEY |
|---|---|---|
| A | 10 | B |
| B | 20 | C |
| C | 10 | D |
| D | 30 | E |

O ⟷ O
O ⟷ O

TAB2

| JKEY | COL3 | COL4 |
|---|---|---|
| A | 10 | zz |
| B | 20 | yy |
| C | 10 | xx |

2건

| J1 | J2 |
|---|---|
| B | B |
| C | C |

## ON JOIN 조건

```
SELECT  T1.JKEY  J1, T2.JKEY J2
  FROM TAB1 T1 INNER JOIN TAB2 T2
       ON T1. JKEY = T2. JKEY;
```

## WHERE절 JOIN 조건

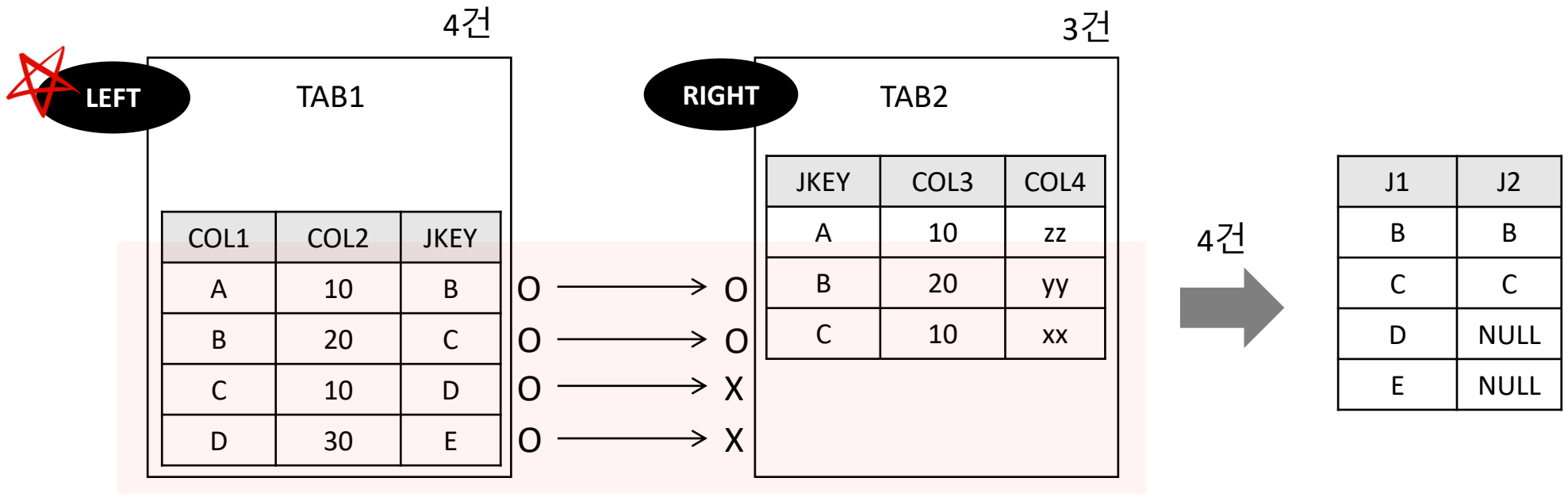
```
SELECT  T1.JKEY J1, T2.JKEY J2
  FROM TAB1 T1 ,TAB2 T2
 WHERE T1. JKEY = T2. JKEY;
```

