

2

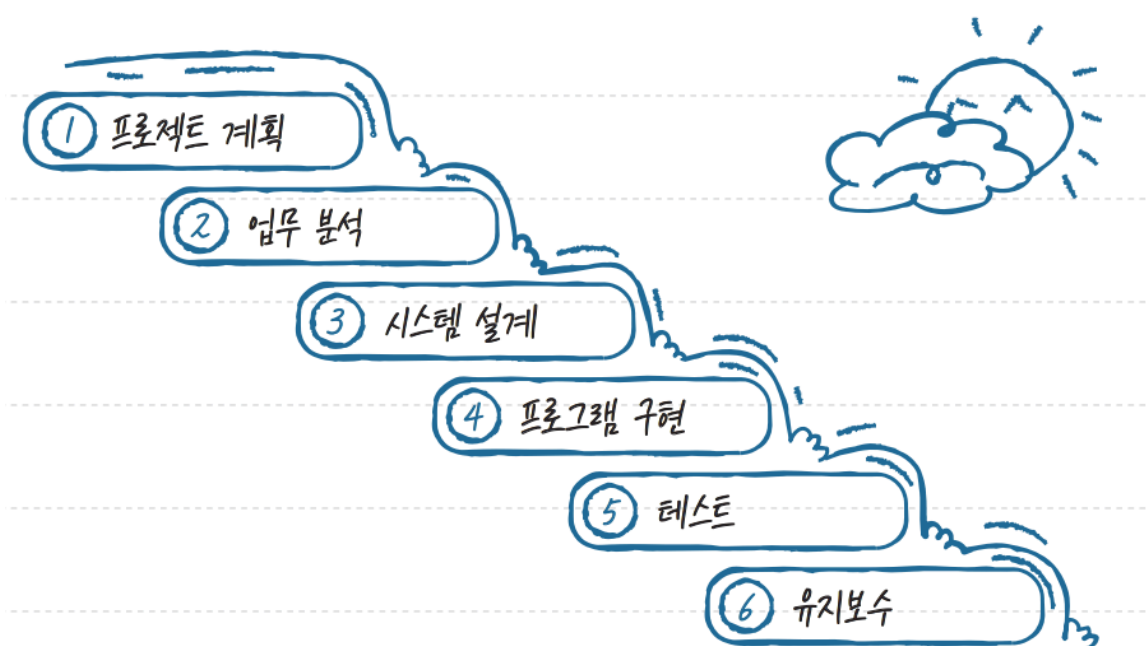
2장 : 실전용 SQL 맛보기

이번 장에서는 실전에서 사용되는 SQL을 간단하게 보면서 SQL과 데이터베이스에 대한 전반적인 흐름을 이해한다.

2-1 : 데이터베이스 모델링

데이터베이스 모델링 : 테이블의 구조를 미리 설계하는 개념. 프로젝트에서도 데이터베이스 모델링이 잘 되어야 제대로 된 데이터베이스를 구축할 수 있음. 프로젝트를 진행할 때 대표적으로 폭포수 모델을 사용한다. 데이터베이스 모델링은 폭포수 모델의 업무 분석과 시스템 설계 단계에 해당한다. 이 단계를 거치면 가장 중요한 데이터베이스 객체인 **테이블 구조**가 결정된다.

프로젝트 진행 단계



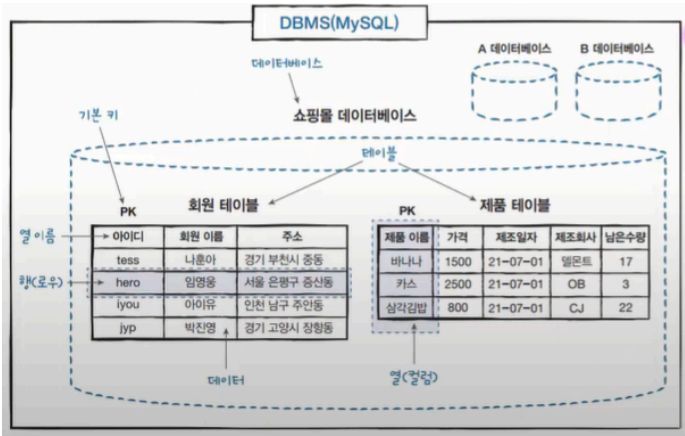
폭포수 모델의 개념 그림


폭포수 모델은 각 단계가 구분되어 프로젝트의 진행 단계가 명확하다는 장점이 있다. 하지만 문제가 발생할 경우 다시 앞 단계로 돌아가기 어렵다는 단점이 있다. 그래도 각 단계가 명확하기 때문에 지금도 많이 사용된다. 우리가 공부하는 데이터베이스 모델링은 **2.업무 분석** 과 **3. 시스템 설계** 단계에 해당한다. 8장에서 프로그램 구현과 데이터베이스 연결도 실제로 구현한다.

데이터베이스 모델링

데이터베이스 모델링이란 우리가 살고 있는 세상에서 사용되는 사물이나 작업을 DBMS의 데이터베이스 개체로 옮기기 위한 과정이라고 할 수 있다. 즉, 현실에서 쓰이는 것을 테이블로 변경하기 위한 작업이라 이해하면 된다. 데이터베이스 모델링은 건물 설계도를 그리는 사람에 따라 다양한 결과물이 나오는 것처럼 정답이 없다. 하지만 **좋은 모델링과 나쁜 모델링은 분명히 존재한다.**

데이터베이스 구성도



 용어 정리

1. 테이블 : 회원이나 제품의 데이터를 입력하기 위한 표 형태로 표현한 것

2. 데이터베이스 : 테이블이 저장되는 저장소. 그림에는 3개의 데이터베이스가 존재한다.(원통)

3. 데이터 형식 : 열에 저장될 데이터의 형식. 각 열에 맞는 형식이어야 한다. 데이터 형식은 테이블을 생성할 때 열 이름과 같이 지정해준다.

4. 기본 키(Prime Key) : 기본 키 열은 각 행을 구분하는 유일한 열. 줄여서 PK라고 표현한다.

기본 키는 중복되거나, 공백이면 안된다. 기본 키는 1개만 지정해야 하며 1개의 열에 지정한다.

2-2 : 데이터베이스 시작부터 끝

데이터베이스는 데이터를 저장하는 공간이다. MySQL을 설치한 후에는 가장 먼저 데이터베이스를 준비하고 내부에 테이블을 생성해야 한다. 테이블은 2차원 표 형태로 이루어져 있으며, 각 열에 해당하는 데이터를 한 행씩 입력할 수 있다. 필요하다면 행에 입력된 데이터를 수정하거나 삭제할 수 있으며, 마지막으로 입력이 완료된 데이터를 조회해서 활용할 수 있다.

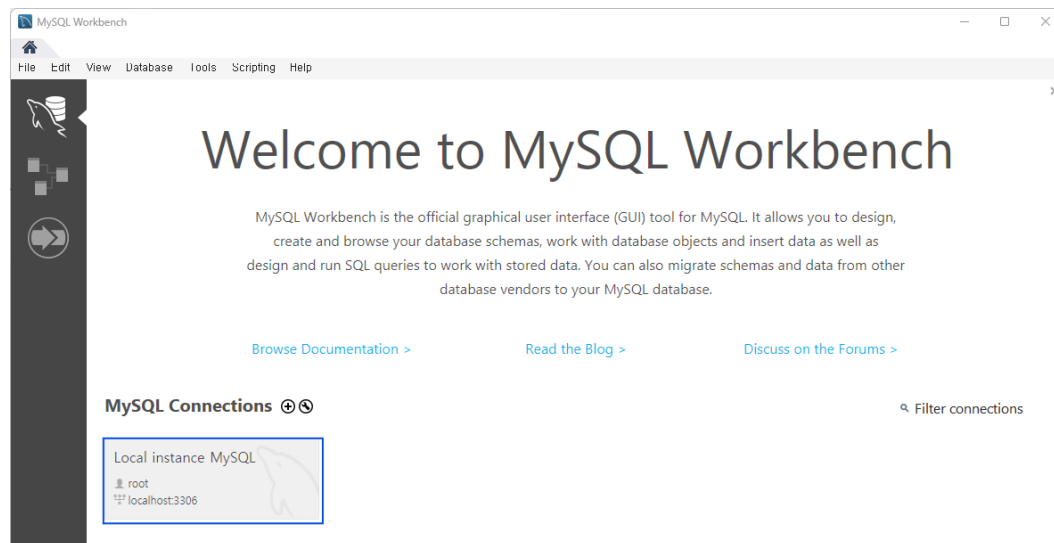
DBMS 설치하기

데이터베이스를 구축하기 위해선 DBMS를 설치해야 한다. 1장에서 우린 MySQL을 설치했다. 아직 내부에 우리가 사용할 데이터베이스는 없는 상태이다.

