

02

CHAPTER

데이터베이스 전체 운영 맛보기



Contents

01 데이터베이스 모델링

02 데이터베이스 구축

03 데이터베이스 개체 활용

학습목표

- 데이터베이스 모델링의 기본 개념을 이해한다.
- 데이터베이스에서 사용하는 주요 용어를 이해한다.
- 데이터베이스 구축 절차를 실습한다.
- 테이블, 뷰, 인덱스, 스토어드 프로시저, 트리거의 개념을 이해한다.

1-1 정보 시스템 구축의 개요

- 정보 시스템 구축 단계
 - 5단계 : 분석-설계-구현-시험-유지 · 보수
- 요구 사항 분석
 - '무엇을(what)' 할지 결정하는 것
 - 사용자 인터뷰와 업무 조사 등을 수행
- 시스템 설계
 - 구축하고자 하는 시스템을 '어떻게(how)' 설계할 것인지 결정하는 것
- 대부분의 프로젝트에서는 분석과 설계 단계가 전체 공정의 50% 이상을 차지

1-2 데이터베이스 모델링과 필수 용어

■ 데이터베이스 모델링

- 현실 세계에서 사용되는 데이터를 MySQL에 어떻게 옮겨놓을지 결정하는 과정
- 예 : 쇼핑몰 데이터베이스(그림 2-1)

