-- 3 장
-- 데이터베이스 모델링 과정이란 실제 데이터를 서버 상에 넣기 위한 과정
-- DBMS 데이터베이스 관리 시스템. 그 중 MySQL 을 설치한 것, SQL 은 언어의 이름입니다.
-- DB 데이터 베이스. 안에 아무것도 없다고 가정하는 것이 좋음
-- DB 를 표형태로 저장하는 것이 일반적인 방법입니다 [table]
-- table 은 열과 행으로 구성되어 있으며 세로가 열[데이터의 특징, 데이터의 형식] , 행의 건수는 대부분 데이터의 건수
-- 열에는 데이터의 형식을 정해야하고, PK 열을 지정가능하다. PK 열은 중복이 불가능하고 NANNULL 이다
-- 데이터 베이스에 데이터를 담음 ! 데이터베이스를 여러개 만들고 이를 관리하는 시스템이 DBMS

-- 데이터베이스생성 -> 테이블 생성 -> 데이터 입력 -> 활용

```
CREATE SCHEMA `shopdb` ;
-- 데이터베이스 생성 완료

CREATE TABLE `shopdb`.`memberdb` (
   `memberid` CHAR(8) NOT NULL,
   `membername` CHAR(5) NOT NULL,
   `memberAddress` CHAR(23) NULL,
   PRIMARY KEY (`memberid`));

   CREATE TABLE `shopdb`.`producttbl` (
   `productname` CHAR(5) NOT NULL,
   `cost` INT NOT NULL,
   `makedate` DATE NULL,
   `company` CHAR(5) NULL,
   `amount` INT NOT NULL,
   PRIMARY KEY (`productname`));
```

```
  -- 데이터베이스 테이블 만들기

 SELECT * FROM shopdb.memberdb;
 -- SQL 문으로 데이터표 클릭하기

INSERT INTO `shopdb`.`memberdb` (`memberid`, `membername`, `memberAddress`)
VALUES ('1234', '박수빈', '부천시');
INSERT INTO `shopdb`.`memberdb` (`memberid`, `membername`, `memberAddress`)
VALUES ('2345', '박수민', '상동구');
INSERT INTO `shopdb`.`memberdb` (`memberid`, `membername`, `memberAddress`)
VALUES ('5678', '박수', '사상시');
INSERT INTO `shopdb`.`memberdb` (`memberid`, `membername`, `memberAddress`)
VALUES ('5768', '박', '고양시');
-- 정보 넣는 방법

SELECT * FROM shopdb.memberdb;
SELECT * FROM memberdb;
SELECT membername FROM memberdb;
-- 특정 열만 고르고 싶을 떄. * 넣으면 전체가 다 골라짐
SELECT * FROM memberdb WHERE memberAddress = '고양시';
-- 특정 조건 , 행 하에서 고르고 싶을 때 WHERE 이용

-- 데이터는 언제나 날아갈 수 있아 ! 백업하고 복원하는 것이 중요합니다
-- 응용프로그램과 데이터베이스 명령문을 연동하는 것도 매우매우 중요함 (책
후반부에서 자세히 다룸)
-- 데이터의 형태는 매우 다양하다.
-- index 책의 색인과 같은 역할 , SELECT 시 매우 유용하게 사용

CREATE TABLE indexTBL
(First_name varchar(14),last_name varchar(16), hire_date date) ;
```

```
INSERT INTO indexTBL SELECT first_name, last_name, hire_date FROM
employees.employees LIMIT 500 ;
-- 다른 데이터부터 데이터를 넣었습니다.

SELECT * FROM indextbl;
SELECT * FROM indextbl WHERE First_name = 'mary';
-- select 하는 방법을 적용, 근데 이걸 분석하면 실행플랜을 보면 full table scan 을
했다고 되어있음. 즉, 전체를 다 봄

-- CREATE INDEX 어쩌구를 통해서 인덱스를 만들수 이쓰어요
-- 요걸 통해서 인덱스를 만드면 실제 데이터의 결과와 쿼리문은 같게 나오는데
NONEUNIQUEKEY 를 통해 찾았다고 나와있음. 즉 빠르게 찾음 !!
-- 실무에서 유용한 기능

-- VIEW 테이블에 링크된 개념. 가상의 테이블
-- 원 데이터에 접근하지 못하게 하고 가상의 데이터만 볼 수 있게 하도록 위해서 , 즉
보안을 위해서 !!

CREATE VIEW v_memberdb
AS
        SELECT memberid, memberaddress FROM memberdb ;
-- VieW 만드는 법
SELECT * FROM v_memberdb ;
-- 이렇게 일부 정보에만 접근가능하게 뷰를 구성가능합니다.

DELIMITER //
CREATE PROCEDURE myProc()
BEGIN
        SELECT * FROM memberTBL WHERE membername = '박' ;
END//
DELIMITER ;
```

```
CALL myProc()
-- procedure 만드는 법

-- 트리거는 테이블을 관리할 시 제거된 기록을 만들때 테이블을 자퇴생 테이블로
옮기고 지우면 되는데
-- 그걸 실수할 수도 있으니까 그냥 지우면 바로 다른 공간으로 쏙 들어가게 해주세요 !!

UPDATE memberdb SET memberadress = '수빈' WHERE membername = '박';
-- 업데이트하는 방법

DELETE FROM memberdb WHERE memberName = '박'
-- 이렇게 하면 삭제 가능함. 그럼데 삭제한 정보도 일단 보관하고 싶어 자동으로 !! 그게
트리거

CREATE TABLE deletedMember (
        memberid char(8),
    memberaddress char(24),
    memberName char(5),
    deletedDate date
    );
    -- 삭제 관리를 위한 테이블을 만든 후

DELIMITER //
CREATE TRIGGER trg_deletedmember
        AFTER DELETE
-- 삭제될때마다자동
    ON memberdb FOR EACH ROW
-- 트리거를 부착할테이블, 각 행마다 적용
BEGIN
        INSERT INTO deletedMember
```

```
                VALUES (OLD.memberid,
OLD.membername,OLD.memberaddress,CURDATE()); -- 아이디를 삭제하면 예전의 것이
되니까 이렇게, 그리고
END //
DELIMITER ;

DELETE FROM memberdb WHERE memberName = '박수민';
-- 삭제했고 이제 확인해보면 트리거 덕분에 자동으로 deleted 에 있어용.

-- 백업하는 법 ! 현재의 데이터베이스를 다른매체에
-- 복원은 문제가 생긴 베이스를 되살리는 것

-- 연동하는 법을 알아봅시다. 비주얼스튜디오 연동
```