08

CHAPTER

# 조인과 SQL 프로그래밍

# Contents

# 1-1 조인의 개요

- 조인(JOIN)
  - 2개 이상의 테이블을 묶어서 하나의 결과 테이블을 만드는 것
- cookDB의 구조


회원 테이블(userTBL)

| 아이디 | 이름 | 생년 | 지역 | 국번 | 전화번호 | 키 | 가입일 |
|---|---|---|---|---|---|---|---|
| YJS | 유재석 | 1972 | 서울 | 010 | 11111111 | 178 | 2008.8.8 |
| KHD | 강호동 | 1970 | 경북 | 011 | 22222222 | 182 | 2007.7.7 |
| KKJ | 김국진 | 1965 | 서울 | 019 | 33333333 | 171 | 2009.9.9 |
| KYM | 김용만 | 1967 | 서울 | 010 | 44444444 | 177 | 2015.5.5 |
| KJD | 김제동 | 1974 | 경남 | | | 173 | 2013.3.3 |
| NHS | 남희석 | 1971 | 충남 | 016 | 66666666 | 180 | 2017.4.4 |
| SDY | 신동엽 | 1971 | 경기 | | | 176 | 2008.10.10 |
| LHJ | 이휘재 | 1972 | 경기 | 011 | 88888888 | 180 | 2006.4.4 |
| LKK | 이경규 | 1960 | 경남 | 018 | 99999999 | 170 | 2004.12.12 |
| PSH | 박수홍 | 1970 | 서울 | 010 | 00000000 | 183 | 2012.5.5 |

PK

구매 테이블(buyTBL)

| 순번 | 아이디 | 물품 | 분류 | 단가 | 수량 |
|---|---|---|---|---|---|
| 1 | KHD | 운동화 | | 30 | 2 |
| 2 | KHD | 노트북 | 전자 | 1000 | 1 |
| 3 | KYM | 모니터 | 전자 | 200 | 1 |
| 4 | PSH | 모니터 | 전자 | 200 | 5 |
| 5 | KHD | 청바지 | 의류 | 50 | 3 |
| 6 | PSH | 메모리 | 전자 | 80 | 10 |
| 7 | KJD | 책 | 서적 | 15 | 5 |
| 8 | LHJ | 책 | 서적 | 15 | 2 |
| 9 | LHJ | 청바지 | 의류 | 50 | 1 |
| 10 | PSH | 운동화 | | 30 | 2 |
| 11 | LHJ | 책 | 서적 | 15 | 1 |
| 12 | PSH | 운동화 | | 30 | 2 |

PK FK

그림 8-1 cookDB 데이터베이스

- 일대다 관계
  - cookDB의 회원 테이블(userTBL)과 구매 테이블(buyTBL)이 맺고 있는 관계
  - 한쪽 테이블에는 하나의 값만 존재하고 그 값과 대응되는 다른 쪽 테이블의 값은 여러 개
  - 기업의 직원 테이블과 급여 테이블, 학교의 학생 테이블과 학점 테이블

# 1-2 내부 조인

- 내부 조인의 형식

```
SELECT <열 목록>
   FROM <첫 번째 테이블>
   INNER JOIN <두 번째 테이블>
      ON <조인될 조건>
[WHERE 검색조건];
```

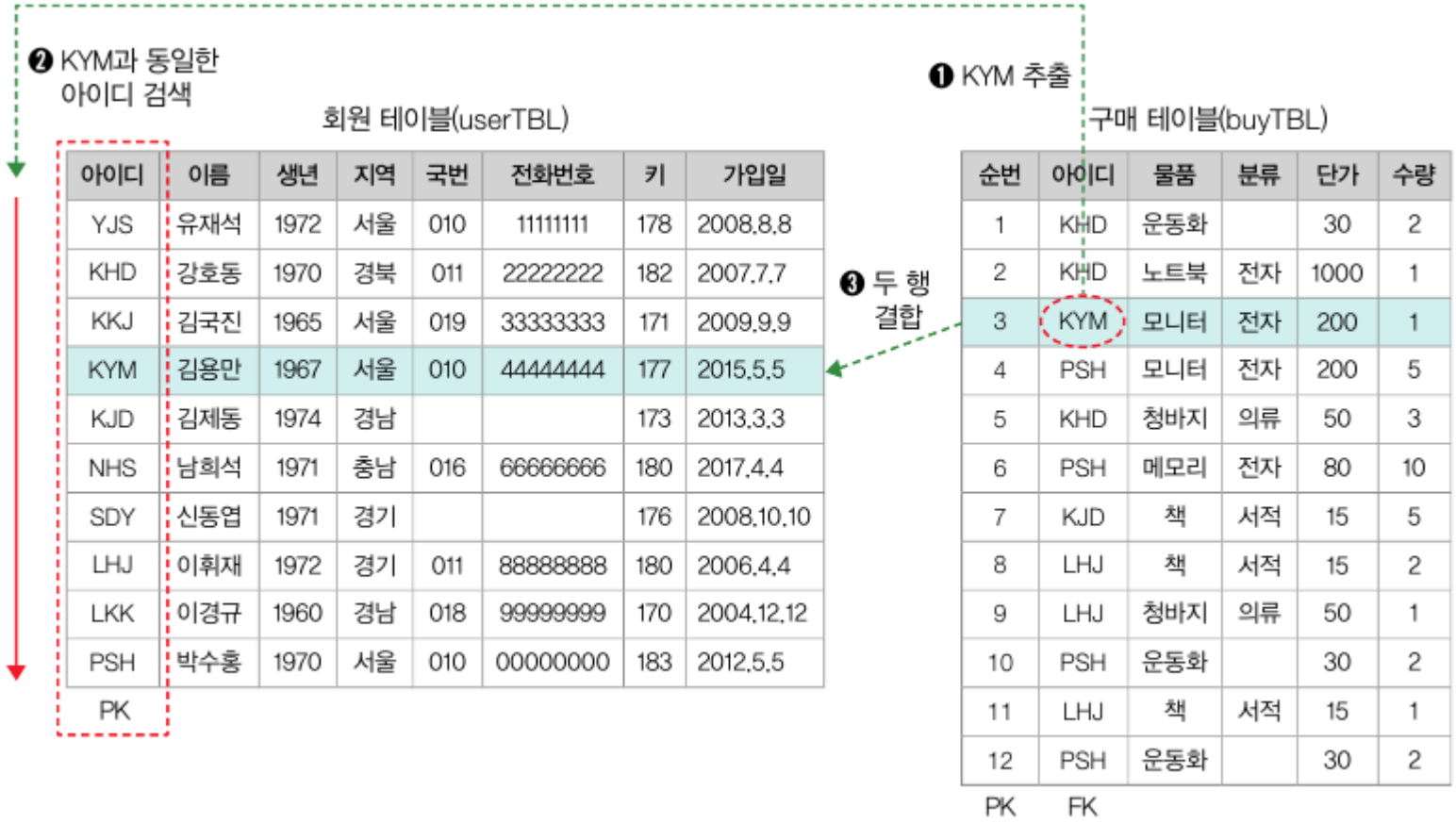
- KYM이라는 아이디를 가진 회원이 구매한 물건을 발송하려면?

```
USE cookDB;
SELECT *
   FROM buyTBL
     INNER JOIN userTBL
        ON buyTBL.userID = userTBL.userID
   WHERE buyTBL.userID = 'KYM';
```

| num | userID | prodName | groupName | price | amount | userID | userName | birthYear | addr | mobile1 | mobile2 | height | mDate |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | KYM | 모니터 | 전자 | 200 | 1 | KYM | 김용만 | 1967 | 서울 | 010 | 44444444 | 177 | 2015-05-05 |

