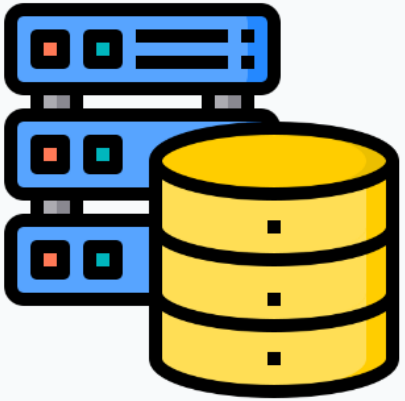
2장. SQL – DML

데이터 베이스 구성요소



도서 테이블 만들기

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000
2	축구아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프바이블	대한미디어	35000
5	피겨교본	굿스포츠	8000
6	양궁의 실제	굿스포츠	6000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

도서 테이블 만들기

```
-- book 테이블 생성
CREATE TABLE book(
    bookid      NUMBER PRIMARY KEY,
    bookname    VARCHAR2(40),
    publisher   VARCHAR2(40),
    price       NUMBER
);
```

```
-- book 자료 삽입
INSERT INTO book VALUES (1, '축구의 역사', '굿스포츠', 7000);
INSERT INTO book VALUES (2, '축구아는 여자', '나무수', 13000);
INSERT INTO book VALUES (3, '축구의 이해', '대한미디어', 22000);
INSERT INTO book VALUES (4, '골프 바이블', '대한미디어', 35000);
INSERT INTO book VALUES (5, '피겨 교본', '굿스포츠', 8000);
INSERT INTO book VALUES (6, '양궁의 실제', '굿스포츠', 6000);
INSERT INTO book VALUES (7, '야구의 추억', '이상미디어', 20000);
INSERT INTO book VALUES (8, '야구를 부탁해', '이상미디어', 13000);
INSERT INTO book VALUES (9, '올림픽 이야기', '삼성당', 7500);
INSERT INTO book VALUES (10, 'Olympic Champions', 'Pearson', 13000);
```

SELECT – 자료 검색

SELECT 문

SELECT 열이름 (or 별칭) **FROM** 테이블 이름

```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;
```

SELECT – 자료 검색

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

-- 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');
```

SELECT – 자료 검색

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

-- '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;
```

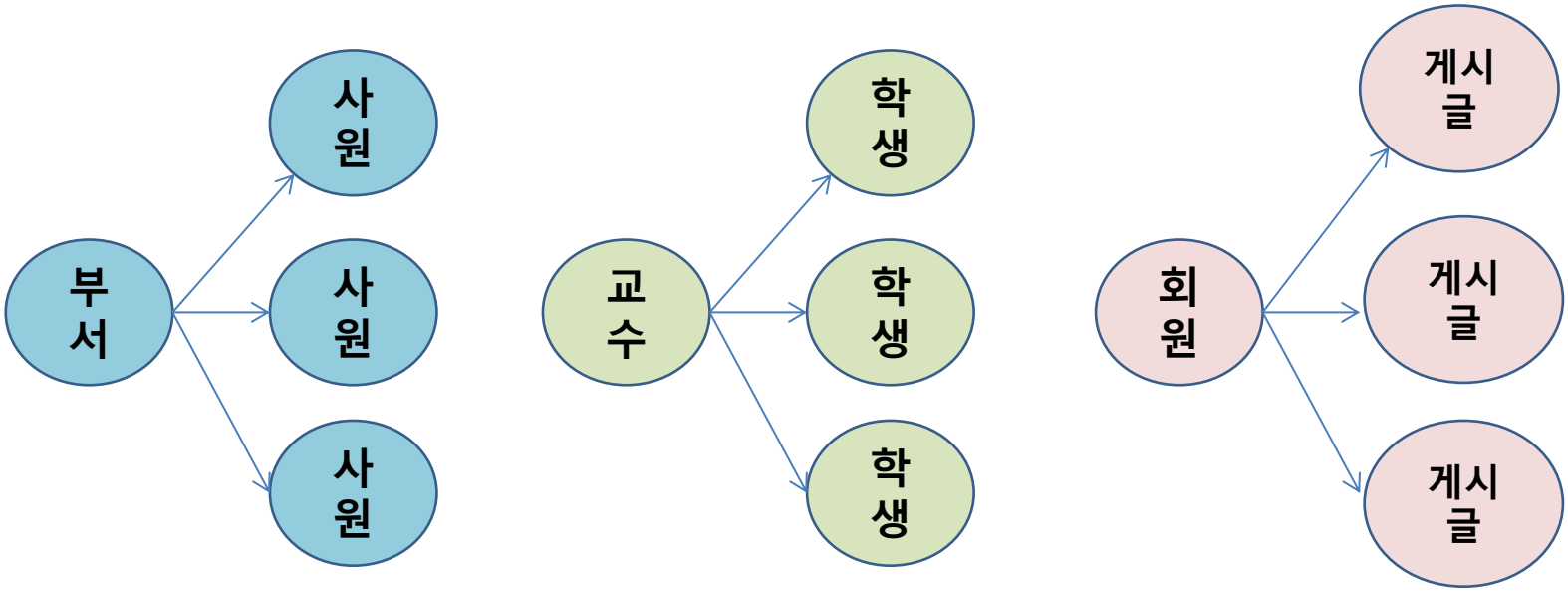
SELECT – 자료 검색

```sql
-- 출판사가 '굿스포츠' 혹은 '대한미디어'인 도서를 검색하시오
SELECT *
FROM book
WHERE publisher = '굿스포츠' OR publisher = '대한미디어';

-- 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;
```