## USING 조건

```
SELECT  T1.JKEY J1, T2.JKEY J2
  FROM TAB1 T1 JOIN TAB2 T2
  USING (JKEY);
```

** 조인 조건 컬럼명이 다르면 USING은 사용 할 수 없음*

# OUTER JOIN 1 = LEFT OUTER JOIN



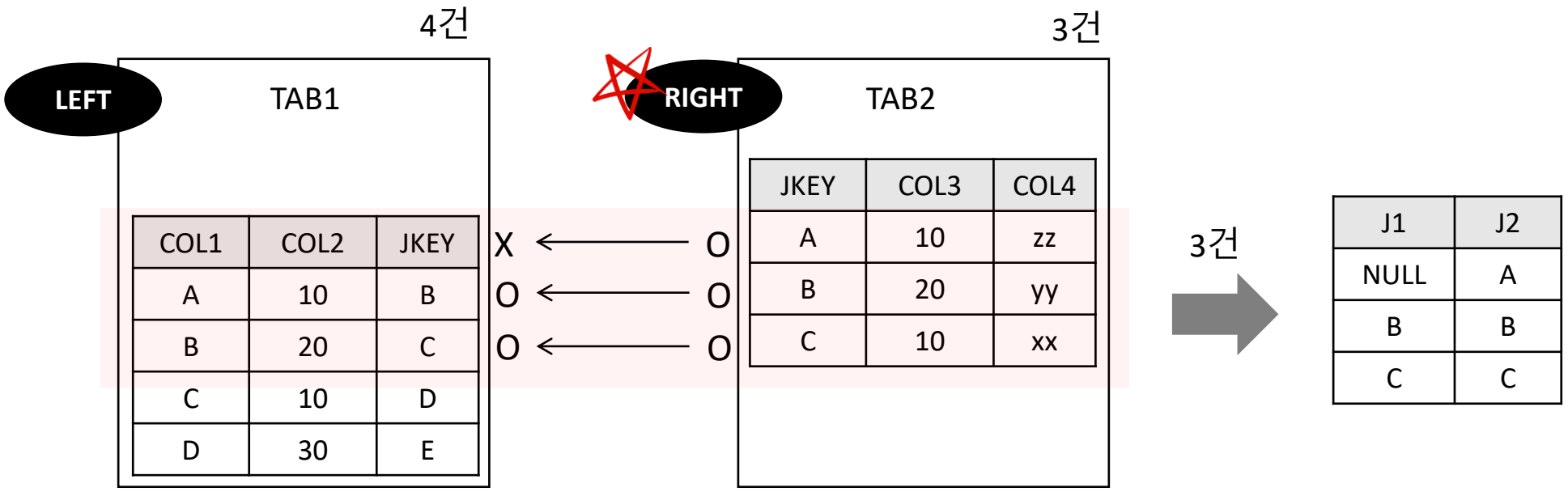

## ANSI

```
SELECT  T1.JKEY  J1, T2.JKEY J2
  FROM TAB1 T1 LEFT OUTER JOIN TAB2 T2
    ON T1. JKEY = T2. JKEY;
```

## ORACLE

```
SELECT  T1.JKEY J1, T2.JKEY J2
  FROM TAB1 T1 ,TAB2 T2
 WHERE T1. JKEY = T2. JKEY(+);
```