# 1-2 내부 조인

- 내부 조인의 작동
  - 구매 테이블의 회원 아이디(buyTBL.userID)인 KYM을 추출
  - KYM과 동일한 값을 회원 테이블의 아이디 (userTBL.userID) 열에서 검색
  - 아이디 KYM을 찾으면 구매 테이블과 회원 테이블의 두 행을 결합(조인)




회원 테이블(userTBL)

| 아이디 | 이름 | 생년 | 지역 | 국번 | 전화번호 | 키 | 가입일 |
|---|---|---|---|---|---|---|---|
| YJS | 유재석 | 1972 | 서울 | 010 | 11111111 | 178 | 2008.8.8 |
| KHD | 강호동 | 1970 | 경북 | 011 | 22222222 | 182 | 2007.7.7 |
| KKJ | 김국진 | 1965 | 서울 | 019 | 33333333 | 171 | 2009.9.9 |
| KYM | 김용만 | 1967 | 서울 | 010 | 44444444 | 177 | 2015.5.5 |
| KJD | 김제동 | 1974 | 경남 | | | 173 | 2013.3.3 |
| NHS | 남희석 | 1971 | 충남 | 016 | 66666666 | 180 | 2017.4.4 |
| SDY | 신동엽 | 1971 | 경기 | | | 176 | 2008.10.10 |
| LHJ | 이휘재 | 1972 | 경기 | 011 | 88888888 | 180 | 2006.4.4 |
| LKK | 이경규 | 1960 | 경남 | 018 | 99999999 | 170 | 2004.12.12 |
| PSH | 박수홍 | 1970 | 서울 | 010 | 00000000 | 183 | 2012.5.5 |
| PK | | | | | | | |

구매 테이블(buyTBL)

| 순번 | 아이디 | 물품 | 분류 | 단가 | 수량 |
|---|---|---|---|---|---|
| 1 | KHD | 운동화 | | 30 | 2 |
| 2 | KHD | 노트북 | 전자 | 1000 | 1 |
| 3 | KYM | 모니터 | 전자 | 200 | 1 |
| 4 | PSH | 모니터 | 전자 | 200 | 5 |
| 5 | KHD | 청바지 | 의류 | 50 | 3 |
| 6 | PSH | 메모리 | 전자 | 80 | 10 |
| 7 | KJD | 책 | 서적 | 15 | 5 |
| 8 | LHJ | 책 | 서적 | 15 | 2 |
| 9 | LHJ | 청바지 | 의류 | 50 | 1 |
| 10 | PSH | 운동화 | | 30 | 2 |
| 11 | LHJ | 책 | 서적 | 15 | 1 |
| 12 | PSH | 운동화 | | 30 | 2 |
| PK | FK | | | | |

그림 8-3 내부 조인의 작동

# 1-2 내부 조인

- 만약 WHERE buyTBL.userID = 'KYM'을 생략하면?

| num | userID | prodName | groupName | price | amount | userID | userName | birthYear | addr | mobile1 | mobile2 | height | mDate |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | KHD | 운동화 | NULL | 30 | 2 | KHD | 강호동 | 1970 | 경북 | 011 | 22222222 | 182 | 2007-07-07 |
| 2 | KHD | 노트북 | 전자 | 1000 | 1 | KHD | 강호동 | 1970 | 경북 | 011 | 22222222 | 182 | 2007-07-07 |
| 5 | KHD | 청바지 | 의류 | 50 | 3 | KHD | 강호동 | 1970 | 경북 | 011 | 22222222 | 182 | 2007-07-07 |
| 7 | KJD | 책 | 서적 | 15 | 5 | KJD | 김제동 | 1974 | 경남 | NULL | NULL | 173 | 2013-03-03 |
| 3 | KYM | 모니터 | 전자 | 200 | 1 | KYM | 김용만 | 1967 | 서울 | 010 | 44444444 | 177 | 2015-05-05 |
| 8 | LHJ | 책 | 서적 | 15 | 2 | LHJ | 이휘재 | 1972 | 경기 | 011 | 88888888 | 180 | 2006-04-04 |
| 9 | LHJ | 청바지 | 의류 | 50 | 1 | LHJ | 이휘재 | 1972 | 경기 | 011 | 88888888 | 180 | 2006-04-04 |
| 11 | LHJ | 책 | 서적 | 15 | 1 | LHJ | 이휘재 | 1972 | 경기 | 011 | 88888888 | 180 | 2006-04-04 |
| 4 | PSH | 모니터 | 전자 | 200 | 5 | PSH | 박수홍 | 1970 | 서울 | 010 | 00000000 | 183 | 2012-05-05 |
| 6 | PSH | 메모리 | 전자 | 80 | 10 | PSH | 박수홍 | 1970 | 서울 | 010 | 00000000 | 183 | 2012-05-05 |
| 10 | PSH | 운동화 | NULL | 30 | 2 | PSH | 박수홍 | 1970 | 서울 | 010 | 00000000 | 183 | 2012-05-05 |
| 12 | PSH | 운동화 | NULL | 30 | 2 | PSH | 박수홍 | 1970 | 서울 | 010 | 00000000 | 183 | 2012-05-05 |

그림 8-4 내부 조인 결과 2

# 1-2 내부 조인

- 아이디, 이름, 구매 물품, 주소, 연락처만 추출

```
SELECT userID, userName, prodName, addr, CONCAT(mobile1, mobile2) AS '연락처'
   FROM buyTBL
      INNER JOIN userTBL
         ON buyTBL.userID = userTBL.userID;
```

**실행 결과**

Error Code: 1052. Column 'userid' in field list is ambiguous

- 어느 테이블의 userID를 추출할지 선택(구매 테이블의 userID라고 명시)

```
SELECT buyTBL.userID, userName,
prodName, addr, CONCAT(mobile1, mobile2)
AS '연락처'
   FROM buyTBL
      INNER JOIN userTBL
         ON buyTBL.userID = userTBL.userID;
```

| userID | userName | prodName | addr | 연락처 |
|---|---|---|---|---|
| KHD | 강호동 | 운동화 | 경북 | 01122222222 |
| KHD | 강호동 | 노트북 | 경북 | 01122222222 |
| KHD | 강호동 | 청바지 | 경북 | 01122222222 |
| KJD | 김제동 | 책 | 경남 | NULL |
| KYM | 김용만 | 모니터 | 서울 | 01044444444 |
| LHJ | 이휘재 | 책 | 경기 | 01188888888 |
| LHJ | 이휘재 | 청바지 | 경기 | 01188888888 |
| LHJ | 이휘재 | 책 | 경기 | 01188888888 |
| PSH | 박수홍 | 모니터 | 서울 | 01000000000 |
| PSH | 박수홍 | 메모리 | 서울 | 01000000000 |
| PSH | 박수홍 | 운동화 | 서울 | 01000000000 |
| PSH | 박수홍 | 운동화 | 서울 | 01000000000 |