관계(Releation)

엔티티(Entity) 관계(releation)

- 1대 多의 관계, 1대 1관계, 多 대 多 관계



관계(Releation)

외래키(FK : Foreign Key)

- 특정 테이블에 포함되어 있으면서 다른 테이블의 기본키로 지정된 키

기본키

부서 코드	부서 이름	위치
10	전산팀	서울
20	총무팀	인천

기본키

외래키

사원 번호	사원 이름	나이	부서코드
501	이강	23	10
502	김산	24	20
503	남한강	31	10
504	북한강	35	20

관계(Releation) – 외래키

부서와 직원 테이블 생성

- 사원(employee) 테이블에 Foreign Key 설정

```
-- 부서 테이블
CREATE TABLE department(
    deptid      NUMBER,
    deptname    VARCHAR2(20) NOT NULL,
    location    VARCHAR2(20) NOT NULL,
    PRIMARY KEY(deptid)
);

CREATE TABLE employee(
    empid       NUMBER,
    empname     VARCHAR2(20) NOT NULL,
    age         NUMBER,
    deptid      NUMBER,
    CONSTRAINT EMP_FK FOREIGN KEY(deptid) REFERENCES department(deptid)
);
```

관계(Releation) – 외래키

부서와 직원 자료 추가

```
-- 부서 자료 추가
INSERT INTO department VALUES (10, '전산팀', '서울');
INSERT INTO department VALUES (20, '총무팀', '인천');

-- 사원 자료 추가
INSERT INTO employee VALUES (101, '이강', 27, 10);
INSERT INTO employee VALUES (102, '김산', 28, 20);
INSERT INTO employee VALUES (103, '정들', 35, 30); -- 부서코드 없음
```

```
명령의 24 행에서 시작하는 중 오류 발생 -
INSERT INTO employee VALUES (103, '정들', 35, 30)
오류 보고 -
ORA-02291: 무결성 제약조건(SYSTEM.SYS_C008344)이 위배되었습니다- 부모 키가 없습니다
```

관계(Releation) – 외래키

부서 자료 삭제

```
-- 부서 삭제
DELETE FROM department WHERE deptid = 20; -- employee 테이블에서 참조하고 있어 오류.
```

```
명령의 27 행에서 시작하는 중 오류 발생 -
DELETE FROM department WHERE deptid = 20
오류 보고 -
ORA-02292: 무결성 제약조건(SYSTEM.SYS_C008344)이 위배되었습니다- 자식 레코드가 발견되었습니다
```