# OUTER JOIN 2 = RIGHT OUTER JOIN



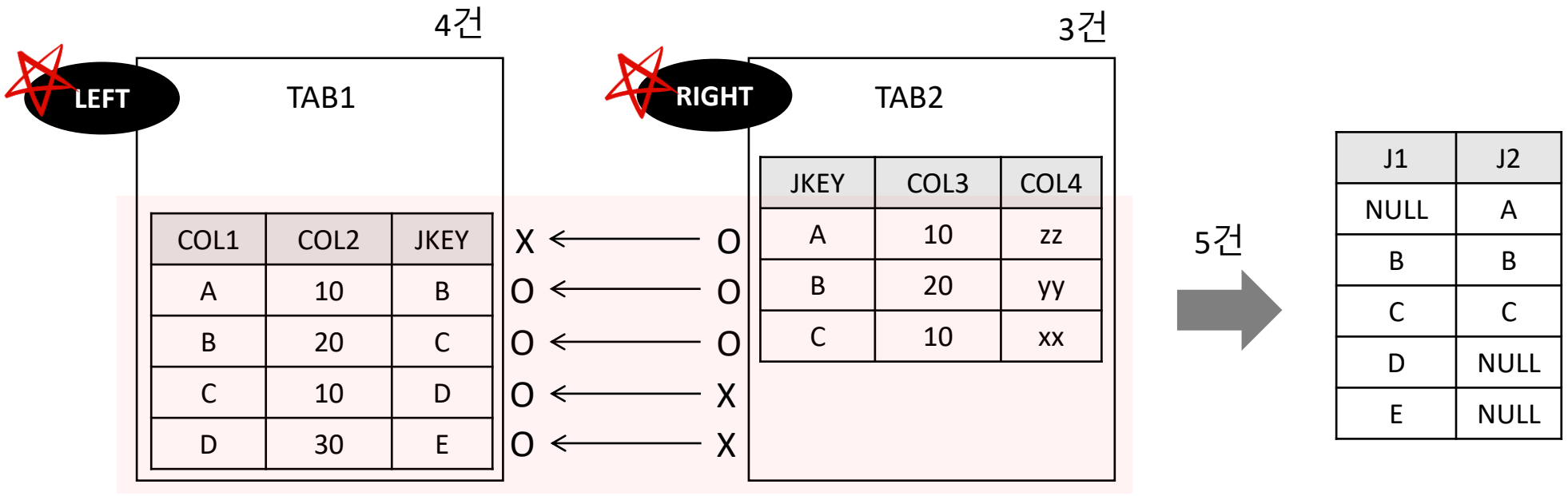

## ANSI

```
SELECT  T1.JKEY  J1, T2.JKEY J2
  FROM TAB1 T1 RIGHT OUTER JOIN TAB2 T2
    ON T1. JKEY = T2. JKEY;
```

## ORACLE

```
SELECT  T1.JKEY J1, T2.JKEY J2
  FROM TAB1 T1 ,TAB2 T2
 WHERE T1. JKEY(+) = T2. JKEY;
```

# OUTER JOIN 3 = FULL OUTER JOIN




## ANSI

```
SELECT  T1.JKEY  J1, T2.JKEY J2
  FROM TAB1 T1 FULL OUTER JOIN TAB2 T2
    ON T1. JKEY = T2. JKEY;
```

# CROSS JOIN = CARTESIAN PRODUCT

4건

TAB1