# 1-2 내부 조인

- SELECT 다음의 열 이름도 모두 '테이블이름.열이름' 형식으로 작성한 예

```
SELECT buyTBL.userID, userTBL.userName, buyTBL.prodName, userTBL.addr,
          CONCAT(userTBL.mobile1, userTBL.mobile2) AS '연락처'
   FROM buyTBL
      INNER JOIN userTBL
         ON buyTBL.userID = userTBL.userID;
```

- 각 테이블에 별칭 부여

```
SELECT B.userID, U.userName, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2) AS '연락처'
   FROM buyTBL B
      INNER JOIN userTBL U
         ON B.userID = U.userID;
```

# 1-2 내부 조인

- KYM이라는 아이디를 가진 회원이 구매한 물건과 회원 정보 조인(아이디, 이름, 물품, 주소, 연락처만 출력되게 하고 코드도 간결하게 수정)

```
SELECT B.userID, U.userName, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2) AS '연락처'
   FROM buyTBL B
      INNER JOIN userTBL U
         ON B.userID = U.userID
   WHERE B.userID = 'KYM';
```

| | userID | userName | prodName | addr | 연락처 |
|---|---|---|---|---|---|
| | KYM | 김용만 | 모니터 | 서울 | 01044444444 |

- 회원 테이블(userTBL)을 기준으로 KYM이 구매한 물건의 목록 조회

```
SELECT U.userID, U.userName, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2) AS '연락처'
   FROM userTBL U
      INNER JOIN buyTBL B
         ON U.userID = B.userID
   WHERE B.userID = 'KYM';
```

# 1-2 내부 조인

- 전체 회원이 구매한 목록 모두 출력(회원 아이디 순으로 정렬)

```
SELECT U.userID, U.userName, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2) AS '연락처'
   FROM userTBL U
      INNER JOIN buyTBL B
         ON U.userID = B.userID
   ORDER BY U.userID;
```

| userID | userName | prodName | addr | 연락처 |
|---|---|---|---|---|
| KHD | 강호동 | 운동화 | 경북 | 01122222222 |
| KHD | 강호동 | 노트북 | 경북 | 01122222222 |
| KHD | 강호동 | 청바지 | 경북 | 01122222222 |
| KJD | 김제동 | 책 | 경남 | NULL |
| KYM | 김용만 | 모니터 | 서울 | 01044444444 |
| LHJ | 이휘재 | 책 | 경기 | 01188888888 |
| LHJ | 이휘재 | 청바지 | 경기 | 01188888888 |
| LHJ | 이휘재 | 책 | 경기 | 01188888888 |
| PSH | 박수홍 | 모니터 | 서울 | 01000000000 |
| PSH | 박수홍 | 메모리 | 서울 | 01000000000 |
| PSH | 박수홍 | 운동화 | 서울 | 01000000000 |
| PSH | 박수홍 | 운동화 | 서울 | 01000000000 |

# 1-2 내부 조인

- 쇼핑몰에서 한 번이라도 구매한 기록이 있는 우수 회원에게 감사의 안내문을 발송하려 할 때 (DISTINCT 키워드 활용)

```
SELECT DISTINCT U.userID, U.userName, U.addr
   FROM userTBL U
      INNER JOIN buyTBL B
         ON U.userID = B.userID
   ORDER BY U.userID;
```

| userID | userName | addr |
|---|---|---|
| KHD | 강호동 | 경북 |
| KJD | 김제동 | 경남 |
| KYM | 김용만 | 서울 |
| LHJ | 이휘재 | 경기 |
| PSH | 박수홍 | 서울 |

- EXISTS 구문을 사용하면 앞과 동일한 결과를 만들 수 있음

```
SELECT U.userID, U.userName, U.addr
   FROM userTBL U
   WHERE EXISTS (
      SELECT *
      FROM buyTBL B
      WHERE U.userID = B.userID );
```

# 1-2 내부 조인

- 다대다 관계
  - 한 학생은 여러 개의 동아리에 가입해서 활동할 수 있고, 하나의 동아리에는 여러 학생이 가입할 수 있음
  - 따라서 학생 테이블과 동아리 테이블은 ' 다대다(many - to - many)' 관계
  - 다대다 관계는 연결 테이블과 두 테이블이 일대다 관계를 맺도록 구성