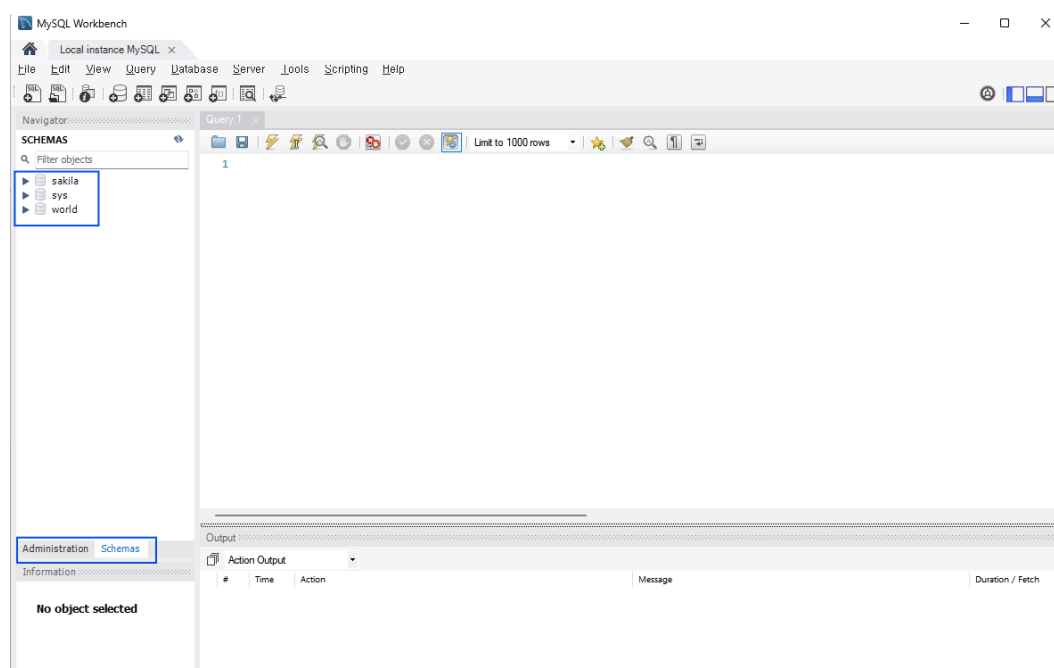
데이터베이스 만들기

이제 DBMS 안에 데이터베이스를 만들 차례이다.

1. SQL WorkBench 를 눌러 실행하고 **[MySQL Connections]**에서 **'Local instance MySQL'**을 클릭한다. Connect to MySQL Server 창이 나타나면 [User]는 root으로 지정되어 있고 비밀번호는 1장에 설정한대로 0000을 입력하고 진행하면 된다.

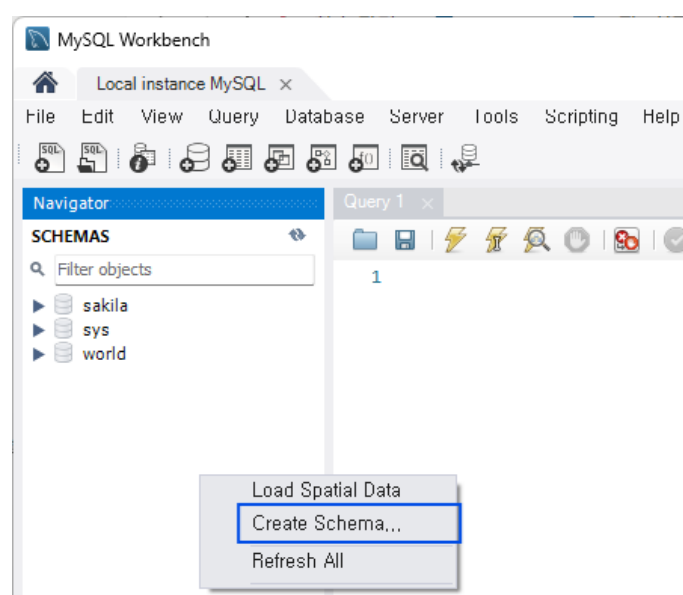


2. 좌측 하단에 **[Administration]**과 **[Schemas]**가 탭으로 구분되어 있다. **[Schemas]** 탭을 클릭하면 MySQL에 기본적으로 들어 있는 데이터베이스 3개가 보인다. 지금은 그냥 무시하고 넘어간다.

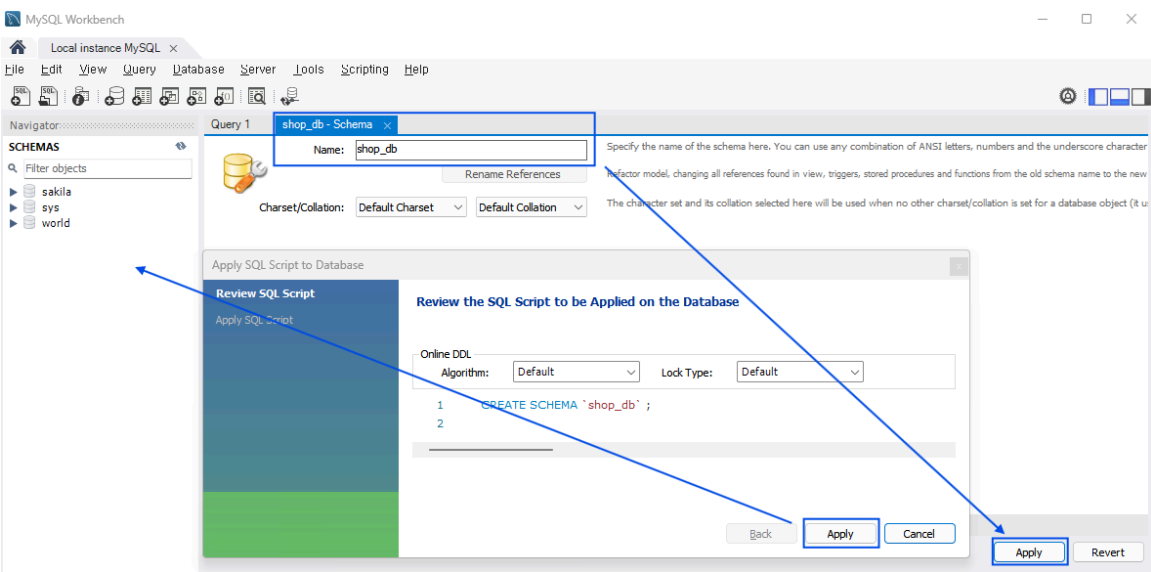


💬 **스키마(Schema)**는 데이터베이스와 동일한 용어이다. 앞으로 스키마 용어는 데이터베이스라고 이해해도 무방하다.

3. **[SCHEMAS]** 패널의 빈 부분에 마우스 오른쪽 클릭을 한 후 **[Create Schema]**를 선택한다.



4. 새로운 쿼리 창에서 **[Name]**에 shop_db를 입력하면 자동으로 탭 이름도 동일하게 변경된다. **[Apply]** 버튼을 클릭하면 Apply SQL Script to Database 창에 SQL 문이 자동으로 생성된다. 다시 **[Apply]**와 **[finish]**버튼을 클릭하면 좌측 **[SCHEMAS]** 패널의 목록에 shop_db가 추가된다.



현재는 처음이라 메뉴를 통해 데이터베이스를 생성했다. 앞으로 실무에서 사용하는 것처럼 CREATE DATABASE 'shop_db'와 같은 SQL문을 통해 DB를 생성한다.

