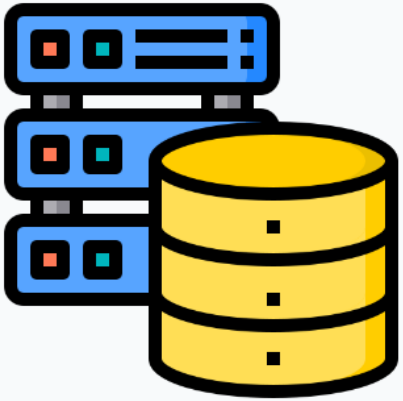
3장. SQL – DML

데이터 베이스 구성요소



SELECT – 자료 검색

조건절

SELECT 칼럼이름 (or 별칭) **FROM** 테이블 이름

WHERE 조건문

연산자	기능
=, <>, >, <	비교(같다, 같지않다, 크다, 작다)
BETWEEN.. AND..	범위
IN (A, B, C)	포함(조건값이 명확)
LIKE	조회(조건값이 불명확), % 사용
IS NULL	데이터 값이 NULL 인 경우

SELECT – 자료 검색

```sql
-- 모든 도서의 이름과 가격을 검색하시오.
SELECT bookname, price
FROM book;

-- 모든 도서의 도서번호, 도서이름, 출판사, 가격을 검색하시오
SELECT bookid, bookname, publisher, price
FROM book;

-- 도서 테이블에 있는 모든 출판사를 검색하시오(중복 제거)
SELECT DISTINCT publisher
FROM book;

-- 가격이 20000원 미만인 도서를 검색하시오
SELECT *
FROM book
WHERE price < 20000;
```

PUBLISHER
굿스포츠
나무수
대한미디어
이상미디어
삼성당
Pearson

BOOKID	BOOKNAME	PUBLISHER	PRICE
1	축구의 역사	굿스포츠	7000
2	축구아는 여자	나무수	13000
5	피겨 교본	굿스포츠	8000
6	양궁의 실제	굿스포츠	6000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

SELECT – 자료 검색

```sql
-- 가격이 10000원 이상 20000원 이하인 도서를 검색하시오
-- BETWEEN ~ AND ~
SELECT *
FROM book
WHERE price BETWEEN 10000 AND 20000;

SELECT *
FROM book
WHERE price >= 10000 AND price <= 20000;

-- 출판사가 '굿스포츠' 혹은 '대한미디어'인 도서를 검색하시오
SELECT *
FROM book
WHERE publisher IN ('굿스포츠', '대한미디어');
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
2	축구아는 여자	나무수	13000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

```sql
-- 출판사가 '굿스포츠' 혹은 '대한미디어'가 아닌 출판사
SELECT *
FROM book
WHERE publisher NOT IN ('굿스포츠', '대한미디어');
```

SELECT – 자료 검색

```sql
-- '축구의 역사'를 출간한 출판사를 검색하시오
SELECT bookname, publisher
FROM book
WHERE bookname LIKE '축구의 역사';

-- 도서 이름에 '축구'가 포함된 출판사를 검색하시오
SELECT bookname, publisher
FROM book
WHERE bookname LIKE '%축구%';

-- '축구'에 관한 도서 중 가격이 20000원 이상인 도서를 검색하시오
SELECT *
FROM book
WHERE bookname LIKE '%축구%' AND price >= 20000;
```

BOOKNAME	PUBLISHER
축구의 역사	굿스포츠
축구아는 여자	나무수
축구의 이해	대한미디어

BOOKID	BOOKNAME	PUBLISHER	PRICE
3	축구의 이해	대한미디어	22000

SELECT – 자료 검색

정렬

SELECT 칼럼이름 (or 별칭) **FROM** 테이블 이름

ORDER BY 칼럼 이름 ASC / DESC (오름차순/내림차순)

```
-- 도서를 이름순으로 검색하시오
SELECT *
FROM book
ORDER BY bookname;

-- 도서를 가격순으로 검색하고, 가격이 같으면 이름순으로 검색하시오
SELECT *
FROM book
ORDER BY price, bookname;

-- 도서를 가격의 내림차순으로 검색하고, 가격이 같으면 출판사를 오름차순으로 검색하시오
SELECT *
FROM book
ORDER BY price DESC, publisher ASC;
```

SELECT – 자료 검색

집계(그룹) 함수

SELECT 그룹함수 (칼럼이름) **FROM** 테이블 이름

ORDER BY 칼럼 이름 (ASC/DESC)

함수	기능
SUM(칼럼)	합계
COUNT(*)	개수
AVG(칼럼)	평균
MAX(칼럼)	최대값
MIN(칼럼)	최소값

SELECT – 자료 검색

```sql
SELECT MAX(price) AS 최고가격, MIN(price) AS 최저가격
FROM book;
```

```sql
-- 고객이 주문한 도서의 총 판매액을 구하시오
SELECT SUM(saleprice) AS 총매출
FROM orders;