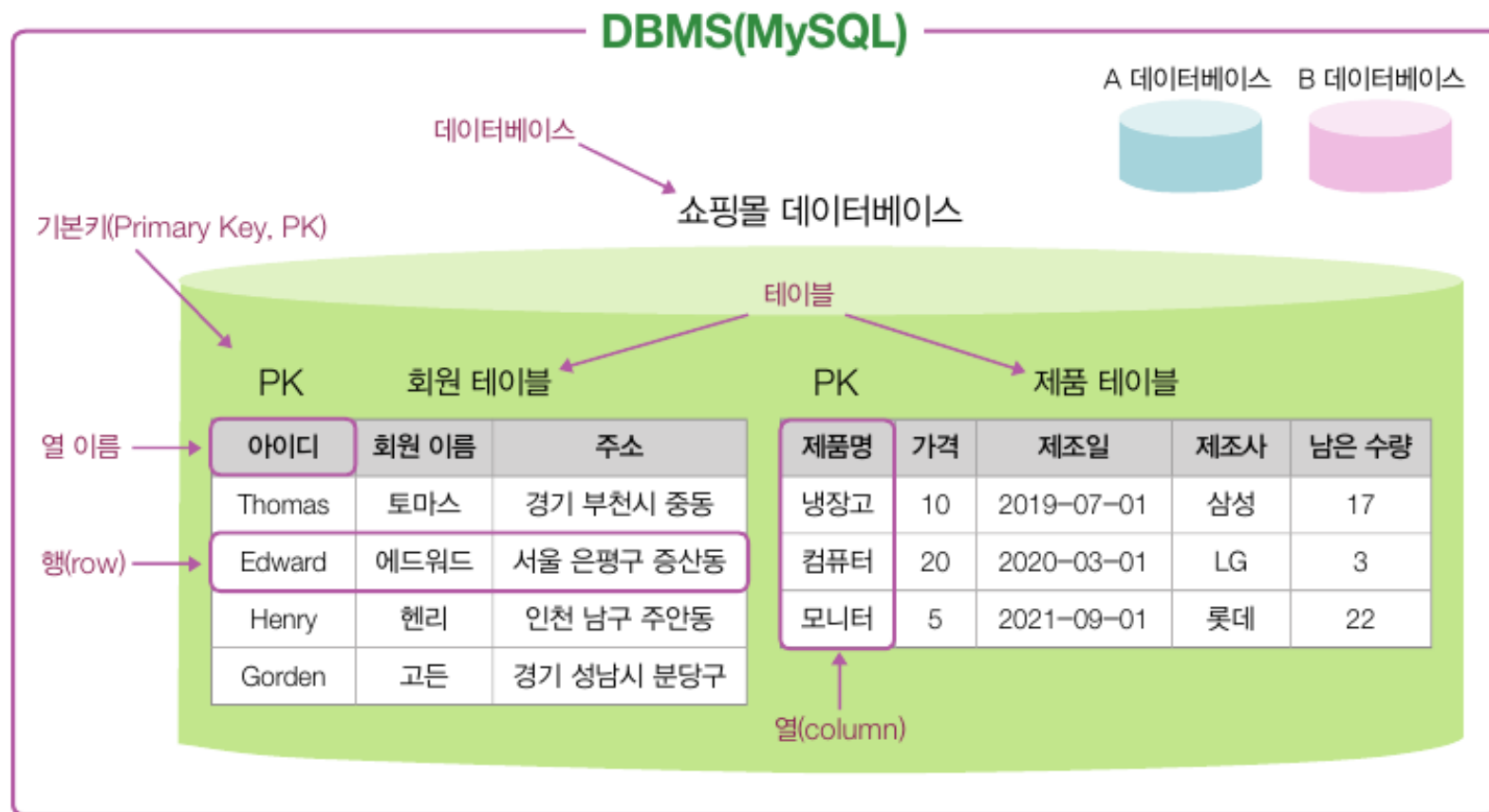


그림 2-1 테이블의 구조와 관련 용어

1-2 데이터베이스 모델링과 필수 용어

■ 데이터베이스 필수 용어

- **데이터** : 테이블에 저장된 하나하나의 단편적인 정보
- **테이블** : 데이터를 입력하기 위해 표 형태로 만든 것
- **데이터베이스** : 테이블이 저장되는 저장소로, 원통 모양으로 표현
- **DBMS** : DataBase Management System의 약자로, 데이터베이스를 관리하는 시스템 또는 소프트웨어
- **열(필드)** : 각 테이블을 열로 구성
- **열 이름** : 각 열을 구분하기 위한 이름
- **데이터 형식** : 열의 데이터 형식
- **행(레코드)** : 실질적인 데이터
- **기본키(주키)** : 각 행을 구분하는 유일한 열로, 기본키는 중복되어서도 비어 있어서도 안 됨
- **외래키** : 두 테이블의 관계를 맺어주는 키
- **SQL(구조화된 질의 언어)** : 사람과 DBMS가 소통하기 위한 말(언어)