테이블 만들기

테이블 설계하기

이제는 테이블을 만들 차례이다. 테이블을 생성하기 위해선 건물과 같이 설계가 필요하다. 테이블을 설계한다는 것은 테이블의 **열 이름**과 **데이터 형식**을 지정하는 것이다. 다음과 같이 회원 테이블의 설계를 완성했다고 가정하자.

| 열 이름(한글) | 영문 이름 | 데이터 형식 | 최대 길이 | Not NULL |
|-----------|-------------|----------|-------|----------|
| 아이디(기본 키) | member_id | 문자(CHAR) | 8글자 | Y |
| 회원 이름 | member_name | 문자(CHAR) | 5글자 | Y |
| 주소 | member_addr | 문자(CHAR) | 20글자 | N |

데이터 형식은 모두 문자이며 CHAR는 Character의 약자라는 MySQL 문법의 **약속된 언어**이다. NULL은 빈 것을 의미하며 Not NULL이 Y이면 반드시 입력해야 하고, N이면 입력을 해도 되고 없어도 무방하다.

다음으로 제품 테이블도 다음과 같이 설계를 완성했다고 가정하자.

| 열 이름(한글) | 영문 이름 | 데이터 형식 | 최대 길이 | Not Null |
|----------|--------------|----------|-------|----------|
| 제품 이름 | product_name | 문자(CHAR) | 4글자 | Y |
| 가격 | cost | 숫자(INT) | | Y |
| 제조일자 | make_date | 날짜(DATE) | | N |
| 제조회사 | company | 문자(CHAR) | 5글자 | N |
| 남은 수량 | amount | 숫자(INT) | | Y |

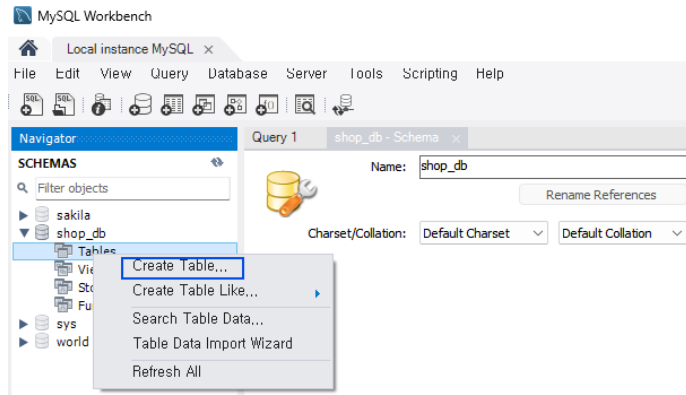
INT 는 Integer의 약자로 소수점이 없는 정수를 의미하고 DATE는 연,월,일을 입력한다.

열 이름을 영문으로 만들 때 띄어쓰기를 하지 않는 것이 좋다. 띄어쓰기를 할 경우에는 열 이름을 큰 따옴표로 묶어줘야 해서 불편하다. 그래서 보통 언더바(_)로 구분한다.

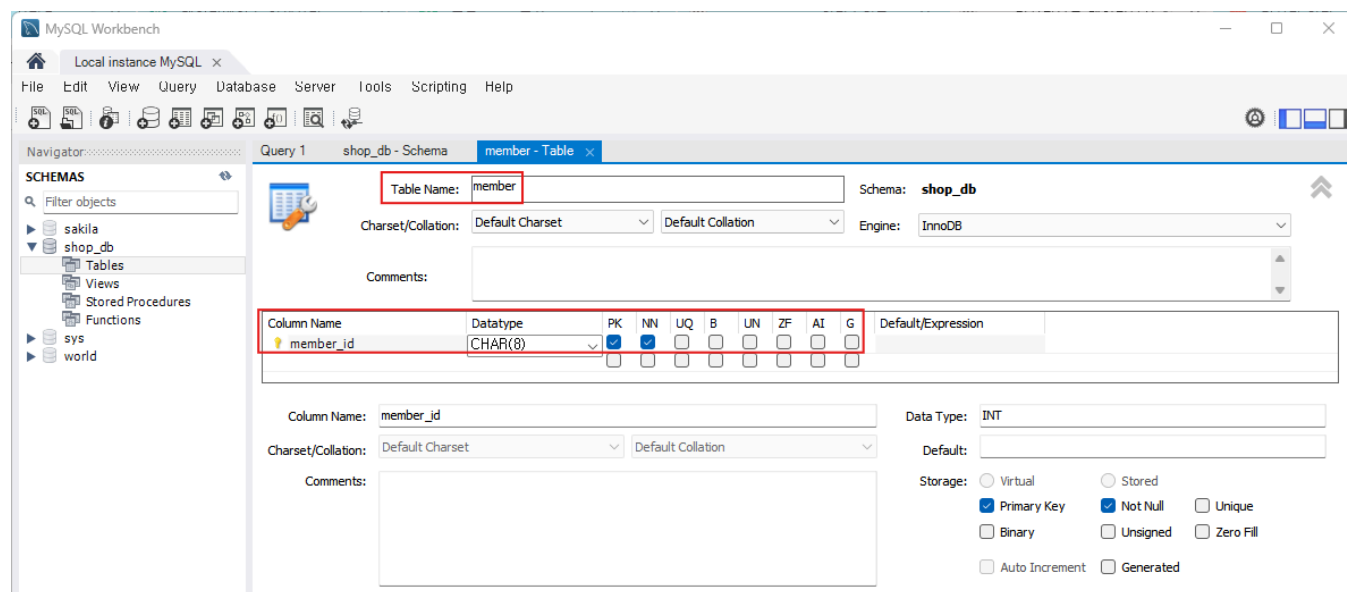
테이블 설계가 끝났으니 생성할 차례이다. 생성은 MySQL에서 진행하면 된다.

테이블 생성하기

1. MYSQL 작업 창의 스키마 패널에서 shop_db의 ►를 클릭해 확장하고 [Tables]를 마우스 오른쪽 버튼으로 클릭한 후 [Create Table]을 선택한다.

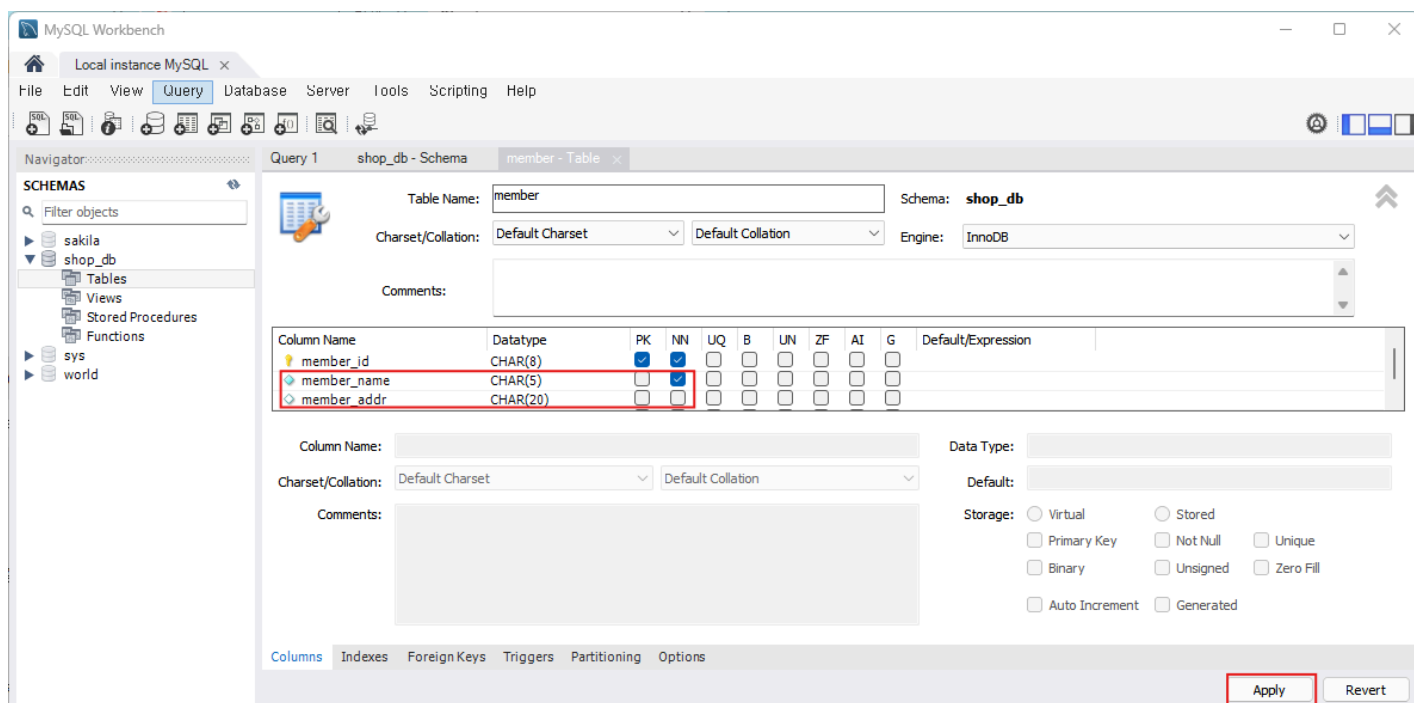


- 우선 앞에서 설계한 회원 테이블의 내용을 입력한다. [Table Name]에 member를 입력하고 [Column Name]의 첫 번째 항목을 더블 클릭한다. 첫 번째 열 이름은 member_id로 입력하고 DataType은 문자 8글자이므로 **CHAR(8)**로 입력한다. 그리고 설계할 때 아이디 열을 기본 키로 설정하기로 했으므로 PK와 NN(Not Null)을 체크한다.