-- '김연아' 고객이 주문한 도서의 총 판매액을 구하시오
SELECT SUM(saleprice) AS 총매출
FROM orders
WHERE custid = 2;

-- 고객이 주문한 도서의 총 판매액, 평균값을 구하시오
SELECT SUM(saleprice) AS Total,
       AVG(saleprice) AS Average
FROM orders;

-- 마당 서점의 도서 판매 건수를 구하시오
SELECT COUNT(*) AS 총판매건수
FROM orders;
```

총매출
118000

총매출
15000

TOTAL	AVERAGE
118000	11800

총판매건수
10

SELECT – 자료 검색

GROUP BY: 그룹으로 묶기

```
SELECT  그룹함수 (칼럼이름) FROM  테이블 이름
[WHERE  조건식]
GROUP BY 칼럼 이름
```

HAVING 절은 GROUP BY 질의 결과 나타내는 그룹을 제한하는 역할

```
SELECT  그룹함수 (칼럼이름) FROM  테이블 이름
[WHERE  조건식]
GROUP BY 칼럼 이름
HAVING  조건식
```

SELECT – 자료 검색

```
-- 고객별로 주문한 도서의 총 수량과 판매액을 구하시오
SELECT custid, COUNT(*) 도서수량, SUM(saleprice) 총액
FROM orders
GROUP BY custid;

-- 가격이 8000원 이상인 도서를 구매한 고객에 대하여 고객별 주문 도서의 총 수량을 구하시오.
-- 단 2권 이상 구매한 고객만 구하시오.
-- HAVING 절은 GROUP BY 질의 결과 나타내는 그룹을 제한하는 역할을 한다.
SELECT custid, COUNT(*) 도서수량
FROM orders
WHERE saleprice >= 8000
GROUP BY custid
HAVING count(*) >= 2;
```

CUSTID	도서수량	총액
1	3	39000
2	2	15000
3	3	31000
4	2	33000

CUSTID	도서수량
1	2
4	2
3	2

실습문제

- 마당 서점의 고객 테이블을 검색하시오.

1. 모든 고객의 이름과 주소를 검색하시오.
2. 모든 고객의 이름, 주소, 전화번호를 검색하시오.
3. 주소가 ' 영국'인 고객을 검색하시오
4. 고객의 이름이 '김연아' 혹은 '안산'인 고객을 검색하시오
5. 주소가 '대한민국'이 아닌 고객을 검색하시오.
6. 전화번호가 없는 고객을 검색하시오
7. 고객을 이름순으로 정렬하시오.
8. 고객의 총 인원수를 구하시오

실습 문제

```
CREATE TABLE emp(
    -- 칼럼 이름  자료형
    empno    NUMBER(3) PRIMARY KEY,      -- 사원번호
    ename    VARCHAR2(20) NOT NULL,      -- 사원이름
    gender   VARCHAR2(6) NOT NULL,       -- 성별
    sal      NUMBER(10),                 -- 급여
    hire_date  VARCHAR2(20) NOT NULL     -- 입사일
);

INSERT INTO emp VALUES (100, '이강', '남자', 3000000, '2019-01-01');
INSERT INTO emp VALUES (101, '김산', '여자', 2500000, '2020-05-15');
INSERT INTO emp VALUES (102, '오상식', '남자', 5000000, '2015-02-22');
INSERT INTO emp VALUES (103, '박신입', '여자', '',  '2023-10-01');
```

실습문제

- 사원 테이블을 관리(검색, 수정, 삭제)하시오.

1. 사원을 입사일 순으로 정렬하시오(오름차순 정렬)
2. 사원을 급여 순으로 정렬하시오(오름차순 정렬)
3. 급여가 300만원 이하인 사원을 급여가 많은 순으로 정렬하시오
4. 급여가 없는 사원을 검색하시오.
5. 사원의 총 수를 구하시오.
6. 사원의 급여 합계와 평균을 구하시오.
7. 이름이 김산인 사원의 성별을 남자로 변경하시오.
8. 이름이 오상식인 사원을 삭제하시오.

조인(JOIN)

조인이란?

조인은 한 개 이상의 테이블과 테이블을 서로 연결하여 사용하는 기법을 말한다.

동등조인, 외부 조인, 자체 조인등이 있다.

조인 기법	개념
동등 조인(equi join)	조인 조건이 정확히 일치하는 경우에 결과를 출력
외부 조인(outer join)	조인 조건이 정확히 일치하지 않아도 모든 결과를 출력
자체 조인(self join)	자체 테이블에서 조인하고자 할 때 사용

문법 규칙