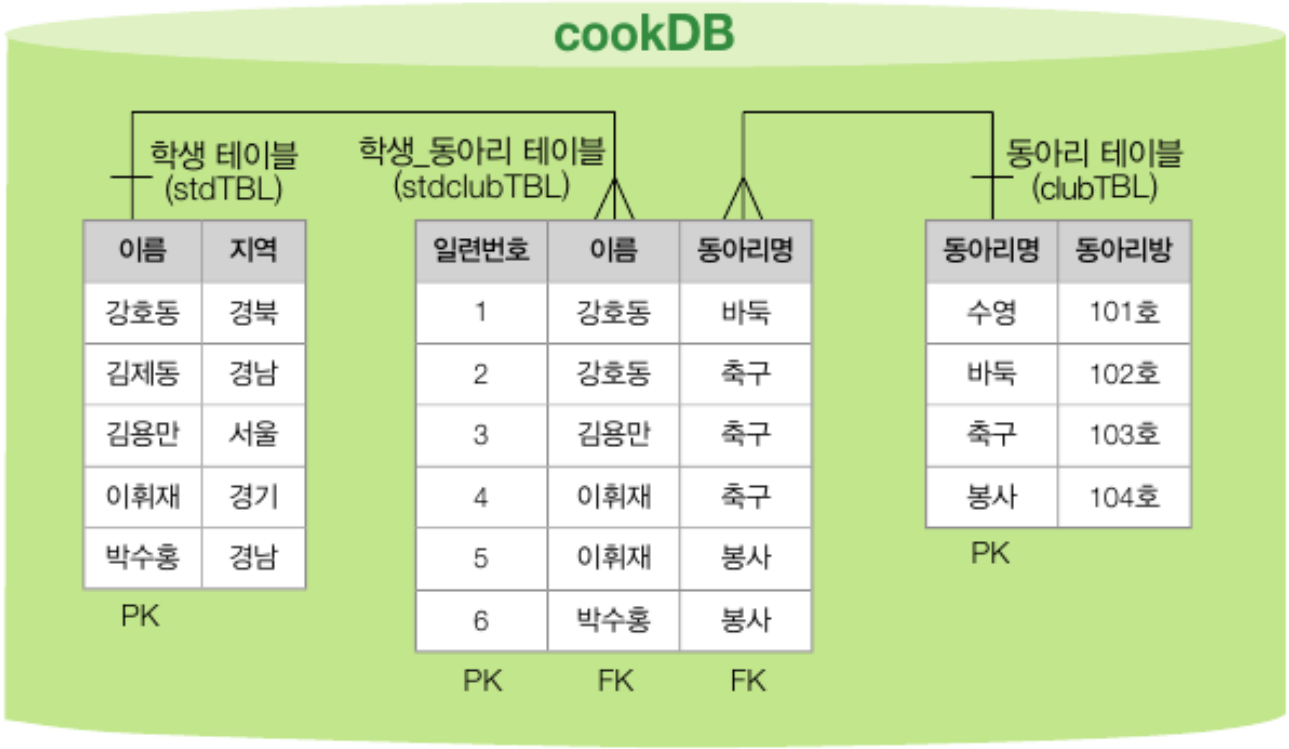



그림 8-9 3개 테이블 샘플

# [실습 8-1] 3개 테이블 내부 조인하기

교재 264~266p 참고

1 테이블 만들기

1-1 학생 테이블, 동아리 테이블, 학생_동아리 테이블 생성

```
USE cookDB;
CREATE TABLE stdTBL
( stdName VARCHAR(10) NOT NULL PRIMARY KEY,
  addr CHAR(4) NOT NULL
);
CREATE TABLE clubTBL
( clubName VARCHAR(10) NOT NULL PRIMARY KEY,
  roomNo CHAR(4) NOT NULL
);
CREATE TABLE stdclubTBL
( num int AUTO_INCREMENT NOT NULL PRIMARY KEY,
  stdName VARCHAR(10) NOT NULL,
  clubName VARCHAR(10) NOT NULL,
  FOREIGN KEY(stdName) REFERENCES stdTBL(stdName),
  FOREIGN KEY(clubName) REFERENCES clubTBL(clubName)
);
INSERT INTO stdTBL VALUES ('강호동', '경북'), ('김제동', '경남'), ('김용만', '서울'), ('이휘 재', '경기'), ('박수홍', '서울');
INSERT INTO clubTBL VALUES ('수영', '101호'), ('바둑', '102호'), ('축구', '103호'), ('봉사', '104호');
INSERT INTO stdclubTBL VALUES (NULL, '강호동', '바둑'), (NULL, '강호동', '축구'), (NULL, '김용만', '축구'), (NULL, '이휘재', '축구'), (NULL, '이휘재', '봉사'), (NULL, '박수홍', '봉사');
```

# [실습 8-1] 3개 테이블 내부 조인하기

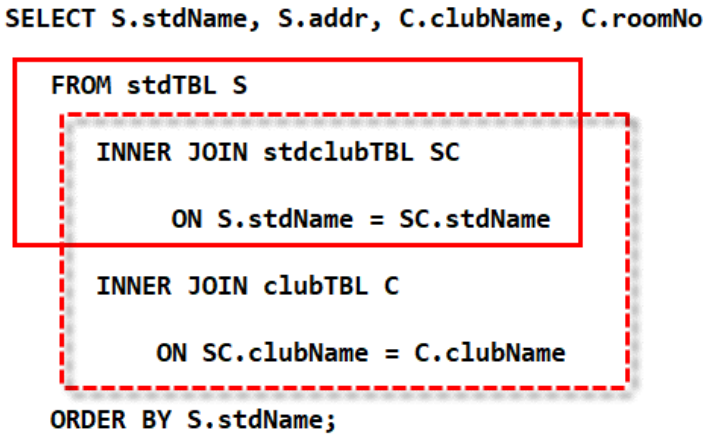
교재 264~266p 참고

2 조인하기

2-1 학생_동아리 테이블과 학생 테이블의 일대다 관계를 내부 조인하고, 학생_동아리 테이블과 동아리 테이블의 일대다 관계를 내부 조인

```
SELECT S.stdName, S.addr, C.clubName, C.roomNo
   FROM stdTBL S
      INNER JOIN stdclubTBL SC
         ON S.stdName = SC.stdName
      INNER JOIN clubTBL C
         ON SC.clubName = C.clubName
   ORDER BY S.stdName;
```

| | stdName | addr | clubName | roomNo |
|---|---|---|---|---|
| ▶ | 강호동 | 경북 | 바둑 | 102호 |
| | 강호동 | 경북 | 축구 | 103호 |
| | 김용만 | 서울 | 축구 | 103호 |
| | 박수홍 | 서울 | 봉사 | 104호 |
| | 이휘재 | 경기 | 봉사 | 104호 |
| | 이휘재 | 경기 | 축구 | 103호 |



# [실습 8-1] 3개 테이블 내부 조인하기

교재 264~266p 참고

2-2 동아리를 기준으로 가입한 학생의 목록

```
SELECT C.clubName, C.roomNo, S.stdName, S.addr
    FROM stdTBL S
        INNER JOIN stdclubTBL SC
            ON SC.stdName = S.stdName
        INNER JOIN clubTBL C
            ON SC.clubName = C.clubName
    ORDER BY C.clubName;
```

| clubName | roomNo | stdName | addr |
|---|---|---|---|
| 바둑 | 102호 | 강호동 | 경북 |
| 봉사 | 104호 | 이휘재 | 경기 |
| 봉사 | 104호 | 박수홍 | 서울 |
| 축구 | 103호 | 강호동 | 경북 |
| 축구 | 103호 | 김용만 | 서울 |
| 축구 | 103호 | 이휘재 | 경기 |

# 1-3 외부 조인

- 외부 조인(outer join)
  - 조인 조건을 만족하지 않는 행까지 포함하여 출력하는 조인

- 외부 조인의 형식

```
SELECT <열 목록>
FROM <첫 번째 테이블(LEFT 테이블)>
   <LEFT | RIGHT> OUTER JOIN <두 번째 테이블(RIGHT 테이블)>
       ON <조인될 조건>
[WHERE 검색조건];
```

# 1-3 외부 조인

- 전체 회원의 구매 기록을 출력하되 구매 기록이 없는 회원도 출력

```
USE cookDB;
SELECT U.userID, U.userName, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2) AS '연락처'
   FROM userTBL U
      LEFT OUTER JOIN buyTBL B
         ON U.userID = B.userID
   ORDER BY U.userID;
```

| userID | userName | prodName | addr | 연락처 |
|---|---|---|---|---|
| KHD | 강호동 | 운동화 | 경북 | 01122222222 |
| KHD | 강호동 | 노트북 | 경북 | 01122222222 |
| KHD | 강호동 | 청바지 | 경북 | 01122222222 |
| KJD | 김제동 | 책 | 경남 | NULL |
| KKJ | 김국진 | NULL | 서울 | 01933333333 |
| KYM | 김용만 | 모니터 | 서울 | 01044444444 |
| LHJ | 이휘재 | 책 | 경기 | 01[illegible]88888 |
| PS[illegible] | 박수홍 | 청[illegible] | [illegible]기 | 0100000[illegible] |
| PSH | 박수홍 | 운동화 | 서울 | 01000000000 |
| SDY | 신동엽 | NULL | 경기 | NULL |
| YJS | 유재석 | NULL | 서울 | 01011111111 |

# 1-3 외부 조인

- 동일한 결과를 출력 하되 쿼리문에 RIGHT OUTER JOIN 사용

```
SELECT U.userID, U.userName, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2) AS '연락처'
   FROM buyTBL B
      RIGHT OUTER JOIN userTBL U
         ON U.userID = B.userID
   ORDER BY U.userID;
```

- 물건을 한 번도 구매한 적이 없는 회원의 목록 출력

```
SELECT U.userID, U.userName, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2) AS '연락처'
   FROM userTBL U
      LEFT OUTER JOIN buyTBL B
         ON U.userID = B.userID
   WHERE B.prodName IS NULL
   ORDER BY U.userID;
```

| userID | userName | prodName | addr | 연락처 |
|---|---|---|---|---|
| KKJ | 김국진 | NULL | 서울 | 01933333333 |
| LKK | 이경규 | NULL | 경남 | 01899999999 |
| NHS | 남희석 | NULL | 충남 | 01666666666 |
| SDY | 신동엽 | NULL | 경기 | NULL |
| YJS | 유재석 | NULL | 서울 | 01011111111 |