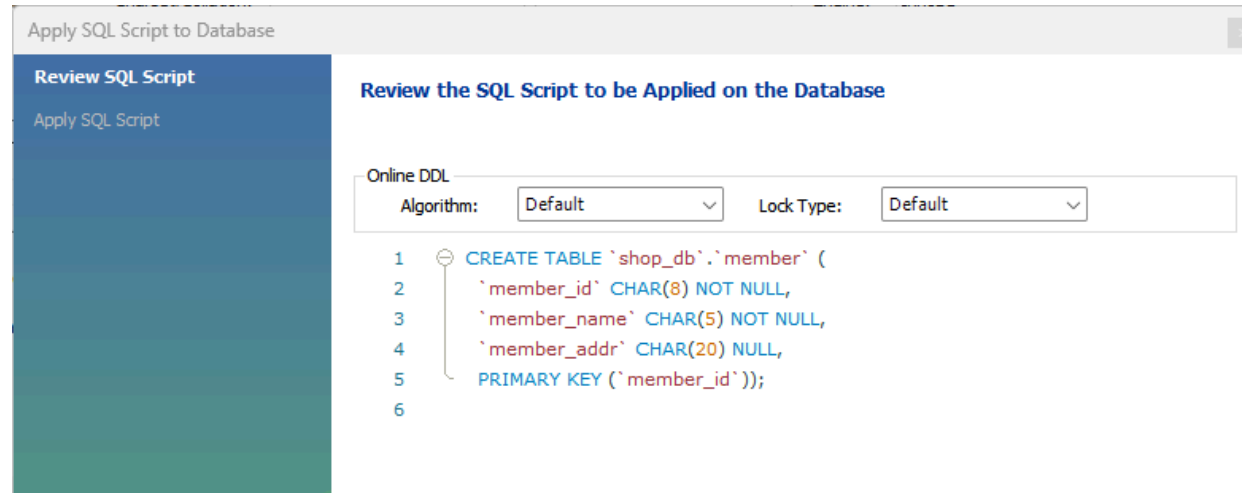


만약 열을 잘못 설정했다면 해당 열을 선택한 후에 마우스 오른쪽 버튼을 클릭하고 [Delete Selected]를 눌러 삭제한다.

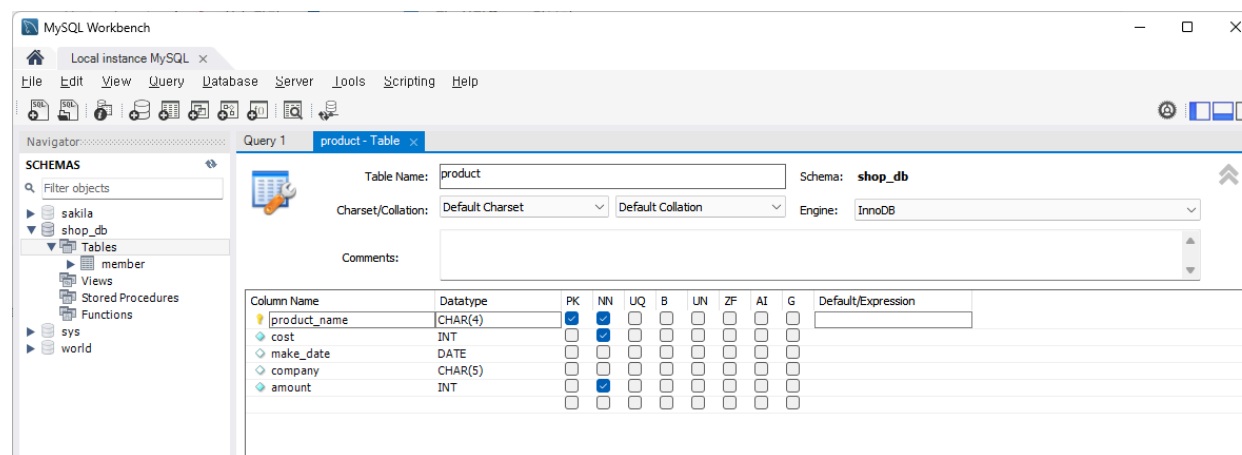
- 이어서 아래 빈줄을 더블 클릭해서 이전에 설계한대로 회원 이름과 주소를 추가한다. 추가가 완료되면 우측 하단에서 **[Apply]** 버튼을 클릭한다.



- Apply SQL Script to Database 창이 나타나고 자동으로 생성된 SQL 문이 보인다. [Apply]→[Finish] 버튼을 클릭해서 테이블 만들기를 완료한다. 다음으로 SQL 작업 창에서 [File]→[Close Tab] 메뉴를 선택해서 테이블 생성 창을 닫는다.



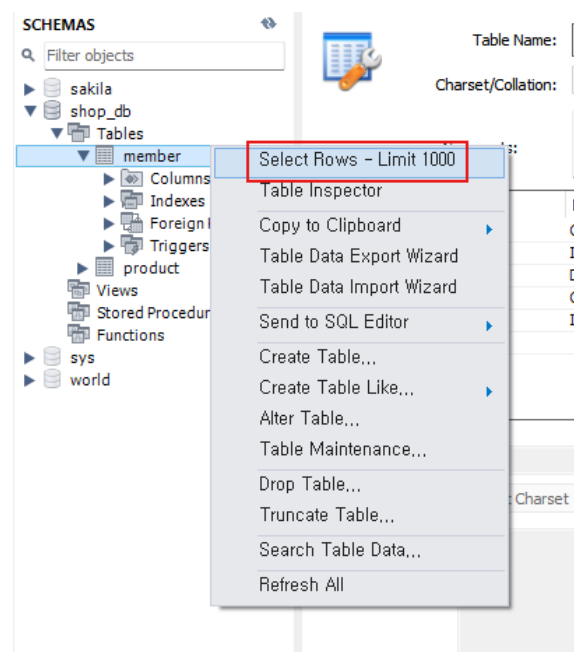
5. 1-4의 과정을 통해 이전에 설계한 제품 테이블도 만들면 된다. 테이블 이름은 product로 하면 된다.



데이터 입력하기

이제는 테이블에 실제로 데이터를 입력할 차례이다. 데이터는 행 단위로 입력한다. 여기서 회원 테이블에 4건, 제품 테이블엔 3건을 입력한다.

1. MySQL 작업 창의 스키마 패널에서 [Shop_db]→[Tables]→[member]를 선택하고 마우스 오른쪽 버튼을 클릭한 후 **[Select Rows] - Limits 1000]**을 선택한다.



2. MySQL 작업 창의 중앙에 **[Result Grid]** 창이 나타나는데 아직은 모두 NULL로 표시되어 있다. 여기에서 member_id, member_name, member_addr 항목의 NULL 부분을 클릭해서 입력한다. 입력이 완료되면 우측 하단의 Apply를 누르면 입력한 내용이 SQL로 생성된다. apply를 누르면 입력한 데이터 순서가 바뀌는데 member_id가 기본 키로 지정되어 있어서 오름차순으로 자동 정렬된 것이다. 입력이 끝나면 [File]→[Close Tab] 메뉴를 클릭해서 데이터 입력 창을 닫는다.

| Result Grid | | | Filter Rows: | Edit: |
|-------------|-------------|-------------|--------------|-------|
| member_id | member_name | member_addr | | |
| tess | 나홀아 | 경기 부천시 ... | | |
| hero | 임영웅 | 서울 은평구 ... | | |
| iyou | 아이유 | 인천 남구 주... | | |
| jyp | 박진영 | 경기 고양시 ... | | |
| NULL | NULL | NULL | | |