```
SELECT  테이블이름1.열 이름1, 테이블이름2.열 이름2
FROM 테이블 이름 1, 테이블 이름2
WHERE 테이블 이름 1.열 이름1 = 테이블 이름2.열 이름 2
```

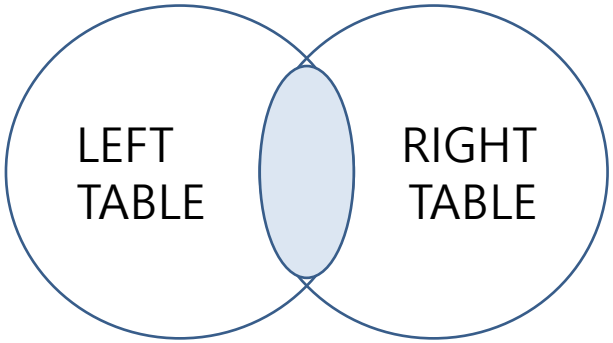
두 테이블의 열이 갖고 있는 데이터 값을 논리적으로 연결

조인(JOIN)

동등 조인(EQUI JOIN)

Equal(=) 조건으로 JOIN 하는 것이다.

양쪽 테이블에서 조인 조건이 일치하는 행만 가져오는 조인으로 기본키와 외래키의 관계를 이용하여 조인하기도 하고 키가 아니더라도 다양한 조건으로 조인할 수 있다.



조인(JOIN)

```
-- 고객과 고객의 주문에 관한 데이터를 모두 검색하시오
SELECT *
FROM customer, orders
WHERE customer.custid = orders.custid;
```

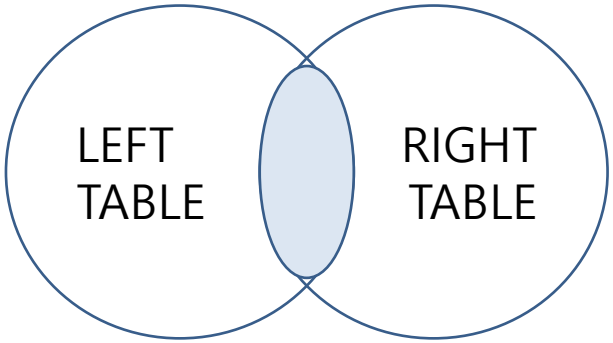
CUSTID	NAME	ADDRESS	PHONE	ORDERID	CUSTID_1	BOOKID	SALEPRICE	ORDERDATE
1	박지성	영국 맨체스타	000-5000-0001	1	1	1	6000	18/07/01
1	박지성	영국 맨체스타	000-5000-0001	2	1	3	21000	18/07/03
2	김연아	대한민국 서울	000-6000-0001	3	2	5	8000	18/07/03
3	안산	대한민국 광주광역시	000-7000-0001	4	3	6	6000	18/07/04
4	류현진	미국 토론토	(null)	5	4	7	20000	18/07/05
1	박지성	영국 맨체스타	000-5000-0001	6	1	2	12000	18/07/07
4	류현진	미국 토론토	(null)	7	4	8	13000	18/07/07
3	안산	대한민국 광주광역시	000-7000-0001	8	3	10	12000	18/07/08
2	김연아	대한민국 서울	000-6000-0001	9	2	10	7000	18/07/09
3	안산	대한민국 광주광역시	000-7000-0001	10	3	8	13000	18/07/10

조인(JOIN)

동등 조인(EQUI JOIN)

Equal(=) 조건으로 JOIN 하는 것이다.

양쪽 테이블에서 조인 조건이 일치하는 행만 가져오는 조인으로 기본키와 외래키의 관계를 이용하여 조인하기도 하고 키가 아니더라도 다양한 조건으로 조인할 수 있다.



조인(JOIN)

```sql
-- 고객의 이름과 고객이 주문한 도서의 판매가격을 검색하시오
SELECT customer.name, orders.saleprice
FROM customer, orders
WHERE customer.custid = orders.custid;
```

NAME	SALEPRICE
박지성	6000
박지성	21000
김연아	8000
안산	6000
류현진	20000
박지성	12000
류현진	13000
안산	12000
김연아	7000
안산	13000

```sql
-- 고객의 이름과 고객이 주문한 도서의 판매가격을 검색하고
-- 고객의 이름은 오름차순, 판매가격은 내림차순 정렬하시오.
SELECT customer.name, orders.saleprice
FROM customer, orders
WHERE customer.custid = orders.custid
ORDER BY customer.name, orders.saleprice DESC;
```

	N...	SALEPRICE
1	김연아	8000
2	김연아	7000
3	류현진	20000
4	류현진	13000
5	박지성	21000
6	박지성	12000
7	박지성	6000
8	안산	13000
9	안산	12000
10	안산	6000

조인(JOIN)

