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
-- 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

```sql
-- 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

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 or inner join)

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

```
-- 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)

```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
-- '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

조인(JOIN)

```sql
-- 고객별로 주문한 모든 도서의 총 판매액을 구하고, 고객별로 정렬하시오
SELECT customer.name, SUM(saleprice)
FROM customer, orders
WHERE customer.custid = orders.custid
GROUP BY customer.name
ORDER BY customer.name;
```

NAME	SUM(SALEP...
김연아	15000
류현진	33000
박지성	39000
안산	31000

조인(JOIN)

```sql
-- 고객의 이름과 주문한 도서의 이름을 검색하시오
SELECT customer.name, book.bookname
FROM customer, orders, book
WHERE customer.custid = orders.custid AND book.bookid = orders.bookid;
```

```sql
-- 가격이 20000원인 도서를 주문한 고객의 이름과 도서의 이름을 검색하시오
SELECT customer.name, book.bookname
FROM customer, orders, book
WHERE customer.custid = orders.custid
    AND book.bookid = orders.bookid
    AND book.price = 20000;
```

NAME	BOOKNAME
류현진	야구의 추억

조인(JOIN)

외부 조인(equi join or inner join)

양쪽 테이블에서 데이터 값이 일치하지 않는 경우에도 모든 데이터를 연결하기

```
-- 도서를 구매하지 않은 고객을 포함하여
-- 고객의 이름과 고객이 주문한 도서의 판매가격을 구하시오
SELECT customer.name, orders.saleprice
FROM customer LEFT OUTER JOIN orders
        ON 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
손흥민	(null)

서브 쿼리

서브 쿼리(Sub-Query)란

부속 질의는 하나의 SQL문아네 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

```sql
-- 도서를 구매한 적이 있는 고객의 이름을 검색하시오
SELECT name
FROM customer
WHERE custid IN(SELECT custid FROM orders);
```

NAME
박지성
김연아
안산
류현진

```sql
-- '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
-- 출판사별로 출판사의 평균 도서 가격보다 비싼 도서를 검색하시오
-- 튜플 변수 : 테이블 이름의 별칭
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. 인라인 뷰 – FROM 부속질의

인라인 뷰는 FROM 절에서 사용되는 부속질의를 말한다.

뷰는 기존 테이블로부터 일시적으로 만들어진 가상의 데이블이다.

```
-- 고객 번호가 2이하인 고객의 판매액을 검색하시오(고객이름과 고객별 판매액 출력)
SELECT cs.name, SUM(od.saleprice) AS total
FROM (SELECT custid, name
      FROM customer
      WHERE custid <= 2) cs,
      orders od
WHERE cs.custid = od.custid
GROUP BY cs.name;
```

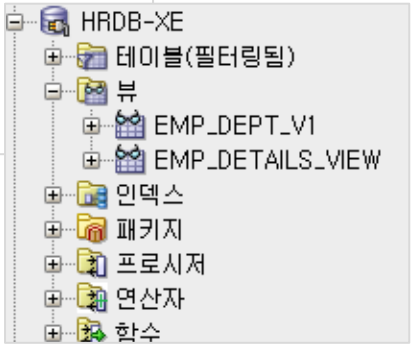
NAME	TOTAL
박지성	39000
김연아	15000

뷰(view)

뷰(VIEW)

뷰는 하나 이상의 테이블이나 다른 뷰의 데이터를 볼 수 있게 하는 데이터베이스 객체다. 테이블이 아닌 뷰를 사용하는 이유는 원본 테이블의 데이터를 안전하게 유지하면서 필요한 사용자에게 적절한 데이터를 제공할 수 있다.

```
-- 해당 사원이 속한 부서명 검색 --
SELECT a.employee_id, a.last_name, a.department_id,
       b.department_name
FROM employees a,
     departments b
WHERE a.department_id = b.department_id;
```



EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
100	King	90	Executive
101	Kochhar	90	Executive
102	De Haan	90	Executive
103	Hunold	60	IT
104	Ernst	60	IT
105	Austin	60	IT

뷰(view)

뷰(VIEW) 생성 및 삭제

CREATE VIEW 뷰이름 AS SELECT 문장;

```
-- VIEW 생성
CREATE VIEW emp_dept_v1 AS
SELECT a.employee_id, a.last_name, a.department_id,
       b.department_name
FROM employees a,
     departments b
WHERE a.department_id = b.department_id;
```

DROP VIEW 뷰이름

뷰(VIEW) 검색

```
21  SELECT *
22  FROM emp_dept_v1;
```

스크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.005초)

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	100	King	90	Executive
2	101	Kochhar	90	Executive
3	102	De Haan	90	Executive
4	103	Hunold	60	IT
5	104	Ernst	60	IT

뷰(view)

뷰(VIEW) 검색

```
--view 검색
SELECT * FROM emp_details_view;

-- employees 테이블과 emp_details_view를 조인하여 employee_id가 100인 직원의
-- employee_id, hire_date, department_name, job_title 출력
SELECT A.employee_id, A.hire_date, B.department_name, B.job_title
FROM employees A, emp_details_view B
WHERE A.employee_id = B.employee_id
AND A.employee_id = 100;
```

	EMPLOYEE_ID	HIRE_DATE	DEPARTMENT_NAME	JOB_TITLE
1	100	2003/06/17	Executive	President