2-1 데이터베이스 구축 절차 요약

■ 데이터베이스 구축 절차

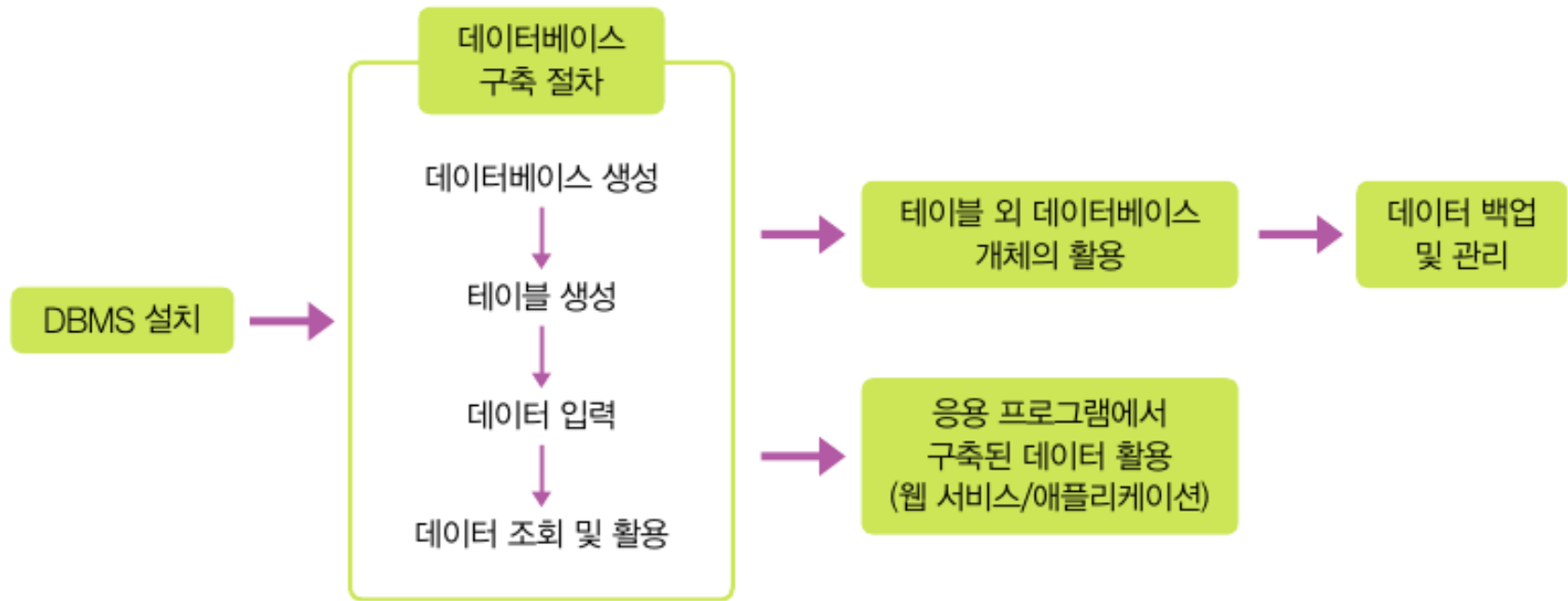


그림 2-2 데이터베이스 구축/관리 및 활용의 전반적인 절차

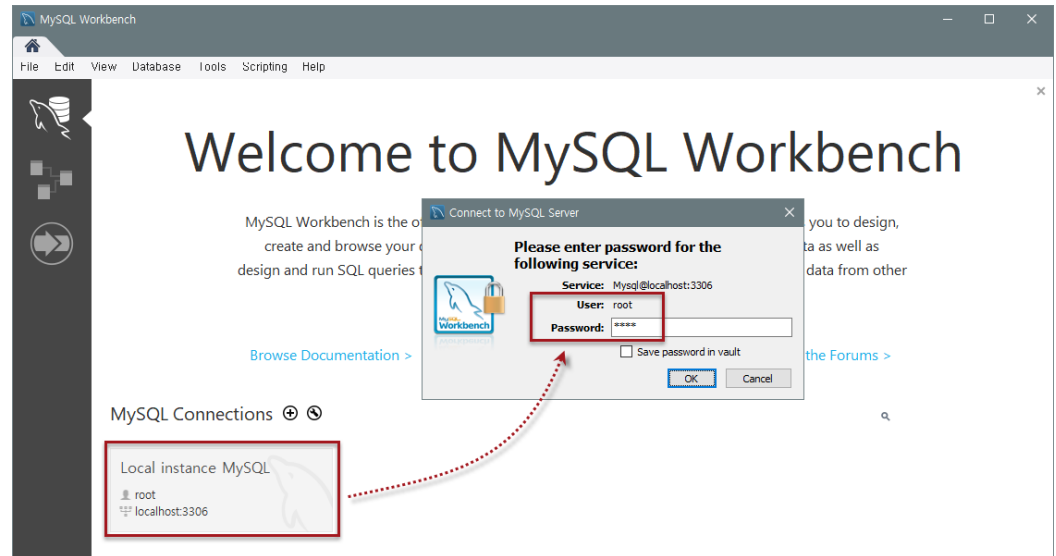
[실습 2-1] 쇼핑몰 데이터베이스(shopDB) 생성하기

교재 52~57p 참고

1 Workbench 실행하기

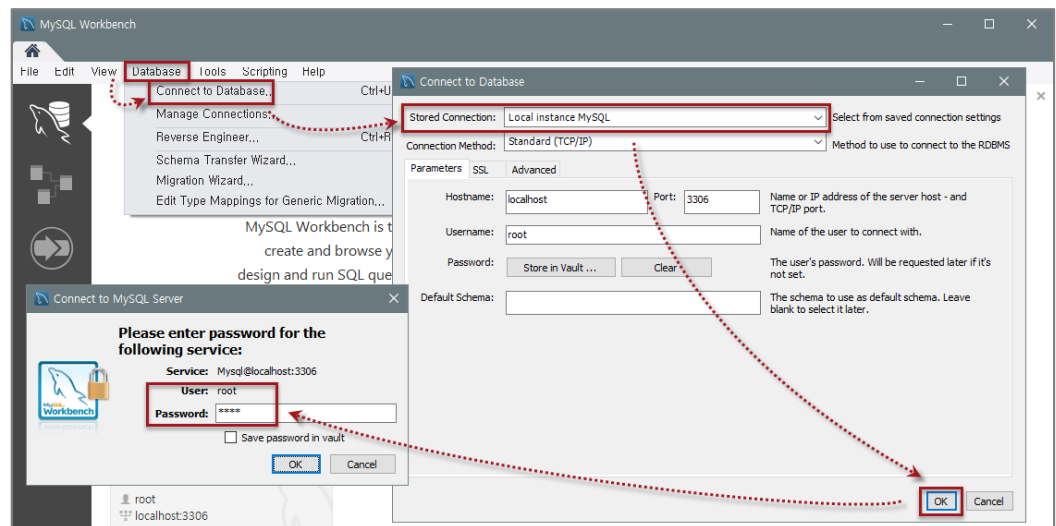
1-1 Workbench 실행

1-2 MySQL 서버에 연결



1-3 Cannot Connect to Database