```
-- 고객의 이름과 주문한 도서의 이름, 주문일, 주문금액 검색
SELECT customer.name, book.bookname,
          orders.orderdate, orders.saleprice
FROM customer, book, orders
WHERE customer.custid = orders.custid
  AND book.bookid = orders.bookid;
```

	NAME	BOOKNAME	ORDERDATE	SALEPRICE
1	박지성	축구의 역사	2018-07-01	6000
2	박지성	축구아는 여자	2018-07-07	12000
3	박지성	축구의 이해	2018-07-03	21000
4	김연아	피겨 교본	2018-07-03	8000
5	안산	양궁의 실제	2018-07-04	6000
6	류현진	야구의 추억	2018-07-05	20000
7	안산	야구를 부탁해	2018-07-10	13000
8	류현진	야구를 부탁해	2018-07-07	13000
9	김연아	Olympic Champions	2018-07-09	7000
10	안산	Olympic Champions	2018-07-08	12000

조인(JOIN)

```sql
-- '박지성' 고객의 주문내역을 검색하시오
SELECT *
FROM customer, orders
WHERE customer.custid = orders.custid
AND customer.name = '박지성';
```

CUSTID	NAME	ADDRESS	PHONE	ORDERID	CUSTID_1	BOOKID	SALEPRICE	ORDERDATE
1	박지성	영국 맨체스타	000-5000-0001	1	1	1	6000	18/07/01
1	박지성	영국 맨체스타	000-5000-0001	2	1	3	21000	18/07/03
1	박지성	영국 맨체스타	000-5000-0001	6	1	2	12000	18/07/07

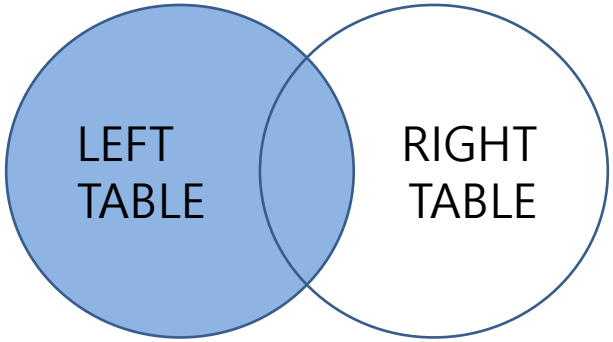
```sql
-- 고객아이디 및 고객 이름별 주문 금액 검색(이름별 정렬)
SELECT cus.custid, cus.name, SUM(ord.saleprice)
FROM customer cus, orders ord
WHERE cus.custid = ord.custid
GROUP BY cus.custid, cus.name
ORDER BY cus.name;
```

	CUSTID	NAME	SUM(ORD.SALEPRICE)
1	2	김연아	15000
2	4	류현진	33000
3	1	박지성	39000
4	3	안산	31000

조인(JOIN)

외부 조인(Outer JOIN)

```
-- 도서를 주문하지 않은 고객까지 포함하여 고객의 이름과 주문 도서의 판매가격 검색
SELECT cus.custid, cus.name, ord.saleprice
FROM customer cus, orders ord
WHERE cus.custid = ord.custid(+);
```



	NAME	SALEPRICE
1	박지성	6000
2	박지성	21000
3	김연아	8000
4	안산	6000
5	류현진	20000
6	박지성	12000
7	류현진	13000
8	안산	12000
9	김연아	7000
10	안산	13000
11	손흥민	(null)

조인(JOIN) 실습

```sql
-- product 테이블 생성
CREATE TABLE product(
    product_code  CHAR(6) PRIMARY KEY,    -- 상품 코드
    product_name  VARCHAR2(50) NOT NULL, -- 상품명
    price         NUMBER NOT NULL         -- 가격
);

INSERT INTO product VALUES ('100001', '무소음 무선 마우스', 25000);
INSERT INTO product VALUES ('100002', '기계식 게이밍 키보드', 30000);
INSERT INTO product VALUES ('100003', '무결점 패널 광시야각 모니터', 120000);
```

조인(JOIN) 실습

```sql
-- product_review 테이블 생성
CREATE TABLE product_review(
    review_no     NUMBER PRIMARY KEY,      -- 리뷰번호
    product_code  CHAR(6),                 -- 상품코드
    member_id     VARCHAR2(20) NOT NULL,   -- 회원아이디
    content       CLOB NOT NULL,           -- 리뷰내용
    regdate       DATE DEFAULT SYSDATE,    -- 리뷰작성일
    FOREIGN KEY(product_code) REFERENCES product(product_code)
);
CREATE SEQUENCE seq_prod NOCYCLE; -- 리뷰 자동 순번
INSERT INTO product_review (review_no, product_code, member_id, content)
VALUES (seq_prod.NEXTVAL, '100001', 'today123', '무소음인데 소음이 조금 있는 듯');
INSERT INTO product_review (review_no, product_code, member_id, content)
VALUES (seq_prod.NEXTVAL, '100001', 'cloud100', '무선이라 정말 편하네요');
INSERT INTO product_review (review_no, product_code, member_id, content)
VALUES (seq_prod.NEXTVAL, '100002', 'sky321', '게임할 맛 납니다.');
```

조인(JOIN) 실습