| COL1 | COL2 | JKEY |
|---|---|---|
| A | 10 | B |
| B | 20 | C |
| C | 10 | D |
| D | 30 | E |

X

3건

TAB2

| JKEY | COL3 | COL4 |
|---|---|---|
| A | 10 | zz |
| B | 20 | yy |
| C | 10 | xx |

12건

| J1 | J2 |
|---|---|
| B | A |
| B | B |
| B | C |
| C | A |
| C | B |
| C | C |
| D | A |
| D | B |
| D | C |
| E | A |
| E | B |
| E | C |

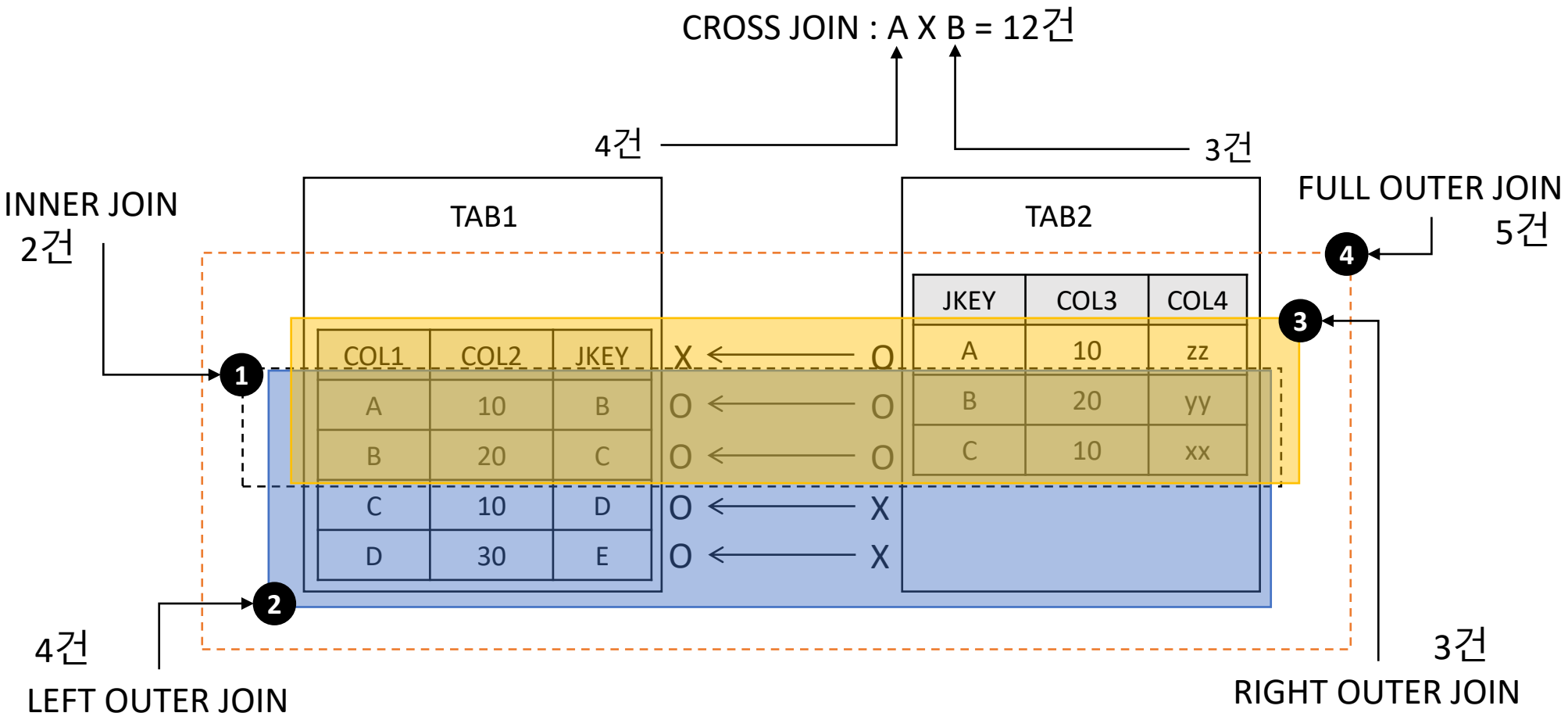
## JOIN절 있음

```
SELECT  T1.JKEY  J1, T2.JKEY J2
 FROM TAB1 T1 CROSS JOIN TAB2 T2
```