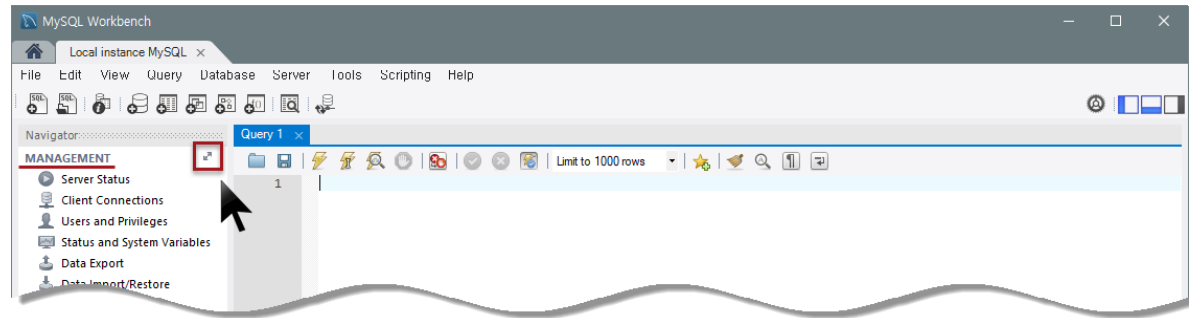
Server 오류 발생 시 대응 방법



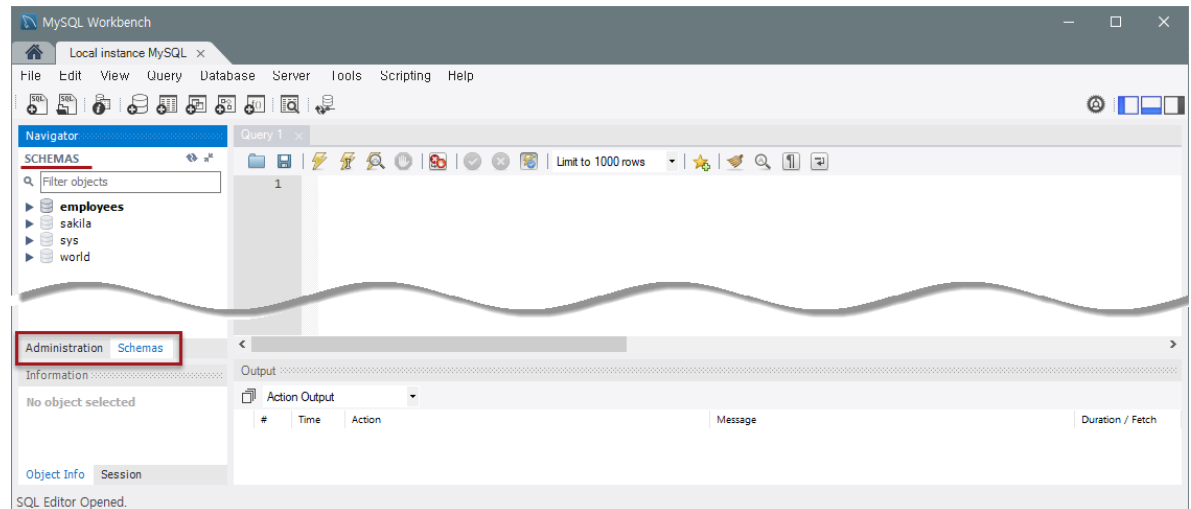
[실습 2-1] 쇼핑몰 데이터베이스(shopDB) 생성하기

교재 52~57p 참고

1-4 [Navigator]를 탭에 있는
확대/축소 아이콘 클릭



1-5 [SCHEMAS] 탭 클릭

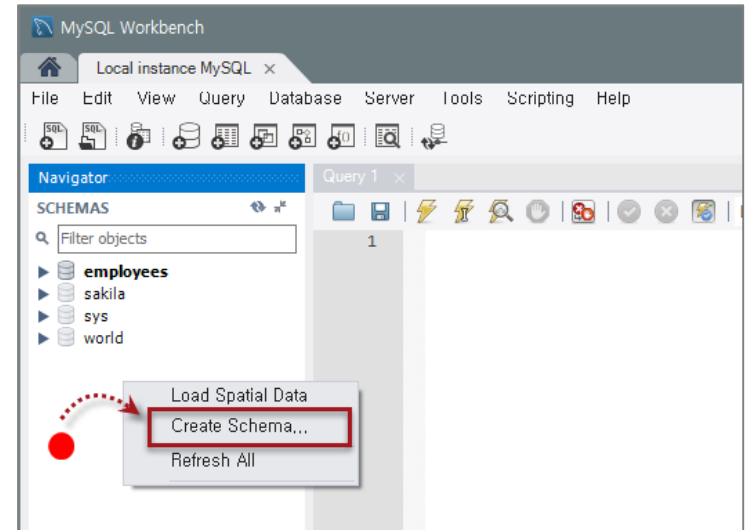


[실습 2-1] 쇼핑몰 데이터베이스(shopDB) 생성하기

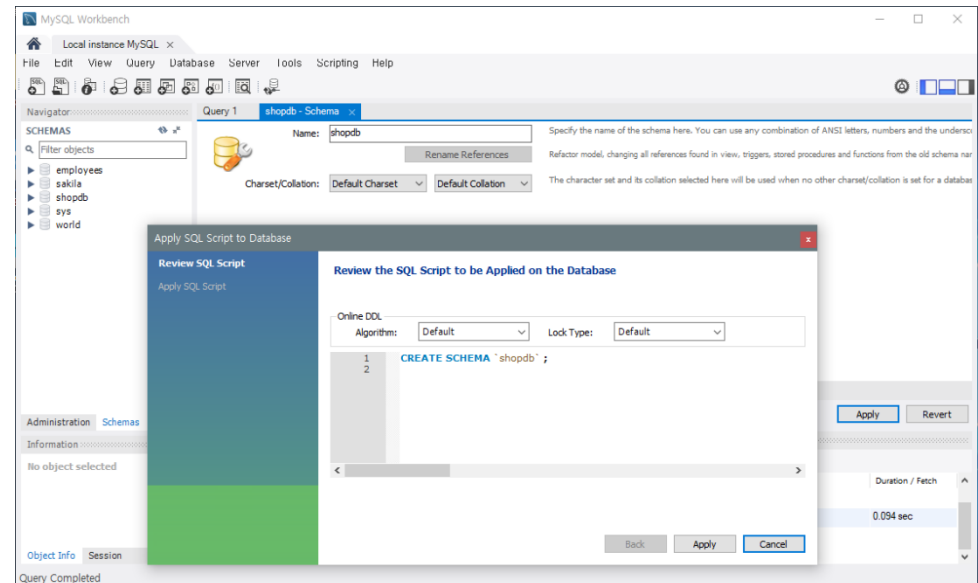
교재 52~57p 참고