```sql
-- 동등 조인(EQUI JOIN) 실습
-- 상품에 review가 있는 상품코드, 상품명, 리뷰를 작성한 회원아이디,
-- 리뷰내용, 리뷰 등록일을 출력
SELECT a.product_code,
       a.product_name,
       b.member_id,
       b.content,
       b.regdate
FROM product a, product_review b
WHERE a.product_code = b.product_code;
```

```sql
-- 위의 검색 데이터에서 무소음 무선 마우스 리뷰만 출력 출력
SELECT a.product_code,
       a.product_name,
       b.member_id,
       b.content,
       b.regdate
FROM product a, product_review b
WHERE a.product_code = b.product_code
  AND a.product_code = '100001';
```

	PRODUCT_CODE	PRODUCT_NAME	MEMBER_ID	CONTENT	REGDATE
1	100001	무소음 무선 마우스	today123	무소음인데 소음이 조금 있는 듯	23/10/15
2	100001	무소음 무선 마우스	cloud100	무선이라 정말 편하네요	23/10/15

조인(JOIN) 실습

```
-- event 테이블 생성
CREATE TABLE event(
    event_no    NUMBER PRIMARY KEY,      -- 이벤트 번호
    event_name  VARCHAR2(30) NOT NULL,   -- 이벤트 이름
    start_date  VARCHAR2(20) NOT NULL,   -- 이벤트 시작일
    end_date    VARCHAR2(20) NOT NULL    -- 이벤트 종료일
);
CREATE SEQUENCE seq_event NOCYCLE; -- 이벤트 자동 순번

INSERT INTO event (event_no, event_name, start_date, end_date)
VALUES (seq_event.NEXTVAL, '20% 할인 쿠폰 증정', '2023-10-10', '2023-10-20');
INSERT INTO event (event_no, event_name, start_date, end_date)
VALUES (seq_event.NEXTVAL, '마우스 패드 증정', '2023-11-15', '2023-11-10');
INSERT INTO event (event_no, event_name, start_date, end_date)
VALUES (seq_event.NEXTVAL, '벚꽃 축제 페스티벌', '2023-12-10', '2023-12-30');
```

조인(JOIN) 실습

2. Non EQUI JOIN

Non Equi Join은 Equal(=) 조건이 아닌 다른 조건 (BETWEEN, >, <)으로 조인하는 방식이다.

예를 들어 이벤트 기간 동안 리뷰를 작성한 고객에게 사은품을 주는 행사를 하고 있다고 가정하면 이 경우 리뷰 테이블과 이벤트 테이블이 JOIN되어야 함

```
-- 리뷰 테이블과 이벤트 테이블 조인
SELECT a.event_name,
       b.member_id,
       b.content,
       b.regdate
FROM event a, product_review b
WHERE b.regdate BETWEEN a.start_date AND a.end_date;
```

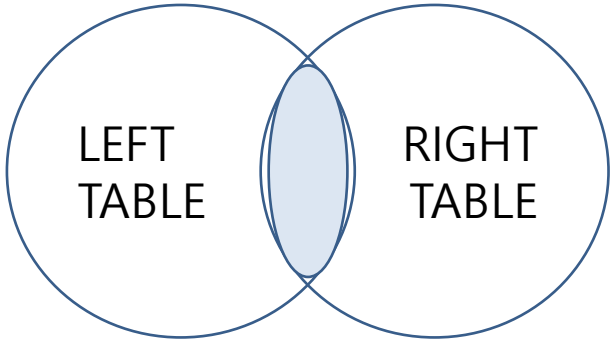
	EVENT_NAME	MEMBER_ID	CONTENT	REGDATE
1	20% 할인 쿠폰 증정	today123	무소음인데 소음이 조금 있는 듯	23/10/15
2	20% 할인 쿠폰 증정	cloud100	무선이라 정말 편하네요	23/10/15
3	20% 할인 쿠폰 증정	sky321	게임할 맛 납니다.	23/10/15

조인(JOIN)

STANDARD JOIN(표준 조인)

ANSI SQL중 하나로써 SQL 문법에 관계없이 모든 DBMS에서 사용가능한 조인을 말함

1. INNER JOIN ~ ON(FROM 절에서 사용)