3. 위의 절차와 같은 방법으로 제품 테이블에 데이터 3건을 추가한다. 입력된 결과는 다음과 같다.

| Result Grid | | | | | | Filter Rows: | Edit: |
|--------------|------|------------|---------|--------|--|--------------|-------|
| product_name | cost | make_date | company | amount | | | |
| 바나나 | 1500 | 2021-07-01 | 멜론트 | 17 | | | |
| 카스 | 2500 | 2022-03-01 | OB | 3 | | | |
| 삼각김밥 | 800 | 2023-09-01 | CJ | 22 | | | |
| NULL | NULL | NULL | NULL | NULL | | | |

4. 이번에는 수정, 삭제를 진행하기 위해 먼저 데이터를 추가한다. 다시 회원 테이블에서 **[Select Rows] - Limits 1000]**를 선택한다. 연습용 데이터를 1건 입력한 후 [Apply]를 두 번 누르고 [Finish] 버튼을 클릭해서 입력 데이터를 저장한다.

| Result Grid | | | Filter Rows: | Edit: |
|-------------|-------------|-------------|--------------|-------|
| member_id | member_name | member_addr | | |
| carry | 머라이어 | 미국 텍사스 사막 | | |
| hero | 임영웅 | 서울 은평구 중산동 | | |
| iyou | 아이유 | 인천 남구 주안동 | | |
| jyp | 박진영 | 경기 고양시 주안동 | | |
| tess | 나홀아 | 경기 부천시 중동 | | |
| NULL | NULL | NULL | | |

5. 조금 전 입력한 데이터 내용을 수정한다. 여기서는 주소를 변경한다. 입력 후 다시 Apply 를 누르면 역시 SQL이 자동으로 생성된다. 이번에는 수정이기 때문에 UPDATE 문이 생성된 것을 확인할 수 있다.

```
UPDATE `shop_db`.`member` SET `member_addr` = '영국 런던 빅자글목' WHERE ('
```

6. 데이터를 삭제해본다. 삭제하고자 하는 행의 제일 앞 부분을 클릭하면 행이 파란색으로 선택된다. 그 상태에서 우클릭→Delete Row를 선택한다. 삭제 후에도 Apply를 클릭해야 한다. 삭제는 DELETE 문이 생성된다. Apply와 Finish 버튼을 눌러 수정한 내용을 적용한다.

```
DELETE FROM `shop_db`.`member` WHERE (`member_id` = 'carry');
```

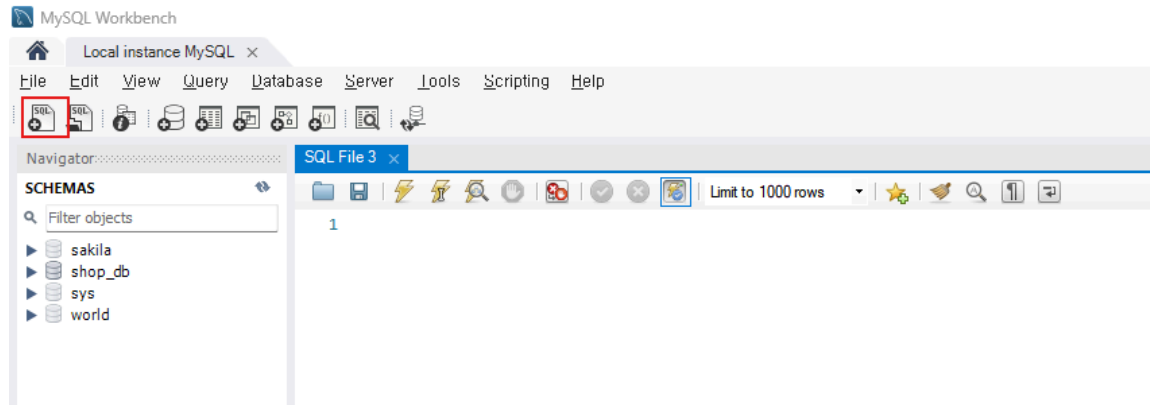


SQL에서 데이터 입력은 **INSERT**, 수정은 **UPDATE**, 삭제는 **DELETE** 문을 사용한다.

데이터 활용하기

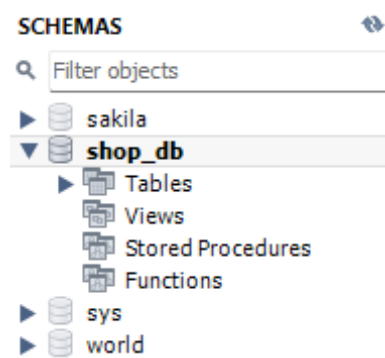
데이터까지 입력하여 데이터베이스 구축을 완료했다. 이번에는 데이터베이스를 활용하는 방법을 살펴본다. SQL에는 데이터베이스를 활용하기 위해 주로 SELECT 문을 사용한다.

1. 쿼리 창이 열려 있지 않은 상태에서 새 SQL을 입력하기 위해 툴바의 Create a new SQL tab for executing queries 아이콘을 누른다.



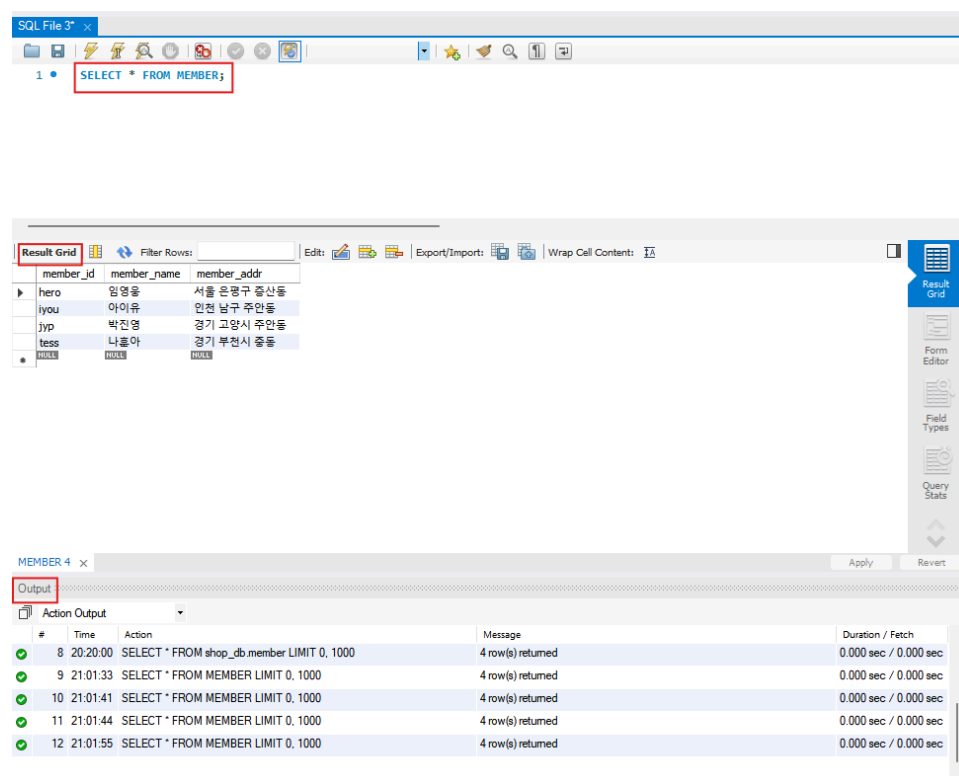
SQL은 쿼리라고도 하며 SQL을 입력할 수 있는 빈 창을 쿼리 창이라고 한다.

2. 작업할 데이터베이스를 선택하기 위해 스키마 패널에서 shop_db를 더블 클릭한다. 진하게 변경되면 앞으로 쿼리 창에 입력할 SQL이 선택된 shop_db에 적용된다는 의미이다.



3. 회원 테이블의 모든 행을 조회하기 위해 다음 SQL을 입력한다. SELECT의 기본 형식은 **SELECT 열_이름 FROM 테이블_이름 [WHERE 조건]**이고 *는 모든 열을 의미한다. : **SELECT * FROM MEMBER ;**

툴바에서 Execute the selected portion of the script or everything 아이콘(번개모양)을 클릭하면 **[Result Grid]**창에는 결과가, [Output] 패널에는 현재 결과의 건수와 조회하는데 소요된 시간(초)이 표시된다. 참고로 R처럼 [Ctrl]+[Enter]를 눌러도 결과가 나온다.