## JOIN절 없음

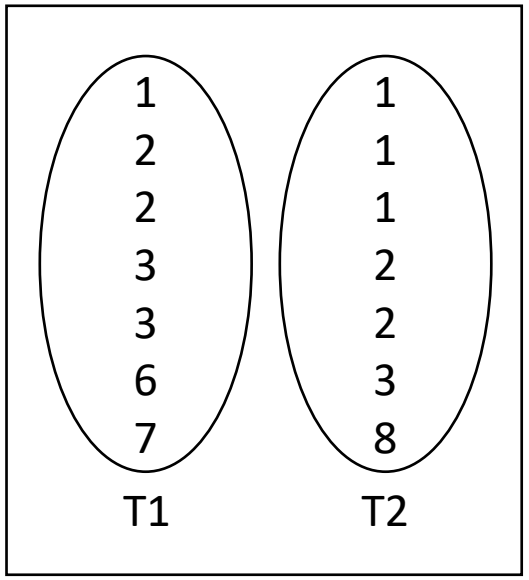
```
SELECT  T1.JKEY  J1, T2.JKEY J2
 FROM TAB1 T1, TAB2 T2
```

# INNER JOIN VS OUTER JOIN VS CROSS JOIN

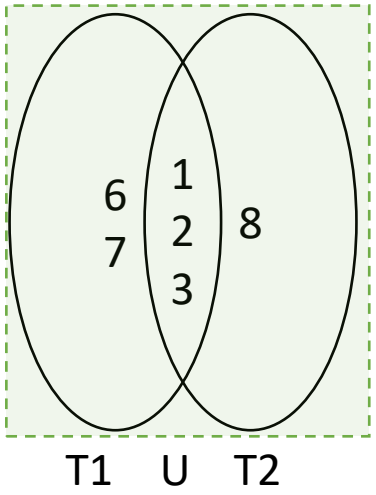


# 집합 연산자 (SET OPERATOR)

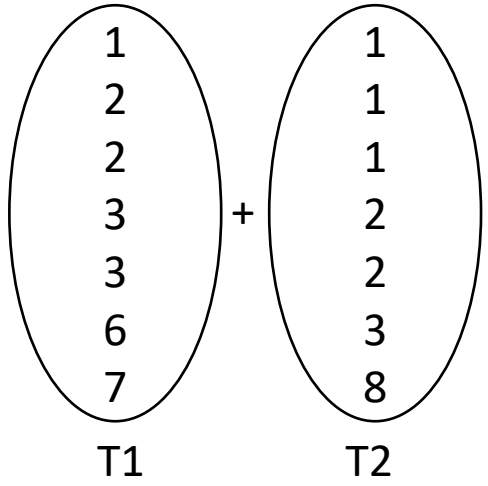
## 집합 T1 과 T2



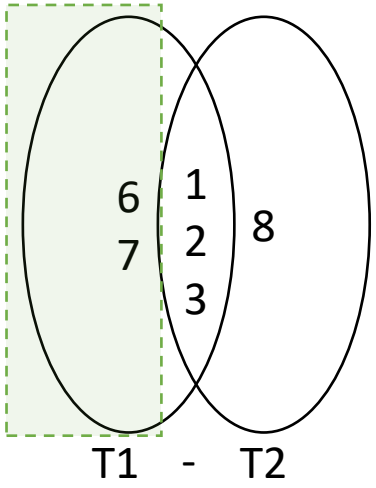

## UNION = 합집합



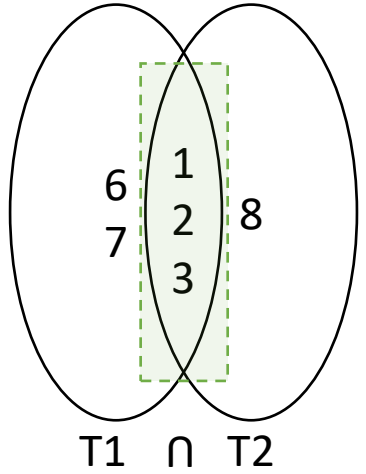

## UNION ALL




## MINUS (EXCEPT) = 차집합




## INTERSECT = 교집합



# 집합 연산자 (SET OPERATOR)

T1

| C1 |
|---|
| 1 |
| 2 |
| 2 |
| 3 |
| 3 |
| 6 |
| 7 |

T2

| C1 |
|---|
| 1 |
| 1 |
| 1 |
| 2 |
| 2 |
| 3 |
| 8 |

## UNION = 합집합

```
SELECT C1 FROM T1
UNION
SELECT C1 FROM T2
```

6건

| C1 |
|---|
| 1 |
| 2 |
| 3 |
| 6 |
| 7 |
| 8 |

## UNION ALL

```
SELECT C1 FROM T1
UNION ALL
SELECT C1 FROM T2
```

14건

| C1 |
|---|
| 1 |
| 2 |
| 2 |
| 3 |
| 3 |
| 6 |
| 7 |
| 1 |
| 1 |
| 1 |
| 2 |
| 2 |
| 3 |
| 8 |

## MINUS (EXCEPT) = 차집합

```
SELECT C1 FROM T1
MINUS
SELECT C1 FROM T2
```

2건

| C1 |
|---|
| 6 |
| 7 |

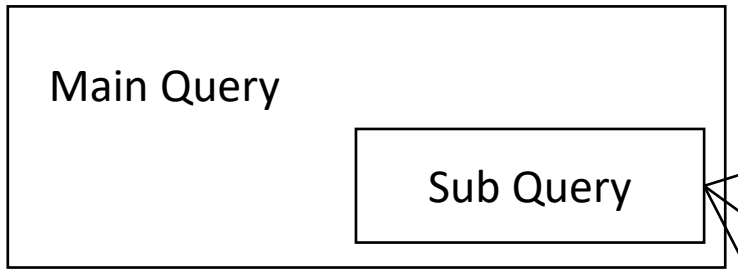
## INTERSECT = 교집합

```
SELECT C1 FROM T1
INTERSECT
SELECT C1 FROM T2
```

3건

| C1 |
|---|
| 1 |
| 2 |
| 3 |

# 서브 쿼리




- 서브쿼리가 메인쿼리에 포함되는 종속적인 관계임.
- 서브쿼리는 메인쿼리의 칼럼을 모두 사용할 수 있지만 메인쿼리는 서브쿼리의 칼럼을 사용할 수 없음.
- 인라인뷰에 정의된 칼럼은 메인쿼리에서 사용 가능.
- 스칼라 서브쿼리는 한 행, 한 칼럼(1Row, 1Column)만을 반환하는 서브쿼리로 ,컬럼을 쓸 수 있는 대부분의 곳에서 사용 가능함 , 그러나 주로 SELECT LIST 에서 사용.