```
-- 고객의 이름과 고객이 주문한 도서의 판매가격을 검색하시오
--  A INNER JOIN B ON 조건
SELECT cus.name, ord.saleprice
FROM customer cus INNER JOIN orders ord
    ON cus.custid = ord.custid;
```

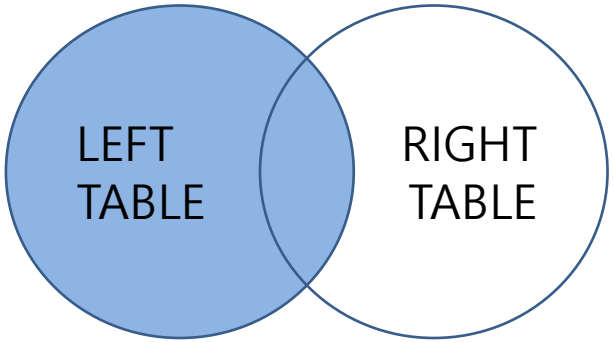
	NAME	SALEPRICE
1	김연아	7000
2	김연아	8000
3	류현진	20000
4	류현진	13000
5	박지성	12000
6	박지성	6000
7	박지성	21000
8	안산	13000
9	안산	12000
10	안산	6000

조인(JOIN)

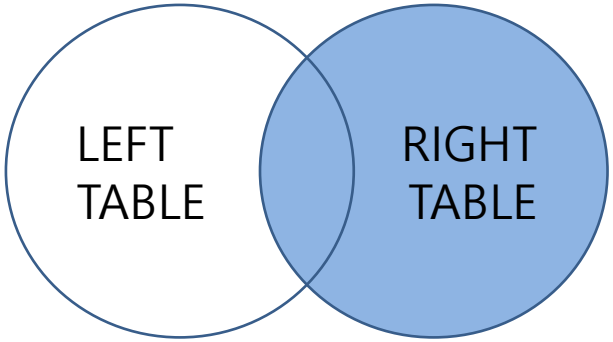
2. OUTER JOIN

JOIN 조건에 충족하는 데이터가 아니어도 출력될 수 있는 방식이다.

LEFT OUTER JOIN



RIGHT OUTER JOIN



조인(JOIN)

LEFT OUTER JOIN

JOIN 조건에 충족하는 데이터가 아니어도 출력될 수 있는 방식이다.

FROM절 : LEFT OUTER JOIN ~ ON

오른쪽에 데이터가 있던지 말던지 왼쪽은 모두 출력되고 본다.

오른쪽 칼럼에 데이터가 없으면 NULL로 출력됨

```
-- 도서를 주문하지 않은 고객을 포함하여
-- 고객의 이름과 주문 판매가격을 검색

SELECT cs.name, od.saleprice
FROM customer cs LEFT OUTER JOIN orders od
    ON cs.custid = od.custid;
```

	NAME	SALEPRICE
1	박지성	6000
2	박지성	21000
3	김연아	8000
4	안산	6000
5	류현진	20000
6	박지성	12000
7	류현진	13000
8	안산	12000
9	김연아	7000
10	안산	13000
11	손흥민	(null)

조인(JOIN) 실습

```sql
-- 리뷰를 남긴 상품 정보 검색하기
-- 1. 동등조인
SELECT a.product_code,
       a.product_name,
       b.member_id,
       b.content,
       b.regdate
FROM product a, product_review b
WHERE a.product_code = b.product_code;
-- 2. INNER JOIN ~ ON
SELECT a.product_code,
       a.product_name,
       b.member_id,
       b.content,
       b.regdate
FROM product a INNER JOIN product_review b
   ON a.product_code = b.product_code;
```

	PRODUCT_CODE	PRODUCT_NAME	MEMBER_ID	CONTENT	REGDATE
1	100001	무소음 무선 마우스	today123	무소음인데 소음이 조금 있는 듯	23/10/15
2	100001	무소음 무선 마우스	cloud100	무선이라 정말 편하네요	23/10/15
3	100002	기계식 게이밍 키보드	sky321	게임할 맛 납니다.	23/10/15

조인(JOIN) 실습

```sql
-- 리뷰를 남기지 않은 상품도 포함하여 모든 상품 정보 검색하기
-- 1. 동등조인
SELECT a.product_code,
       a.product_name,
       b.member_id,
       b.content,
       b.regdate
FROM product a, product_review b
WHERE a.product_code = b.product_code(+);

-- 2. LEFT OUTER JOIN
SELECT a.product_code,
       a.product_name,
       b.member_id,
       b.content,
       b.regdate
FROM product a LEFT OUTER JOIN product_review b
   ON a.product_code = b.product_code;
```

	PRODUCT_CODE	PRODUCT_NAME	MEMBER_ID	CONTENT	REGDATE
1	100001	무소음 무선 마우스	today123	무소음인데 소음이 조금 있는 듯	23/10/15
2	100001	무소음 무선 마우스	cloud100	무선이라 정말 편하네요	23/10/15
3	100002	기계식 게이밍 키보드	sky321	게임할 맛 납니다.	23/10/15
4	100003	무결점 패널 광시야각 모니터	(null)	(null)	(null)

뷰(view)

뷰(VIEW)

뷰는 하나 이상의 테이블을 합하여 만든 가상의 테이블로써 실제 테이블처럼 사용할 수 있도록 만든 데이터베이스 개체이다.

데이터베이스에서 SELECT문을 저장한 Object라고 할 수 있음

뷰를 사용하는 이유는 원본 테이블의 데이터를 안전하게 유지하면서 필요한 사용자에게 적절한 데이터를 제공(보고서 형태) 할 수 있다.

뷰의 생성