# [실습 8-2] 왼쪽/오른쪽 외부 조인하기

교재 268~269p 참고

1 왼쪽/오른쪽 외부 조인하기

1-1 외부 조인을 수행하여 동아리에 가입하지 않은 학생도 출력

```
USE cookDB;
SELECT S.stdName, S.addr, C.clubName, C.roomNo
   FROM stdTBL S
      LEFT OUTER JOIN stdclubTBL SC
         ON S.stdName = SC.stdName
      LEFT OUTER JOIN clubTBL C
         ON SC.clubName = C.clubName
   ORDER BY S.stdName;
```

| stdName | addr | clubName | roomNo |
|---|---|---|---|
| 강호동 | 경북 | 바둑 | 102호 |
| 강호동 | 경북 | 축구 | 103호 |
| 김용만 | 서울 | 축구 | 103호 |
| 김제동 | 경남 | NULL | NULL |
| 박수홍 | 서울 | 봉사 | 104호 |
| 이휘재 | 경기 | 축구 | 103호 |
| 이휘재 | 경기 | 봉사 | 104호 |

1-2 동아리를 기준으로 가입 학생을 출력하되, 가입 학생이 한 명도 없는 동아리도 출력

```
SELECT C.clubName, C.roomNo, S.stdName, S.addr
   FROM stdTBL S
      LEFT OUTER JOIN stdclubTBL SC
         ON SC.stdName = S.stdName
      RIGHT OUTER JOIN clubTBL C
         ON SC.clubName = C.clubName
   ORDER BY C.clubName;
```

| clubName | roomNo | stdName | addr |
|---|---|---|---|
| 바둑 | 102호 | 강호동 | 경북 |
| 봉사 | 104호 | 이휘재 | 경기 |
| 봉사 | 104호 | 박수홍 | 서울 |
| 수영 | 101호 | NULL | NULL |
| 축구 | 103호 | 강호동 | 경북 |
| 축구 | 103호 | 김용만 | 서울 |
| 축구 | 103호 | 이휘재 | 경기 |

# [실습 8-2] 왼쪽/오른쪽 외부 조인하기

교재 268~269p 참고

2 완전 외부 조인을 한 것과 같은 효과 내기

2-1 동아리에 가입하지 않은 학생도 출력하고 학생이 한 명도 없는 동아리도 출력

```
SELECT S.stdName, S.addr, C.clubName, C.roomNo
   FROM stdTBL S
      LEFT OUTER JOIN stdclubTBL SC
         ON S.stdName = SC.stdName
      LEFT OUTER JOIN clubTBL C
         ON SC.clubName = C.clubName
UNION
SELECT S.stdName, S.addr, C.clubName, C.roomNo
   FROM stdTBL S
      LEFT OUTER JOIN stdclubTBL SC
         ON SC.stdName = S.stdName
      RIGHT OUTER JOIN clubTBL C
         ON SC.clubName = C.clubName;
```

| stdName | addr | clubName | roomNo |
|---|---|---|---|
| 강호동 | 경북 | 바둑 | 102호 |
| 강호동 | 경북 | 축구 | 103호 |
| 김용만 | 서울 | 축구 | 103호 |
| 김제동 | 경남 | NULL | NULL |
| 박수홍 | 서울 | 봉사 | 104호 |
| 이휘재 | 경기 | 축구 | 103호 |
| 이휘재 | 경기 | 봉사 | 104호 |
| NULL | NULL | 수영 | 101호 |

# 1-4 상호 조인

- 상호 조인(cross join)
  - 한쪽 테이블의 모든 행과 다른 쪽 테이블의 모든 행을 조인하는 것
  - 상호 조인 결과 테이블의 행수는 두 테이블의 행수를 곱한 값
  - 상호 조인은 카티션곱(cartesian product)이라고도 함

회원 테이블(userTBL)

| 아이디 | 이름 | 생년 | 지역 | 국번 | 전화번호 | 키 | 가입일 |
|---|---|---|---|---|---|---|---|
| YJS | 유재석 | 1972 | 서울 | 010 | 11111111 | 178 | 2008.8.8 |
| KHD | 강호동 | 1970 | 경북 | 011 | 22222222 | 182 | 2007.7.7 |
| KKJ | 김국진 | 1965 | 서울 | 019 | 33333333 | 171 | 2009.9.9 |
| KYM | 김용만 | 1967 | 서울 | 010 | 44444444 | 177 | 2015.5.5 |
| KJD | 김제동 | 1974 | 경남 | | | 173 | 2013.3.3 |
| NHS | 남희석 | 1971 | 충남 | 016 | 66666666 | 180 | 2017.4.4 |
| SDY | 신동엽 | 1971 | 경기 | | | 176 | 2008.10.10 |
| LHJ | 이휘재 | 1972 | 경기 | 011 | 88888888 | 180 | 2006.4.4 |
| LKK | 이경규 | 1960 | 경남 | 018 | 99999999 | 170 | 2004.12.12 |
| PSH | 박수홍 | 1970 | 서울 | 010 | 00000000 | 183 | 2012.5.5 |

PK

구매 테이블(buyTBL)

| 순번 | 아이디 | 물품 | 분류 | 단가 | 수량 |
|---|---|---|---|---|---|
| 1 | KHD | 운동화 | | 30 | 2 |
| 2 | KHD | 노트북 | 전자 | 1000 | 1 |
| 3 | KYM | 모니터 | 전자 | 200 | 1 |
| 4 | PSH | 모니터 | 전자 | 200 | 5 |
| 5 | KHD | 청바지 | 의류 | 50 | 3 |
| 6 | PSH | 메모리 | 전자 | 80 | 10 |
| 7 | KJD | 책 | 서적 | 15 | 5 |
| 8 | LHJ | 책 | 서적 | 15 | 2 |
| 9 | LHJ | 청바지 | 의류 | 50 | 1 |
| 10 | PSH | 운동화 | | 30 | 2 |
| 11 | LHJ | 책 | 서적 | 15 | 1 |
| 12 | PSH | 운동화 | | 30 | 2 |

PK FK

그림 8-18 상호 조인 방식

# 1-4 상호 조인

- 회원 테이블과 구매 테이블을 상호 조인

```
USE cookDB;
SELECT *
   FROM buyTBL
      CROSS JOIN userTBL;
```