2 쇼핑몰 데이터베이스(ShopDB) 생성하기

2-1 데이터베이스 생성



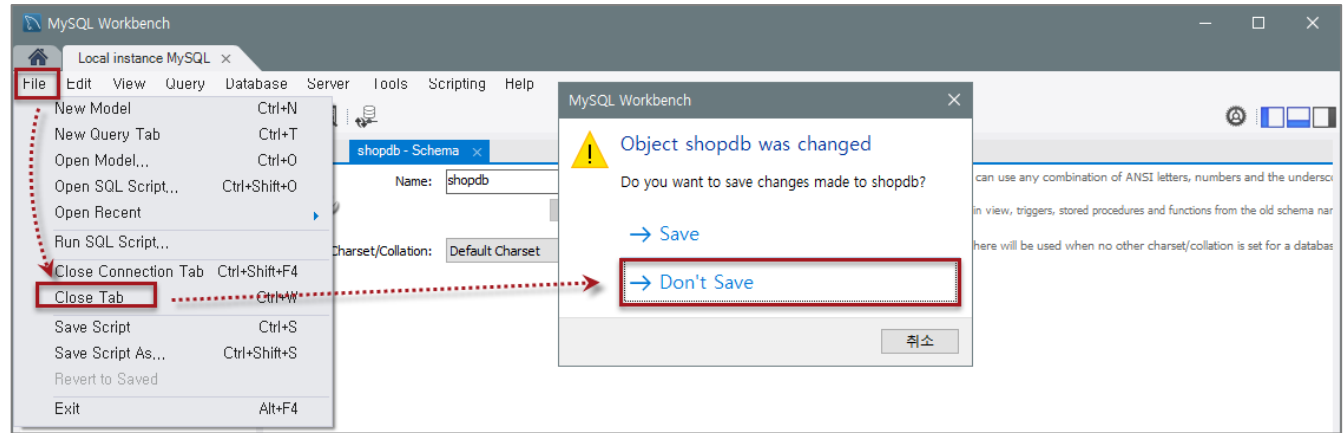
2-2 ShopDB 데이터베이스(스키마) 추가



[실습 2-1] 쇼핑몰 데이터베이스(shopDB) 생성하기

교재 52~57p 참고

2-3 탭 닫기



1 개체 이름 정하기

1-1 [그림 2-1]에는 나타나 있지 않은 각 열의 영문 이름과 데이터 형식을 결정해야 함

1-2 회원 테이블(memberTBL)의 데이터 형식 지정

표 2-1 회원 테이블 정의

열 이름(한글)	영문 이름	데이터 형식	길이	NULL 허용
아이디	memberID	문자(CHAR)	8글자(영문)	×