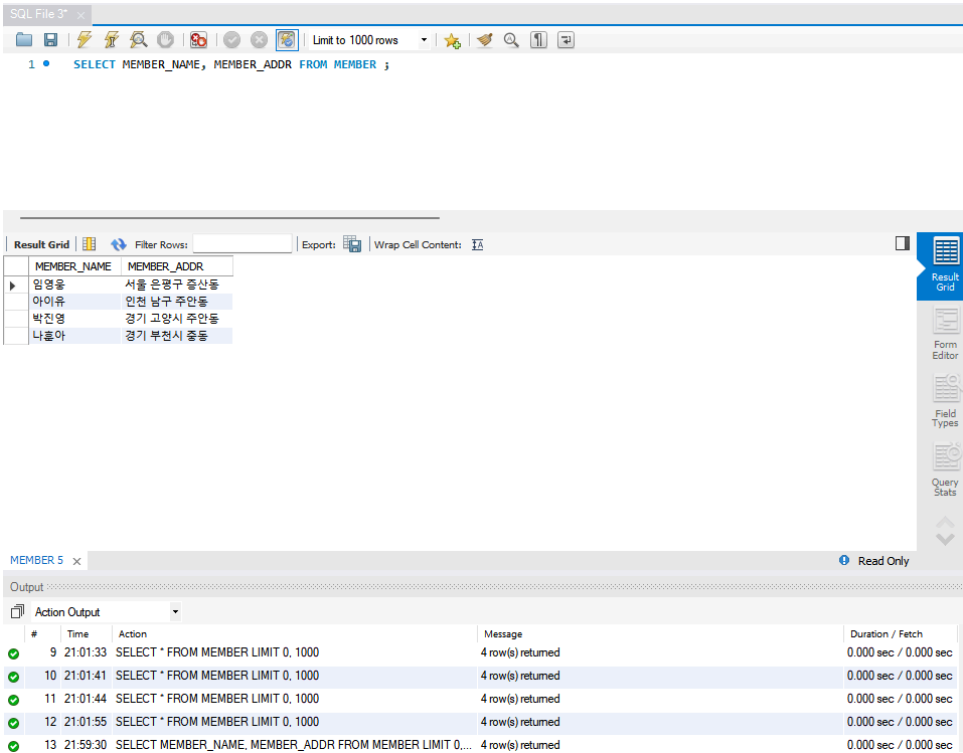


SQL의 대소문자

SQL은 대소문자를 구분하지 않는다. 하지만 이 책에서는 구분하기 쉽도록 SQL의 **예약어**는 대문자로 표시한다. SQL의 제일 뒤에는 **세미콜론(;)** 이 꼭 있어야 한다. 가끔 없어도 되는 경우도 있지만 그걸 모두 기억하는 것보다 있어야 한다고 생각하는게 더 편하다.

4. 회원 테이블 중에 이름과 주소만 출력해본다. 기존 SQL을 지우고 다음과 같이 새로 입력한 후 실행한다. 여러개의 열 이름을 콤마로 분리하면 필요한 열만 추출된다.

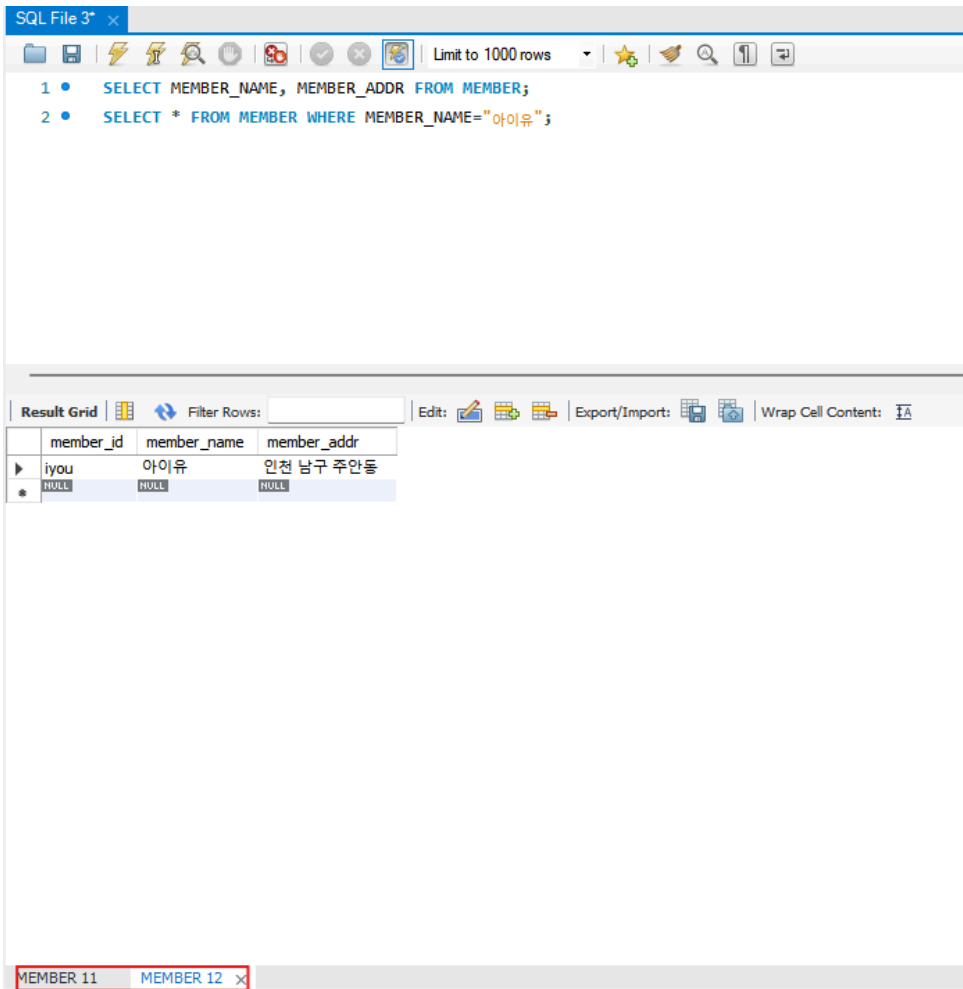
SELECT MEMBER_NAME, MEMBER_ADDR FROM MEMBER;



5. 특정 회원에 대한 정보만 추출해본다. 앞의 내용을 지우지 말고 다음 줄에 이어서 명령을 추가한다.

SELECT * FROM MEMBER WHERE MEMBER_NAME='아이유';

만약 툴바의 번개모양 아이콘을 누르면 모든 명령이 실행되고, [Ctrl]+[Enter]를 누르면 해당 줄만 실행된다. 현재는 2개의 명령밖에 없어서 괜찮지만, 현업에선 SQL 문이 많아지므로 주의해야 한다.



SQL 예약어와 자동 완성

쿼리 창에서 SQL을 입력하면 예약어는 자동으로 **파란색**으로 표시된다. 위에서 사용한 SELECT나 FROM은 이미 SQL에서 약속된 예약어이므로 파란색으로 표시되는 것이다.

2-3 : 데이터베이스 개체

테이블을 데이터베이스의 핵심 개체이다. 하지만 데이터베이스에서는 테이블 외에 **인덱스, 뷰, 스토어드 프로시저**, 트리거, 함수, 커서 등의 개체도 필요하다. 이번 절에선 5-7장에서 다루는 데이터베이스 개체, 즉 테이블을 제외한 인덱스, 뷰, 스토어드 프로시저 등에 대해 간단히 알아본다. 실무에서는 테이블 뿐만 아니라 데이터베이스 개체를 함께 활용해서 데이터베이스를 운영한다.