- COUNT( * ) 함수로 상호 조인한 결과의 개수만 알아본 예

```
USE employees;
SELECT COUNT( * ) AS '데이터개수'
   FROM employees
      CROSS JOIN titles;
```

| | 데이터개수 |
|---|---|
| ▶ | 133003039392 |

# 1-5 자체 조인

- 자체 조인(self join)
  - 자기 자신과 자기 자신을 조인하는 것
  - 자체 조인을 활용하는 대표적인 예는 조직도와 관련된 테이블

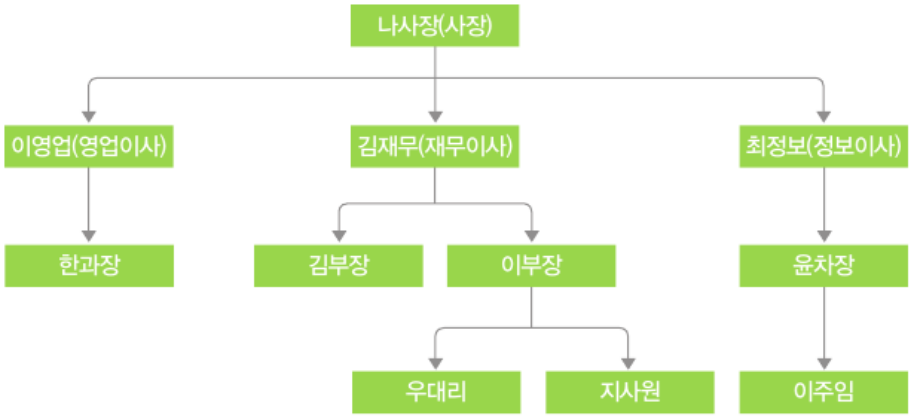



그림 8-20 간단한 조직도의 예

표 8-1 조직도 테이블

| 직원 이름(emp): 기본키 | 상관 이름(manager) | 구내 번호(empTel) |
|---|---|---|
| 나사장 | 없음(NULL) | 0000 |
| 김재무 | 나사장 | 2222 |
| 김부장 | 김재무 | 2222-1 |
| 이부장 | 김재무 | 2222-2 |
| 우대리 | 이부장 | 2222-2-1 |
| 지사원 | 이부장 | 2222-2-2 |
| 이영업 | 나사장 | 1111 |
| 한과장 | 이영업 | 1111-1 |
| 최정보 | 나사장 | 3333 |
| 윤차장 | 최정보 | 3333-1 |
| 이주임 | 윤차장 | 3333-1-1 |

- 우대리 상관의 구내번호를 알고 싶다면 ?

# [실습 8-3] 자체 조인하기

교재 272~273p 참고

1 테이블 만들기

1-1 조직도 테이블 만들기

```
USE cookDB;
CREATE TABLE empTBL(emp CHAR(3), manager CHAR(3), empTel VARCHAR(8));
```

1-2 조직도 테이블에 데이터 입력

```
INSERT INTO empTBL VALUES ('나사장', NULL, '0000');
INSERT INTO empTBL VALUES ('김재무', '나사장', '2222');
INSERT INTO empTBL VALUES ('김부장', '김재무', '2222-1');
INSERT INTO empTBL VALUES ('이부장', '김재무', '2222-2');
INSERT INTO empTBL VALUES ('우대리', '이부장', '2222-2-1');
INSERT INTO empTBL VALUES ('지사원', '이부장', '2222-2-2');
INSERT INTO empTBL VALUES ('이영업', '나사장', '1111');
INSERT INTO empTBL VALUES ('한과장', '이영업', '1111-1');
INSERT INTO empTBL VALUES ('최정보', '나사장', '3333');
INSERT INTO empTBL VALUES ('윤차장', '최정보', '3333-1');
INSERT INTO empTBL VALUES ('이주임', '윤차장', '3333-1-1');
```

# [실습 8-3] 자체 조인하기

교재 272~273p 참고

2 자체 조인하기

2-1 우대리 상관의 구내 번호 확인

```
SELECT A.emp AS '부하직원', B.emp AS '직속상관', B.empTel AS '직속상관연락처'
   FROM empTBL A
      INNER JOIN empTBL B
         ON A.manager = B.emp
   WHERE A.emp = '우대리';
```

| | 부하직원 | 직속상관 | 직속상관연락처 |
|---|---|---|---|
| ▶ | 우대리 | 이부장 | 2222-2 |

# 1-6 UNION/UNION ALL

- UNION
  - 두 쿼리의 결과를 행으로 합치는 연산자

SELECT stdName, addr FROM stdTBL

| stdName | addr |
|---|---|
| 강호동 | 경북 |
| 김제동 | 경남 |
| 김용만 | 서울 |
| 이휘재 | 경기 |
| 박수홍 | 서울 |

SELECT clubName, roomNo FROM clubTBL

| clubName | roomNo |
|---|---|
| 수영 | 101호 |
| 바둑 | 102호 |
| 축구 | 103호 |
| 봉사 | 104호 |

UNION ALL

| stdName | addr |
|---|---|
| 강호동 | 경북 |
| 김제동 | 경남 |
| 김용만 | 서울 |
| 이휘재 | 경기 |
| 박수홍 | 서울 |
| 수영 | 101호 |
| 바둑 | 102호 |
| 축구 | 103호 |
| 봉사 | 104호 |

그림 8-22 UNION의 결합 과정

# 1-6 UNION/UNION ALL

- UNION 연산자를 사용하는 형식

```
SELECT 문장1
   UNION [ALL]
SELECT 문장2
```

- 중복된 열까지 모두 출력하려면 UNION ALL 사용

```
USE cookDB;
SELECT stdName, addr FROM stdTBL
    UNION ALL
SELECT clubName, roomNo FROM clubTBL;
```

| stdName | addr |
|---|---|
| 강호동 | 경북 |
| 김용만 | 서울 |
| 김제동 | 경남 |
| 박수홍 | 서울 |
| 이휘재 | 경기 |
| 바둑 | 102호 |
| 봉사 | 104호 |
| 수영 | 101호 |
| 축구 | 103호 |

# 1-7 NOT IN/IN

- NOT IN
  - 첫 번째 쿼리의 결과 중에서 두 번째 쿼리에 해당하는 것을 제외하고 출력

- cookDB의 사용자를 모두 출력하되 전화번호가 없는 사람을 제외

```
SELECT userName, CONCAT(mobile1, '-', mobile2) AS '전화번호' FROM userTBL
  WHERE userName NOT IN (SELECT userName FROM userTBL WHERE mobile1 IS NULL);
```

| userName | 전화번호 |
|---|---|
| 강호동 | 011-22222222 |
| 김국진 | 019-33333333 |
| 김용만 | 010-44444444 |
| 이휘재 | 011-88888888 |
| 이경규 | 018-99999999 |
| 남희석 | 016-66666666 |
| 박수홍 | 010-00000000 |
| 유재석 | 010-11111111 |

- 전화번호가 없는 사람만 조회

```
SELECT userName, CONCAT(mobile1, mobile2) AS '전화번호' FROM userTBL
  WHERE userName IN (SELECT userName FROM userTBL WHERE mobile1 IS NULL);
```

| userName | 전화번호 |
|---|---|
| 김제동 | NULL |
| 신동엽 | NULL |

# 2-1 SQL 프로그래밍의 개요

- SQL 프로그래밍
  - SQL에도 다른 프로그래밍 언어와 비슷한 분기, 흐름 제어, 반복 등의 기능이 있음

- 스토어드 프로시저의 작성 형식

```
DELIMITER $$
CREATE PROCEDURE 스토어드프로시저이름()
BEGIN

이곳에 SQL 프로그래밍 코딩

END $$
DELIMITER ;
CALL 스토어드프로시저이름();
```

# 2-2 IF … ELSE … END IF 문

- IF … ELSE … END IF 문
  - 조건에 따라 분기하는 명령
  - 명령을 실행한 결과 한 문장 이상 처리해야 할 때는 BEGIN … END 문으로 묶어야 하는데, 실행할 문장이 한 문장뿐이더라도 습관적으로 BEGIN … END 문으로 묶는 것이 좋음

```
IF <불 표현식> THEN
        SQL문장들1 …
ELSE
        SQL문장들2 …
END IF;
```