```
CREATE VIEW 뷰이름
AS SELECT 문
```

뷰의 삭제

```
DROP VIEW 뷰이름
```

뷰(view)

```sql
-- 주소에 '대한민국'을 포함하고 있는 고객들로 구성된 뷰 생성
CREATE VIEW vw_customer AS
SELECT * FROM customer
WHERE address LIKE '%대한민국%';

SELECT * FROM vw_customer;

SELECT *
FROM vw_customer
WHERE address LIKE '%광주%';

-- 뷰 삭제
DROP TABLE vw_customer;
```

CUSTID	NAME	ADDRESS	PHONE
2	김연아	대한민국 서울	000-6000-0001
3	안산	대한민국 광주광역시	000-7000-0001

뷰(view) 실습 예제

```
CREATE TABLE JOB_INFO(
    JOB_ID  VARCHAR(10),
    MIN_SALARY  NUMBER,
    MAX_SALARY  NUMBER
);

INSERT INTO JOB_INFO VALUES ('AD_PRES', 20080, 40000);
INSERT INTO JOB_INFO VALUES ('AD_VP', 15000, 30000);
INSERT INTO JOB_INFO VALUES ('AD_ASST', 3000, 6000);
INSERT INTO JOB_INFO VALUES ('FI_MGR', 8200, 16000);
INSERT INTO JOB_INFO VALUES ('FI_ACCOUNT', 4200, 9000);
INSERT INTO JOB_INFO VALUES ('AC_MGR', 8200, 16000);
INSERT INTO JOB_INFO VALUES ('AC_ACCOUNT', 4200, 9000);
```

뷰(view) 실습 예제

```sql
-- 집계 함수 - COUNT(), SUM(), AVG()
SELECT COUNT(*) 총개수,
       ROUND(AVG(MIN_SALARY), -1) 최저급여평균,
       AVG(MAX_SALARY) 최대급여평균
FROM JOB_INFO;

-- 최저 급여가 5000 달러 이상인 직업 아이디를 검색하시오
SELECT JOB_ID, MIN_SALARY "min_sal"
FROM JOB_INFO
WHERE MIN_SALARY > 5000;
--WHERE min_sal > 5000;  -- 실행순서가 SELECT 전이므로 별칭 사용 불가
```

```sql
/*실행순서
 1. FROM절
 2. WHERE 절
 3. SELECT 절
 4. ORDER BY
*/
```

뷰(view) 실습 예제

```
-- 최저 급여가 5000 달러 이상인 직업 아이디를 검색한 뷰 생성
-- CREATE VIEW 뷰이름 AS 구문
CREATE VIEW V_JOB_INFO
AS SELECT *
   FROM JOB_INFO
   WHERE MIN_SALARY > 5000;

SELECT * FROM V_JOB_INFO;

-- 최고급여와 최저급여의 차가 8000 이상인 직업아이디의 수를 검색하시오
SELECT COUNT(*)
FROM V_JOB_INFO
WHERE MAX_SALARY - MIN_SALARY > 8000;
```

	JOB_ID	MIN_SALARY	MAX_SALARY
1	AD_PRES	20080	40000
2	AD_VP	15000	30000
3	FI_MGR	8200	16000
4	AC_MGR	8200	16000

뷰(view) 실습 예제

```sql
-- 최고급여와 최저급여의 차가 8000 이상인 직업아이디를 검색하시오
SELECT JOB_ID
FROM V_JOB_INFO
WHERE MAX_SALARY - MIN_SALARY > 8000;

-- 직업 아이디의 이름이 AD로 시작하는 자료를 검색하시오
SELECT JOB_ID
FROM V_JOB_INFO
WHERE JOB_ID LIKE 'AD%';

-- 뷰 삭제
DROP VIEW V_JOB_INFO;
```

	JOB_ID
1	AD_PRES
2	AD_VP

서브 쿼리

서브 쿼리(Sub-Query)란

부속 질의는 하나의 SQL문 안에 다른 SQL문이 중첩된 질의를 말한다.

다른 테이블에서 가져온 데이터로 현재 테이블에 있는 정보를 찾거나 가공할 때 사용한다.

최종 결과를 출력하는 쿼리를 메인 쿼리라고 한다면, 이를 위한 중간단계 혹은 보조 역할을 하는 SELECT문을 서브 쿼리라 한다.

1. WHERE 절 부속질의