외래키 제약 조건 삭제

```
ALTER TABLE employee DROP CONSTRAINT EMP_FK;
```

테이블 삭제

```
-- 테이블 삭제 - CONSTRAINT가 설정되어 있는 경우
DROP TABLE department CASCADE CONSTRAINTS;
```

고객 테이블 만들기

custid	name	address	phone
1	박지성	영국 맨체스터	000-5000-0001
2	김연아	대한민국 서울	000-6000-0001
3	안산	대한민국 광주광역시	000-7000-0001
4	류현진	미국 토론토	NULL
5	손흥민	영국 토트넘	000-8000-0001

주문 테이블 만들기

orderid	custid	bookid	saleprice	orderdate
1	1	1	6000	2018-07-01
2	1	3	21000	2018-07-03
3	2	5	8000	2018-07-03
4	3	6	6000	2018-07-04
5	4	7	20000	2018-07-05
6	1	2	12000	2018-07-07
7	4	8	13000	2018-07-07
8	3	10	12000	2018-07-08
9	2	10	7000	2018-07-09
10	3	8	13000	2018-07-10

고객 및 주문 테이블 만들기

```
CREATE TABLE customer(
    custid      NUMBER PRIMARY KEY,
    name        VARCHAR2(40),
    address     VARCHAR2(50),
    phone       VARCHAR2(20)
);

CREATE TABLE orders(
    orderid     NUMBER PRIMARY KEY,
    custid      NUMBER,
    bookid      NUMBER,
    saleprice   NUMBER,
    orderdate   DATE,
    FOREIGN KEY(custid) REFERENCES customer(custid),   --외래키(custid)
    FOREIGN KEY(bookid) REFERENCES book(bookid)        --외래키(bookid)
);
```

고객 및 주문 테이블 만들기

```
-- customer 자료 삽입
INSERT INTO customer VALUES (1, '박지성', '영국 맨체스타', '000-5000-0001');
INSERT INTO customer VALUES (2, '김연아', '대한민국 서울', '000-6000-0001');
INSERT INTO customer VALUES (3, '안산', '대한민국 광주광역시', '000-7000-0001');
INSERT INTO customer VALUES (4, '류현진', '미국 토론토', NULL);
INSERT INTO customer VALUES (5, '손흥민', '영국 토트넘', '000-8000-0001');

-- orders 자료 삽입
INSERT INTO orders VALUES (1, 1, 1, 6000, TO_DATE('2018-07-01', 'YYYY-MM-DD'));
INSERT INTO orders VALUES (2, 1, 3, 21000, TO_DATE('2018-07-03', 'YYYY-MM-DD'));
INSERT INTO orders VALUES (3, 2, 5, 8000, TO_DATE('2018-07-03', 'YYYY-MM-DD'));
INSERT INTO orders VALUES (4, 3, 6, 6000, TO_DATE('2018-07-04', 'YYYY-MM-DD'));
INSERT INTO orders VALUES (5, 4, 7, 20000, TO_DATE('2018-07-05', 'YYYY-MM-DD'));
INSERT INTO orders VALUES (6, 1, 2, 12000, TO_DATE('2018-07-07', 'YYYY-MM-DD'));
INSERT INTO orders VALUES (7, 4, 8, 13000, TO_DATE('2018-07-07', 'YYYY-MM-DD'));
INSERT INTO orders VALUES (8, 3, 10, 12000, TO_DATE('2018-07-08', 'YYYY-MM-DD'));
INSERT INTO orders VALUES (9, 2, 10, 7000, TO_DATE('2018-07-09', 'YYYY-MM-DD'));
INSERT INTO orders VALUES (10, 3, 8, 13000, TO_DATE('2018-07-10', 'YYYY-MM-DD'));
```

고객 및 주문 테이블 만들기

```sql
-- 집계 함수와 GROUP BY
-- 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;
```

고객 및 주문 테이블 만들기

```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;
```

조인

```sql
-- 2개 이상의 테이블에서 sql 질의

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

-- 고객과 고객의 주문에 관한 데이터를 고객별로 정렬하시오
SELECT *
FROM customer, orders
WHERE customer.custid = orders.custid
ORDER BY customer.custid;

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

조인

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

-- 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;

-- 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;
```