# 2-2 IF ... ELSE ... END IF 문

- <불 표현식>이 거짓이면서 이후에 어떤 처리도 하지 않는다면 ELSE 이하 구문을 생략할 수 있음

```
DROP PROCEDURE IF EXISTS ifProc; -- 기존에 스토어드 프로시저를 만든 적이 있다면 삭제
DELIMITER $$
CREATE PROCEDURE ifProc()
BEGIN
  DECLARE var1 INT; -- var1 변수 선언
  SET var1 = 100; -- 변수에 값 대입

  IF var1 = 100 THEN -- 만약 @var1이 100이라면
   SELECT '100입니다.';
  ELSE
   SELECT '100이 아닙니다.';
  END IF;
END $$
DELIMITER ;
CALL ifProc();
```

# 2-2 IF ... ELSE ... END IF 문

- 직원 번호가 10001번인 직원의 입사일이 5년이 넘었는지 확인

```
DROP PROCEDURE IF EXISTS ifProc2;
USE employees;

DELIMITER $$
CREATE PROCEDURE ifProc2()
BEGIN
    DECLARE hireDATE DATE; -- 입사일
    DECLARE curDATE DATE; -- 오늘
    DECLARE days INT; -- 근무한 일수

    SELECT hire_date INTO hireDate -- hire_date 열의 결과를 hireDATE에 대입
      FROM employees.employees
      WHERE emp_no = 10001;

    SET curDATE = CURRENT_DATE(); -- 현재 날짜
    SET days = DATEDIFF(curDATE, hireDATE); -- 날짜의 차이, 일 단위

    IF (days/365) >= 5 THEN -- 5년이 지났다면
        SELECT CONCAT('입사한지 ', days, '일이나 지났습니다. 축하합니다!') AS '메시지';
    ELSE
        SELECT '입사한지 ' + days + '일밖에 안되었네요. 열심히 일하세요.' AS '메시지';
    END IF;
END $$
DELIMITER ;
CALL ifProc2();
```

**실행 결과**

'입사한지 000000일이나 지났습니다. 축하합니다!'

# 2-3 CASE 문

- 다중 분기를 IF문으로 작성

```
DROP PROCEDURE IF EXISTS ifProc3;
DELIMITER $$
CREATE PROCEDURE ifProc3()
BEGIN
    DECLARE point INT;
    DECLARE credit CHAR(1);
    SET point = 77;

    IF point >= 90 THEN
        SET credit = 'A';
    ELSEIF point >= 80 THEN
        SET credit = 'B';
    ELSEIF point >= 70 THEN
        SET credit = 'C';
    ELSEIF point >= 60 THEN
        SET credit = 'D';
    ELSE
        SET credit = 'F';
    END IF;
    SELECT CONCAT('취득점수==>', point), CONCAT('학점==>', credit);
END $$
DELIMITER ;
CALL ifProc3();
```

**실행 결과**

```
취득점수==>77 학점==>C
```

# 2-3 CASE 문

- IF 문을 CASE 문으로 변경

```
DROP PROCEDURE IF EXISTS caseProc;
DELIMITER $$
CREATE PROCEDURE caseProc()
BEGIN
    DECLARE point INT;
    DECLARE credit CHAR(1);
    SET point = 77;

CASE
        WHEN point >= 90 THEN
                SET credit = 'A';
        WHEN point >= 80 THEN
                SET credit = 'B';
        WHEN point >= 70 THEN
                SET credit = 'C';
        WHEN point >= 60 THEN
                SET credit = 'D';
        ELSE SET credit = 'F';
    END CASE;
    SELECT CONCAT('취득점수==>', point), CONCAT('학점==>', credit);
END $$
DELIMITER ;
CALL caseProc();
```

# [실습 8-4] CASE 문을 활용하여 SQL 프로그래밍하기

교재 280~283p 참고

0 구매액에 따라 고객 등급 분류하기

0-1 최종 결과 확인

| userID | userName | 총구매액 | 고객등급 |
|---|---|---|---|
| PSH | 박수홍 | 1920 | 최우수고객 |
| KHD | 강호동 | 1210 | 우수고객 |
| KYM | 김용만 | 200 | 일반고객 |
| LHJ | 이휘재 | 95 | 일반고객 |
| KJD | 김제동 | 75 | 일반고객 |
| SDY | 신동엽 | NULL | 유령고객 |
| YJS | 유재석 | NULL | 유령고객 |
| KKJ | 김국진 | NULL | 유령고객 |
| LKK | 이경규 | NULL | 유령고객 |
| NHS | 남희석 | NULL | 유령고객 |

1 cookDB 초기화하기

1-1 C:₩SQL₩cookDB.sql 파일을 열고 실행

1-2 열린 쿼리 창을 모두 닫고 새 쿼리 창을 염

# [실습 8-4] CASE 문을 활용하여 SQL 프로그래밍하기

교재 280~283p 참고

## 2 고객 등급 분류하기

### 2-1 구매액이 높은 순으로 정렬

```
USE cookDB;
SELECT userID, SUM(price * amount) AS '총구매액'
   FROM buyTBL
   GROUP BY userID
   ORDER BY SUM(price * amount) DESC;
```

| userID | 총구매액 |
|---|---|
| PSH | 1920 |
| KHD | 1210 |
| KYM | 200 |
| LHJ | 95 |
| KJD | 75 |

### 2-2 회원 이름을 넣어 출력

```
SELECT B.userID, U.userName, SUM(price * amount) AS '총구매액'
   FROM buyTBL B
      INNER JOIN userTBL U
         ON B.userID = U.userID
   GROUP BY B.userID, U.userName
   ORDER BY SUM(price * amount) DESC;
```

| userID | userName | 총구매액 |
|---|---|---|
| PSH | 박수홍 | 1920 |
| KHD | 강호동 | 1210 |
| KYM | 김용만 | 200 |
| LHJ | 이휘재 | 95 |
| KJD | 김제동 | 75 |

# [실습 8-4] CASE 문을 활용하여 SQL 프로그래밍하기

교재 280~283p 참고

2-3 조건을 만족하지 않는 행도 모두 출력

```
SELECT B.userID, U.userName, SUM(price * amount) AS '총구매액'
   FROM buyTBL B
      RIGHT OUTER JOIN userTBL U
         ON B.userID = U.userID
   GROUP BY B.userID, U.userName
   ORDER BY SUM(price * amount) DESC;
```

| | userID | userName | 총구매액 |
|---|---|---|---|
| ▶ | PSH | 박수홍 | 1920 |
| | KHD | 강호동 | 1210 |
| | KYM | 김용만 | 200 |
| | LHJ | 이휘재 | 95 |
| | KJD | 김제동 | 75 |
| | NULL | 유재석 | NULL |
| | NULL | 김국진 | NULL |
| | NULL | 이경규 | NULL |
| | NULL | 남희석 | NULL |
| | NULL | 신동엽 | NULL |

2-4 userID의 기준을 구매 테이블 (buyTBL)에서 회원 테이블(userTBL)로 변경

```
SELECT U.userID, U.userName, SUM(price * amount) AS '총구매액'
   FROM buyTBL B
      RIGHT OUTER JOIN userTBL U
         ON B.userID = U.userID
   GROUP BY U.userID, U.userName
   ORDER BY SUM(price * amount) DESC;
```

| | userID | userName | 총구매액 |
|---|---|---|---|
| ▶ | PSH | 박수홍 | 1920 |
| | KHD | 강호동 | 1210 |
| | KYM | 김용만 | 200 |
| | LHJ | 이휘재 | 95 |
| | KJD | 김제동 | 75 |
| | YJS | 유재석 | NULL |
| | KKJ | 김국진 | NULL |
| | LKK | 이경규 | NULL |
| | NHS | 남희석 | NULL |
| | SDY | 신동엽 | NULL |