```
-- 가장 비싼 도서의 이름을 검색하시오
SELECT bookname, price
FROM book
WHERE price = (SELECT MAX(price) FROM book);
```

BOOKNAME	PRICE
골프 바이블	35000

서브 쿼리

```
-- 다중행 서브쿼리 : 검색 결과가 여러개인 경우 IN 사용
-- 도서를 구매한 적이 있는 고객을 검색하시오
SELECT name
FROM customer
 WHERE custid IN (SELECT custid
                  FROM orders);
```

NAME
박지성
김연아
안산
류현진

```
-- 단일행 서브쿼리 : 검색 결과가 한 개인 경우 '=' 사용
-- 박지성 고객의 주문 내역을 검색하시오
SELECT *
FROM orders
 WHERE custid = (SELECT custid
                 FROM customer
                 WHERE name = '박지성');
```

ORDERID	CUSTID	BOOKID	SALEPRICE	ORDERDATE
1	1	1	6000	18/07/01
2	1	3	21000	18/07/03
6	1	2	12000	18/07/07

서브 쿼리

```sql
-- 이상 미디어에서 출판한 도서를 구매한 고객의 이름을 검색하시오
SELECT name
FROM customer
WHERE custid IN (SELECT custid
                 FROM orders
                 WHERE bookid IN(SELECT bookid
                                 FROM book
                                 WHERE publisher='이상미디어'));
```

NAME
류현진
안산

```sql
-- join
SELECT DISTINCT cs.name
FROM book bk, customer cs, orders od
WHERE bk.bookid = od.bookid
  AND cs.custid = od.custid
  AND bk.publisher = '이상미디어';
```

서브 쿼리

```sql
-- 급여가 가장 많은 사원과 가장 적은 사원 검색
SELECT ename, salary
FROM emp
WHERE salary = (SELECT MAX(salary) FROM emp)
   OR salary = (SELECT MIN(salary) FROM emp);
```

	ENAME	SALARY
1	오상식	5000000
2	이신입	2000000

서브 쿼리

```
-- 출판사별로 출판사의 평균 도서 가격보다 비싼 도서를 검색하시오
-- 튜플 변수 - 테이블 이름의 별칭
SELECT AVG(price) 가격평균
FROM book;

SELECT b1.bookname
FROM book b1
WHERE b1.price > (SELECT AVG(b2.price)
                  FROM book b2
                  WHERE b2.publisher = b1.publisher);
```

BOOKNAME
골프 바이블
피겨 교본
야구의 추억

서브 쿼리

2. FROM 부속질의 – 인라인 뷰

인라인 뷰는 FROM 절에서 사용되는 부속질의를 말한다..

```
-- 고객 번호가 2이하인 고객의 이름과 고객의 판매액을 검색하시오.
-- 1. 동등 조인을 사용한 경우
SELECT cs.name, SUM(od.saleprice) AS total
FROM customer cs, orders od
WHERE cs.custid = od.custid
  AND cs.custid <= 2
GROUP BY cs.name;
--HAVING SUM(od.saleprice) >= 20000;

-- 2. subquery 사용 : FROM절(인라인뷰)
SELECT cs.name, SUM(od.saleprice) total
FROM (SELECT * FROM customer
        WHERE custid <= 2) cs,
      orders od
WHERE cs.custid = od.custid
GROUP BY cs.name;
```

NAME	TOTAL
박지성	39000
김연아	15000

서브 쿼리

3. 스칼라 서브쿼리(Scalar Subquery)

주로 SELECT 절에 위치하며 칼럼 대신에 사용하므로 반드시 하나의 값만을 반환해야 함.

```
-- 상품 리뷰 테이블에 없는 product_name 출력
SELECT a.product_code,
       (SELECT b.product_name
          FROM product b
         WHERE a.product_code = b.product_code) as product_name,
       a.member_id,
       a.content
FROM product_review a;
```

	PRODUCT_CODE	PRODUCT_NAME	MEMBER_ID	CONTENT
1	100001	무소음 무선 마우스	today123	무소음인데 소음이 조금 있는 듯
2	100001	무소음 무선 마우스	cloud100	무선이라 정말 편하네요
3	100002	기계식 게이밍 키보드	sky321	게임할 맛 납니다.

서브 쿼리

3. 스칼라 서브쿼리(Scalar Subquery)

주로 SELECT 절에 위치하며 칼럼 대신에 사용하므로 반드시 하나의 값만을 반환해야 하며 그렇지 않은 경우 에러를 발생함.

```
SELECT a.product_code,
       (SELECT b.product_name, b.price
          FROM product b
         WHERE a.product_code = b.product_code) as product_name,
       a.member_id,
       a.content
FROM product_review a;
```

```
ORA-00913: 값의 수가 너무 많습니다
00913. 00000 - "too many values"
*Cause:
*Action:
65행, 9열에서 오류 발생
```