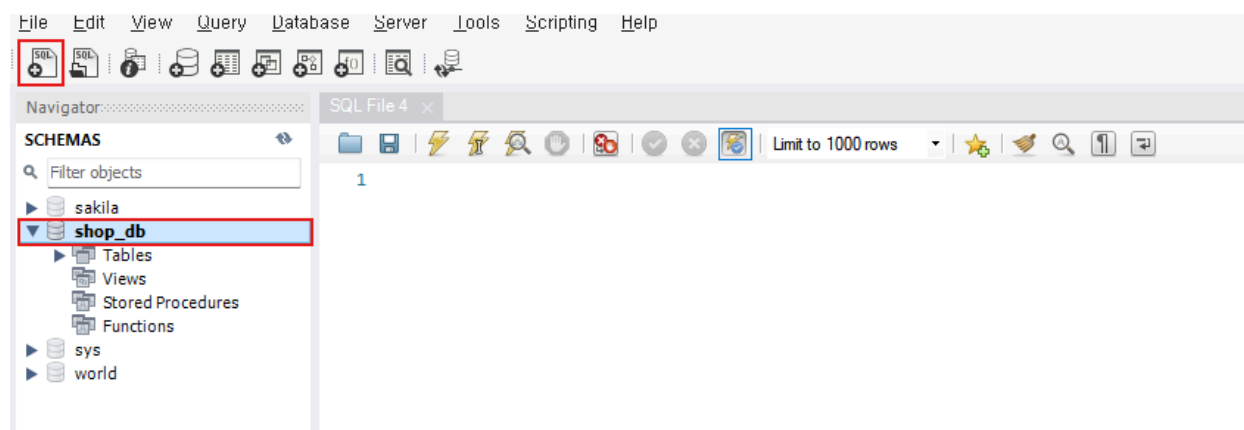
인덱스

데이터를 조회할 때 테이블에 데이터가 적다면 결과가 금방 나오지만 데이터가 많아질수록 결과가 나오는 시간이 많이 소요된다. 인덱스는 이런 경우 결과가 나오는 시간을 대폭 줄여준다. 책을 예시로 생각해보자. 책의 내용 중에 특정 단어를 찾고자 할 때, 책의 처음부터 마지막까지 한 페이지씩 전부 찾아보는 것은 상당히 시간이 오래 걸린다. 인덱스는 책의 제일 뒤에 수록되는 **‘찾아보기’**와 비슷한 개념으로 찾고 싶은 단어를 찾고 페이지로 이동하는 편리한 도구이다. 인덱스도 찾아보기와 비슷한 개념이다. 실무에선 대용량 데이터를 다룰 때 인덱스를 사용할 줄 아는 것이 큰 도움이 될 것이다.

인덱스 실습

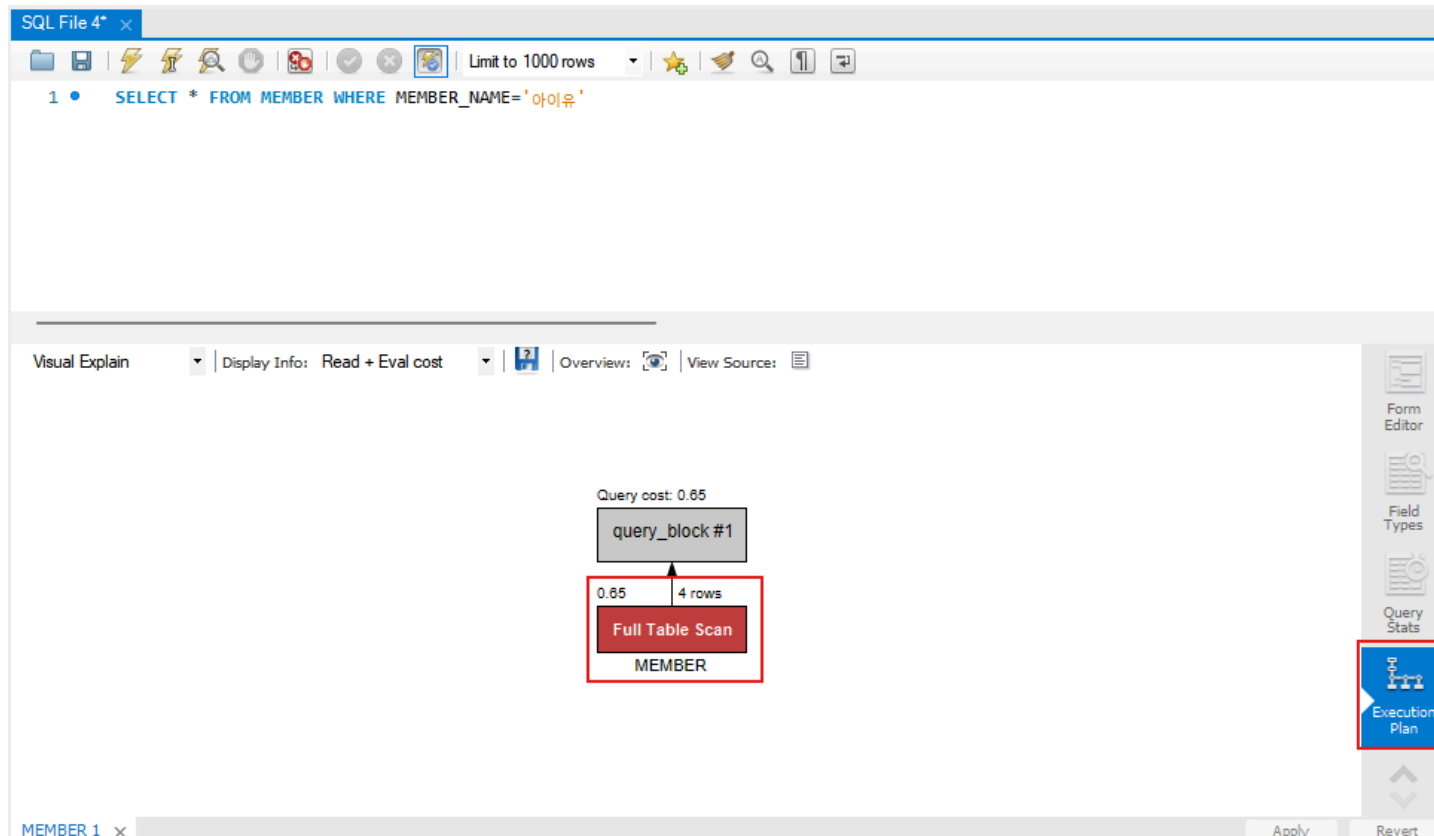
6장에서 인덱스에 대해 상세히 배우지만 먼저 여기서 간단한 실습을 통해 익혀보자.

1. MySQL을 실행하여 [File]→[Close Tab] 메뉴를 실행해서 쿼리 창이 열려 있지 않은 상태로 만든다. 새로운 쿼리 창을 여는 아이콘(좌측 상단 File 바로 아래)을 누르고 스키마 패널의 shop_db로 진행한다.



쿼리 창을 닫을 때 저장하겠냐고 물을 때 필요한 경우가 아니라면 저장할 필요없다.

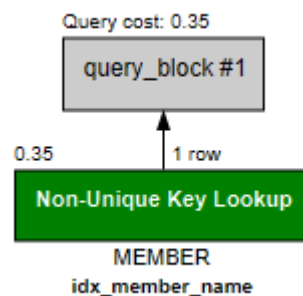
2. 회원 테이블에는 아직 인덱스가 만들어진 상태가 아니다. 테이블에서 아이디를 찾을 때는 회원 테이블의 1행부터 끝까지 전체를 살펴봐야 한다. **SELECT * FROM MEMBER WHERE MEMBER_NAME='아이유';** 문을 작성하고 실행한다. 결과는 해당 조건을 만족하는 데이터를 당연히 잘 찾았을 것이다. 어떻게 데이터를 찾았는지 확인하기 위해 **Result grid**의 맨 오른쪽에 있는 **[Execution plan]** 탭을 클릭하면 **Full Table Scan**이라고 나온다. 처음부터 끝까지 탐색을 해서 조건에 맞는 데이터를 찾은 것이다. 데이터가 많으면 많을수록 전체 테이블 탐색은 많은 시간을 소모할 것이다.