# [실습 8-4] CASE 문을 활용하여 SQL 프로그래밍하기

교재 280~283p 참고

2-5 총구매액에 따라 고객 분류

```
CASE
     WHEN (총구매액 >= 1500) THEN '최우수고객'
     WHEN (총구매액 >= 1000) THEN '우수고객'
     WHEN (총구매액 >= 1 ) THEN '일반고객'
     ELSE '유령고객'
END
```

2-6 CASE 구문을 SELECT 절에 추가하여 최종 쿼리문 작성

```
SELECT U.userID, U.userName, SUM(price * amount) AS '총구매액',
       CASE
          WHEN (SUM(price * amount) >= 1500) THEN '최우수고객'
          WHEN (SUM(price * amount) >= 1000) THEN '우수고객'
          WHEN (SUM(price * amount) >= 1 ) THEN '일반고객'
          ELSE '유령고객'
       END AS '고객등급'
    FROM buyTBL B
       RIGHT OUTER JOIN userTBL U
         ON B.userID = U.userID
    GROUP BY U.userID, U.userName
    ORDER BY sum(price * amount) DESC;
```

# 2-4 WHILE 문, ITERATE/LEAVE 문

- WHILE 문
  - 다른 프로그래밍 언어의 WHILE 문과 동일한 개념
  - <불식>이 참인 동안 WHILE 문 내의 명령을 계속 반복

```
WHILE <불식> DO
        SQL 명령문들 ...
END WHILE;
```

# 2-4 WHILE 문, ITERATE/LEAVE 문

- 1부터 100까지의 값을 모두 더하는 코드

```
DROP PROCEDURE IF EXISTS whileProc;
DELIMITER $$
CREATE PROCEDURE whileProc()
BEGIN
    DECLARE i INT; -- 1부터 100까지 증가할 변수
    DECLARE hap INT; -- 더한 값을 누적할 변수
    SET i = 1;
    SET hap = 0;

    WHILE (i <= 100) DO
        SET hap = hap + i; -- hap의 원래 값에 i를 더하여 hap에 넣으라는 의미
        SET i = i + 1; -- i의 원래 값에 1을 더하여 i에 넣으라는 의미
    END WHILE;

    SELECT hap;
END $$
DELIMITER ;
CALL whileProc();
```

**실행 결과**

```
5050
```

# 2-4 WHILE 문, ITERATE/LEAVE 문

- 1~100 중 7의 배수를 합계에서 제외
  - ITERATE 문과 LEAVE 문 사용

```
DROP PROCEDURE IF EXISTS whileProc2;
DELIMITER $$
CREATE PROCEDURE whileProc2()
BEGIN
    DECLARE i INT; -- 1부터 100까지 증가할 변수
    DECLARE hap INT; -- 더한 값을 누적할 변수
    SET i = 1;
    SET hap = 0;

    myWhile: WHILE (i <= 100) DO -- While 문에 label을 지정
      IF (i%7 = 0) THEN
          SET i = i + 1;
          ITERATE myWhile; -- 지정한 label 문으로 가서 계속 진행
      END IF;

      SET hap = hap + i;
      IF (hap > 1000) THEN
         LEAVE myWhile; -- 지정한 label 문을 떠남(While 종료)
      END IF;
      SET i = i + 1;
    END WHILE;

    SELECT hap;
END $$
DELIMITER ;
CALL whileProc2();
```

**실행 결과**

```
1029
```

# 2-5 오류 처리

- 오류 처리 형식

```
DECLARE 액션 HANDLER FOR 오류조건 처리할_문장;
```

  - 액션: 오류가 발생했을 때의 행동 정의. CONTINUE 또는 EXIT 사용. CONTINUE가 나오면 맨 뒤의 '처리할 문장' 부분이 처리되고 EXIT가 나오면 종료된다.
  - 오류조건: 어떤 오류를 처리할 것인지 지정
  - 처리할 문장: 처리할 문장이 하나라면 한 문장이 오고, 처리할 문장이 여러 개라면 BEGIN ... END 문으로 묶음

- 조회하는 테이블이 없을 때 오류를 처리하는 코드

```
DROP PROCEDURE IF EXISTS errorProc;
DELIMITER $$
CREATE PROCEDURE errorProc()
BEGIN
    DECLARE CONTINUE HANDLER FOR 1146 SELECT '테이블이 없어요ㅠㅠ' AS '메시지';
    SELECT * FROM noTable; -- noTable은 없음
END $$
DELIMITER ;
CALL errorProc();
```

**실행 결과**

```
테이블이 없어요ㅠㅠ
```

# 2-5 오류 처리

- cookDB의 회원 테이블(userTBL)에 이미 존재하는 아이디인 YJS를 생성(오류 발생)

```
DROP PROCEDURE IF EXISTS errorProc2;
DELIMITER $$
CREATE PROCEDURE errorProc2()
BEGIN
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    BEGIN
      SHOW ERRORS; -- 오류 메시지를 보여줌
      SELECT '오류가 발생했네요. 작업은 취소시켰습니다.' AS '메시지';
      ROLLBACK; -- 오류 발생 시 작업을 롤백
    END;
    INSERT INTO userTBL VALUES('YJS', '윤정수', 1988, '서울', NULL, NULL, 170, CURRENT_DATE());
    -- 중복되는 아이디이므로 오류 발생
END $$
DELIMITER ;
CALL errorProc2();
```

| | Level | Code | Message |
|---|---|---|---|
| | Error | 1062 | Duplicate entry 'YJS' for key 'PRIMARY' |

| | 메시지 |
|---|---|
| | 오류가 발생했네요. 작업은 취소시켰습니다. |

# 2-6 동적 SQL

- 동적 SQL
  - PREPARE 문은 SQL 문을 실행하지는 않고 따로 준비만 해놓으며, EXECUTE 문은 PREPARE문으로 준비한 쿼리문을 실행
  - 미리 쿼리문을 준비한 후 나중에 실행하는 것을 '동적 SQL'이라고 함
  - 동적 SQL로 쿼리문을 실행한 후에는 DEALLOCATE PREPARE 문으로 준비 했던 문장을 해제하는 것이 바람직

```
use cookDB;
PREPARE myQuery FROM 'SELECT * FROM userTBL WHERE userID = "NHS"';
EXECUTE myQuery;
DEALLOCATE PREPARE myQuery;
```

- 쿼리문을 실행하는 순간의 날짜와 시간이 입력되는 코드

```
USE cookDB;
DROP TABLE IF EXISTS myTable;
CREATE TABLE myTable (id INT AUTO_INCREMENT PRIMARY KEY, mDate DATETIME);

SET @curDATE = CURRENT_TIMESTAMP(); -- 현재 날짜와 시간

PREPARE myQuery FROM 'INSERT INTO myTable VALUES(NULL, ?)';
EXECUTE myQuery USING @curDATE;
DEALLOCATE PREPARE myQuery;

SELECT * FROM myTable;
```