**Scalar Subquery** → OUTER JOIN 과 결과가 같음

```
SELECT EMPNO, ENAME,
(SELECT D.DNAME
  FROM DEPT D
 WHERE D.DEPTNO = A.DEPTNO) DNAME
FROM EMP A;
```

**Inline View** → View 유사

```
SELECT A.item_name, subquery1.total_amt
FROM suppliers A,
  (SELECT supplier_id, SUM(B.amount) AS total_amt
   FROM orders B
   GROUP BY supplier_id) subquery1
WHERE subquery1.supplier_id = A.supplier_id;
```

**Nested Subquery** → 조회조건

```
SELECT *
FROM student A
WHERE A.student_name IN (SELECT B.student_name
    FROM subject B
   WHERE B.subject_name = 'MATH');
```

# 서브 쿼리와 조인(JOIN)

## Scalar Subquery

```
SELECT EMPNO, ENAME,
(SELECT D.DNAME
  FROM DEPT D
 WHERE D.DEPTNO = A.DEPTNO) DNAME
FROM EMP A;
```

=

## Outer Join

```
SELECT EMPNO, ENAME,
       D.DNAME
FROM EMP A LEFT OUTER JOIN DEPT D
    ON D.DEPTNO = A.DEPTNO;
```

---

## Nested Subquery

```
SELECT *
FROM student A
WHERE A.student_name
IN (SELECT B.student_name
      FROM subject B
     WHERE B.subject_name = 'MATH');
```

=

## Join

```
SELECT DISTINCT A.*
FROM student A inner join subject B on
on A.student_name = B.student_name
WHERE B.subject_name = 'MATH';
```

# GROUP BY 와 GROUPING SETS

```
SELECT Color, Dimension, SUM(Quantity) as Quantity
FROM Inventory
GROUP BY Color, Dimension;
```

| Color | Dimension | Quantity |
|---|---|---|
| Red | Large | 10 |
| Blue | Medium | 20 |
| Red | Medium | 15 |
| Blue | Large | 5 |

결과집합

| Color | Dimension | Quantity |
|---|---|---|
| Blue | Medium | 20 |
| Blue | Large | 5 |
| Blue | NULL | 25 |
| Red | Medium | 15 |
| Red | Large | 10 |
| Red | NULL | 25 |
| NULL | NULL | 50 |

# GROUP BY 와 GROUPING SETS (계속..)

```
1 SELECT Color, Dimension, SUM(Quantity) as Quantity
  FROM Inventory
  GROUP BY Color, Dimension
  UNION ALL
2 SELECT Color, NULL, SUM(Quantity) as Quantity
  FROM Inventory
  GROUP BY Color
  UNION ALL
3 SELECT NULL, NULL, SUM(Quantity) as Quantity
  FROM Inventory;
```

=

```
SELECT Color, Dimension, SUM(Quantity) as Quantity
FROM Inventory
GROUP BY
GROUPING SETS ( 1(Color, Dimension) ,2(Color), 3() );
```

결과집합

| Color | Dimension | Quantity |
|---|---|---|
| Blue | Medium | 20 |
| Blue | Large | 5 |
| Blue | NULL | 25 |
| Red | Medium | 15 |
| Red | Large | 10 |
| Red | NULL | 25 |
| NULL | NULL | 50 |

# GROUPING SETS 과 ROLL UP

```
SELECT Color, Dimension, SUM(Quantity) as Quantity
FROM Inventory
GROUP BY
GROUPING SETS ( (Color, Dimension) ,(Color), () );
```

=

```
SELECT Color, Dimension, SUM(Quantity) as Quantity
FROM Inventory
GROUP BY ROLLUP (Color, Dimension);
```

**ROLLUP ( n ) = GROUPING SETS ( n + 1 )**

**ROLLUP (Color, Dimension)**
→ GROUPING SETS ( (Color, Dimension) ,(Color), () )

**ROLLUP (a, b, c)**
→ GROUPING SETS ( (a, b, c) ,(a, b), (a), () )

결과집합

| Color | Dimension | Quantity |
|---|---|---|
| Blue | Medium | 20 |
| Blue | Large | 5 |
| Blue | NULL | 25 |
| Red | Medium | 15 |
| Red | Large | 10 |
| Red | NULL | 25 |
| NULL | NULL | 50 |