회원 이름	memberName	문자(CHAR)	5글자(한글)	×
주소	memberAddress	문자(CHAR)	20글자(한글)	○

1-3 품 테이블(productTBL)의 데이터 형식 지정

표 2-2 제품 테이블 정의

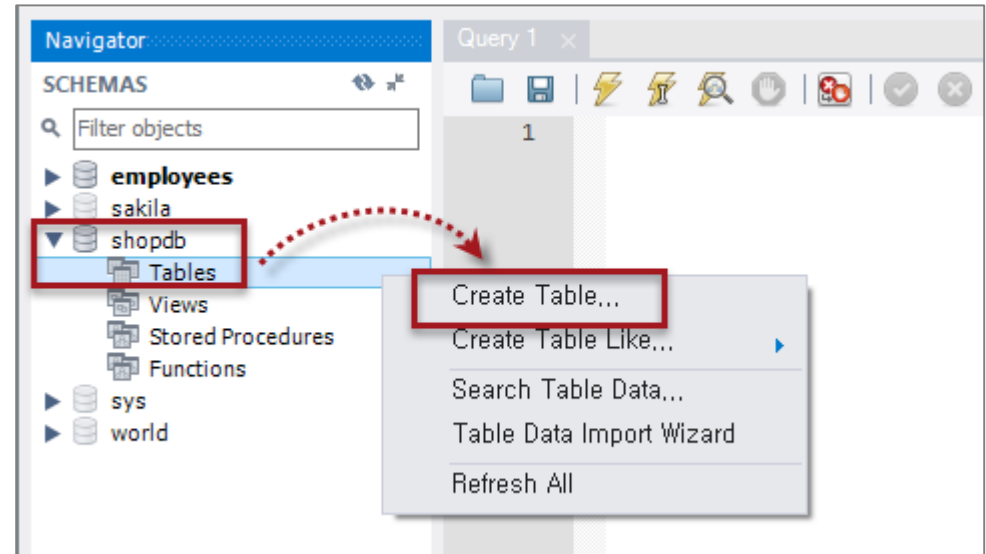
열 이름(한글)	영문 이름	데이터 형식	길이	NULL 허용
제품명	productName	문자(CHAR)	4글자(한글)	×
가격	cost	숫자(INT)	정수	×
제조일	makeDate	날짜(DATE)	날짜형	○
제조사	company	문자(CHAR)	5글자(한글)	○
남은 수량	amount	숫자(INT)	정수	×