3. 이제 회원 테이블에 인덱스를 만들어본다. 다음 SQL 문을 실행하면 인덱스가 생성된다. 인덱스는 열에 지정한다. SQL의 마지막에 ON member (member_name)의 의미는 member 테이블의 member_name 열에 인덱스를 저장하라는 의미이다. 결과는 눈에 보이지 않지만 Output을 보면 실행된 것을 알 수 있을것이다

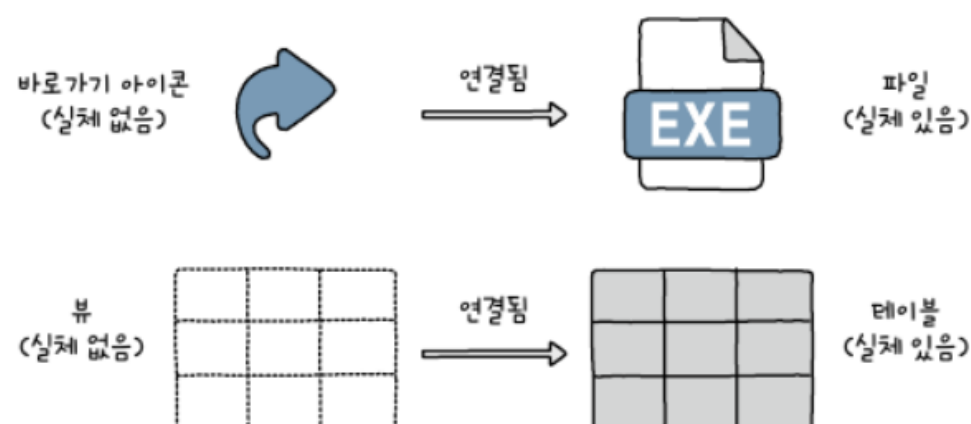
CREATE INDEX idx_member_name ON member (member_name) ;

4. 이제 인덱스가 생긴 회원 테이블에서 아이유를 찾아본다. 2번에서 사용한 SQL을 다시 실행해본다. 결과는 동일하지만 [Execution Plan] 탭을 보면 **Non-Unique Key Lookup**이라고 나온다. 자세한 의미는 6장에서 알아보고, Key Lookup은 인덱스를 통해 결과를 찾았다고 기억하면 된다. 이런 방법을 **인덱스 검색(Index Scan)**이라 한다.



뷰

뷰는 테이블과 상당히 동일한 성격의 데이터베이스 개체이다. 뷰를 활용하면 보안도 강화하고, SQL문도 간단하게 사용할 수 있다. 뷰를 한마디로 정의하면 가상의 테이블이라고 할 수 있다. 일반 사용자의 입장에서는 테이블과 뷰를 구분할 수 없다. 즉, 일반 사용자는 테이블과 동일하게 뷰를 취급하면 된다. 다만 뷰는 실제 데이터를 가지고 있지 않으며, 진짜 테이블에 링크된 개념이라고 생각하면 된다. 뷰는 윈도우 운영체제의 바로가기 아이콘과 비슷하다. 윈도우에서 바탕 화면의 바로가기를 누르면 실행되지만, 실제로 실행되는 파일은 다른 폴더에 있다. 뷰의 실체는 SELECT 문이다. 실습을 통해 뷰를 확인해보자.



뷰 실습

뷰도 SQL문을 통해 MySQL에서 생성할 수 있다.

- 1. 현재 진행 중인 쿼리 창을 닫고 새로운 쿼리 창을 열고 스키마 패널에서 shop_db를 선택한다. 먼저 기본적인 뷰를 만들어본다. 회원 테이블과 연결되는 회원 뷰(member_view)를 만들기 위해 다음 SQL을 실행한다. [Output] 패널에 초록색 체크 표시가 나타나면 SQL이 제대로 실행되었다는 의미이다.

```
CREATE VIEW member_view AS SELECT * FROM member;
```

- 2. 이제 회원 테이블이 아닌 회원 뷰에 접근해보자. 뷰에 접근하는 것은 테이블에 접근하는 것과 동일하다. 다음 SQL을 실행하면 회원 테이블에 접근했을 때와 동일한 결과가 나온다.

```
SELECT * FROM member_view;
```

동일한 결과임에도 테이블을 사용하지 않고 굳이 뷰를 사용하는 이유는 다음과 같다.

- 보안에 도움이 된다
- 긴 SQL 문을 간략하게 만들 수 있다.

이에 대한 설명은 5장에서 더 자세히 다룬다.

스토어드 프로시저

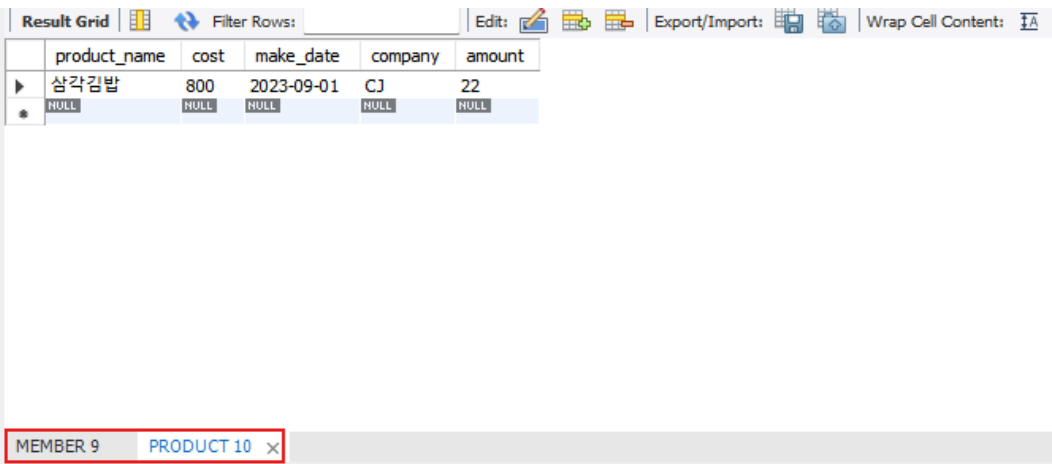
스토어드 프로시저란 MySQL에서 제공하는 **프로그래밍 기능**으로, 여러 개의 SQL문을 하나로 묶어서 편리하게 사용할 수 있게 해준다. SQL을 묶는 개념 외에 C, 자바, 파이썬과 같은 프로그래밍 언어에서 사용되는 연산식, 조건문, 반복문 등을 사용할 수도 있다. 스토어드 프로시저를 통해서 MySQL에서도 기본적인 형태의 일반 프로그래밍 로직을 코딩할 수 있다. 실습을 통해 알아보자.

스토어드 프로시저 실습

- 1. MySQL에서 이전 쿼리 창을 열려 있지 않은 상태로 만든다. 새로운 쿼리 창을 열고 스키마 패널에서 shop_db를 선택한다. 다음 두 SQL을 입력하고 한꺼번에 실행한다. 별도의 탭으로 동시에 결과가 나온다. 그런데 만약 이 두 SQL을 앞으로 계속 사용해야한다고 가정해보자. 매번 두 줄의 SQL을 입력해야 한다면 상당히 불편할 것이다.

```
SELECT * FROM member WHERE member_name = '나훈아';
```

```
SELECT * FROM product WHERE product_name = '삼각김밥';
```



- 2. 두 SQL을 하나의 스토어드 프로시저로 만들어본다. 다음 SQL을 입력하고 실행해본다. 첫 행과 마지막 행에 **구분 문자**라는 의미의 **DELIMITER // ~DELIMITER ;** 문이 있다. 일단 이것은 스토어드 프로시저를 묶는 약속이라고 생각하면 된다. 그리고 **BEGIN**과 **END** 사이에 SQL 문을 넣으면 된다.

```

1 DELIMITER //
2 CREATE PROCEDURE myProc()
3 BEGIN
4     SELECT * FROM MEMBER WHERE MEMBER_NAME = '나훈아';
5     SELECT * FROM PRODUCT WHERE PRODUCT_NAME = '삼각김밥';
6 END //
7 DELIMITER ;
8

```

myProc()은 스토어드 프로시저 이름을 지정한 것이다.

3. 이제부터는 두 줄의 SQL 문을 실행할 필요 없이 앞에서 만든 스토어드 프로시저를 호출하기 위해서 **CALL 문**을 실행하면 된다.

```

1 DELIMITER //
2 CREATE PROCEDURE myProc()
3 BEGIN
4     SELECT * FROM MEMBER WHERE MEMBER_NAME = '나훈아';
5     SELECT * FROM PRODUCT WHERE PRODUCT_NAME = '삼각김밥';
6 END //
7 DELIMITER ;
8
9 CALL myProc();

```

| product_name | cost | make_date | company | amount |
|--------------|------|------------|---------|--------|
| 삼각김밥 | 800 | 2023-09-01 | CJ | 22 |



CREATE 문과 DROP 문

테이블, 인덱스, 뷰, 스토어드 프로시저 등의 데이터베이스 개체를 만들기 위해서는 CREATE 개체_종류 개체_이름 ~~ 형식을 사용한다. 반대로 삭제하기 위해서는 DROP 개체_종류 개체_이름 형식을 사용한다.