# GROUPING SETS 과 CUBE

```
SELECT Color, Dimension, SUM(Quantity) as Quantity
FROM Inventory
GROUP BY
GROUPING SETS
( (Color, Dimension) ,(Color) ,(Dimension) ,() );
```

=

```
SELECT Color, Dimension, SUM(Quantity) as Quantity
FROM Inventory
GROUP BY CUBE (Color, Dimension);
```

**CUBE ( n ) = GROUPING SETS ( $2^n$ )**

**CUBE (Color, Dimension)**

→ GROUPING SETS

( (Color, Dimension) ,(Color) ,(Dimension) ,() )

**CUBE (a, b, c)**

→ GROUPING SETS

( (a, b, c), (a, b), (a, c), (b, c), (a), (b), (c), () )

결과집합

| Color | Dimension | Quantity |
|---|---|---|
| Blue | Medium | 20 |
| Blue | Large | 5 |
| Blue | NULL | 25 |
| Red | Medium | 15 |
| Red | Large | 10 |
| Red | NULL | 25 |
| NULL | Medium | 35 |
| NULL | Large | 15 |
| NULL | NULL | 50 |

# ROLLUP과 CUBE의 컬럼 순서

**ROLLUP**

컬럼 순서에 따라 결과집합이 달라진다.

**CUBE**

컬럼 순서가 달라져도 결과집합은 같다. (정렬 순서는 다를 수 있음)

**A**

```
SELECT Color, Dimension, SUM(Quantity) as Quantity
FROM Inventory
GROUP BY ROLLUP (Color, Dimension);
```

≠

**B**

```
SELECT Color, Dimension, SUM(Quantity) as Quantity
FROM Inventory
GROUP BY ROLLUP (Dimension, Color);
```

**C**

```
SELECT Color, Dimension, SUM(Quantity) as Quantity
FROM Inventory
GROUP BY CUBE (Color, Dimension);
```

=

**D**

```
SELECT Color, Dimension, SUM(Quantity) as Quantity
FROM Inventory
GROUP BY CUBE (Dimension, Color);
```

**A**

```
.....
GROUPING SETS ( (Color, Dimension) ,(Color), () );
```

**B**

```
.....
GROUPING SETS ( (Dimension ,Color), (Dimension), () );
```

**C** **D**

```
.....
GROUPING SETS
( (Color, Dimension) ,(Color) ,(Dimension) ,() );
```

# WINDOW FUNCTION (윈도우 함수) - ROW_NUMBER, RANK, DENSE_RANK 차이

SQL

```
SELECT JOB, ENAME, SAL,
        RANK() OVER (ORDER BY SAL DESC) RANK,
        DENSE_RANK() OVER (ORDER BY SAL DESC) DENSE_RANK,
        ROW_NUMBER() OVER (ORDER BY SAL DESC) ROW_NUMBER
FROM EMP;
```

결과집합

| JOB | ENAME | SAL | RANK | DENSE_RANK | ROW_NUMBER |
|---|---|---|---|---|---|
| PRESIDENT | KING | 5000 | 1 | 1 | 1 |
| ANALYST | FORD | 3000 | 2 | 2 | 2 |
| ANALYST | SCOTT | 3000 | 2 | 2 | 3 |
| MANAGER | JONES | 2975 | 4 | 3 | 4 |
| MANAGER | BLAKE | 2850 | 5 | 4 | 5 |
| MANAGER | CLARK | 2450 | 6 | 5 | 6 |
| SALESMAN | ALLEN | 1600 | 7 | 6 | 7 |
| SALESMAN | TURNER | 1500 | 8 | 7 | 8 |
| CLERK | MILLER | 1300 | 9 | 8 | 9 |
| SALESMAN | WARD | 1250 | 10 | 9 | 10 |
| SALESMAN | MARTIN | 1250 | 10 | 9 | 11 |
| CLERK | ADAMS | 1100 | 12 | 10 | 12 |
| CLERK | JAMES | 950 | 13 | 11 | 13 |
| CLERK | SMITH | 800 | 14 | 12 | 14 |

# 수고하셨습니다.