[실습 2-2] 테이블 생성하기

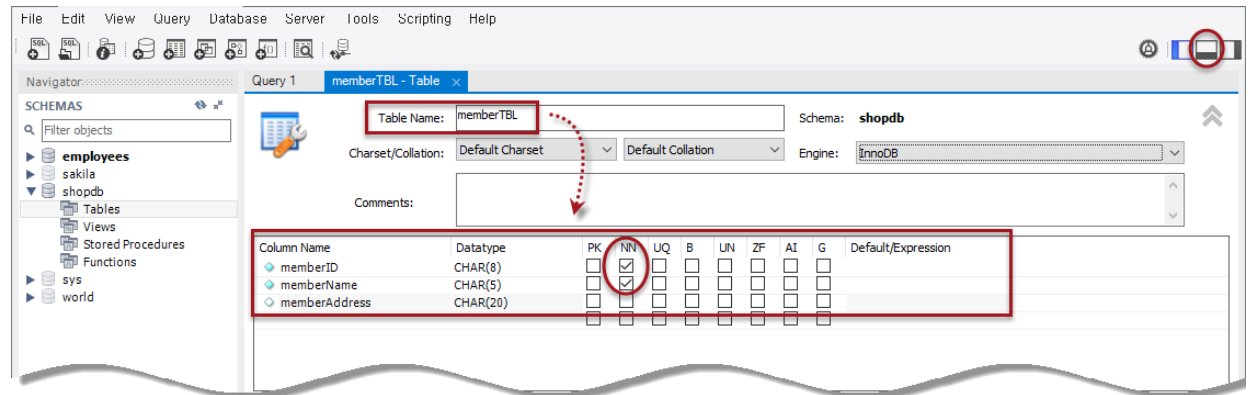
교재 57~61p 참고

2 회원 테이블(memberTBL) 만들기

2-1 [Create Table] 선택



2-2 회원 테이블 내용 입력



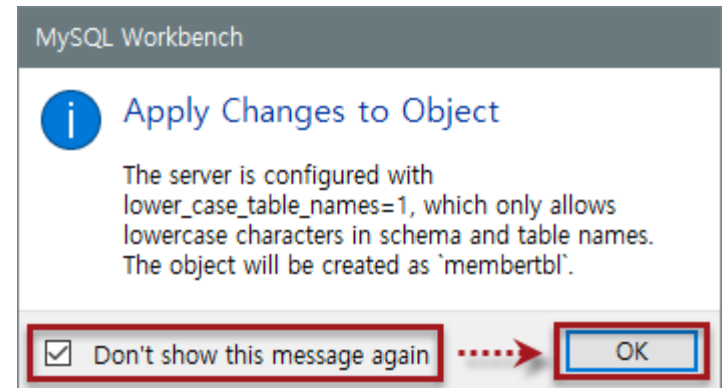
[실습 2-2] 테이블 생성하기

교재 57~61p 참고

2-3 기본키 지정

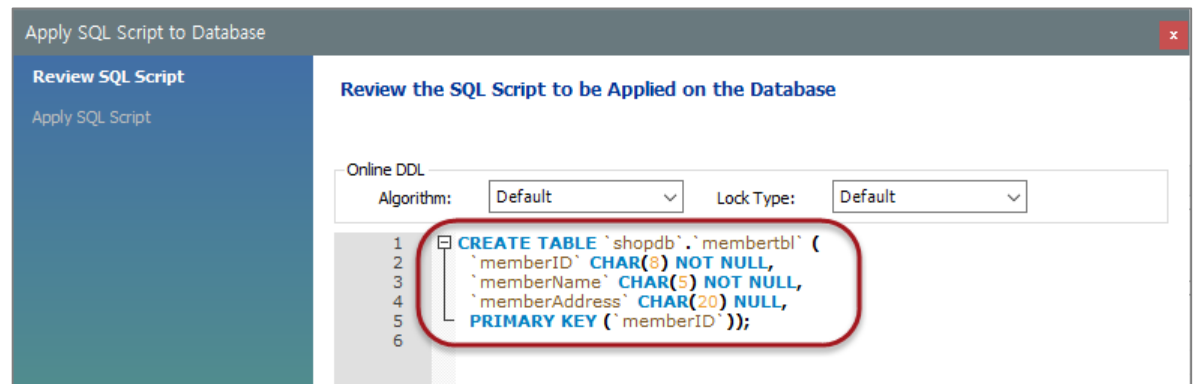
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
memberID	CHAR(8)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
memberName	CHAR(5)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
memberAddress	CHAR(20)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

2-4 <Apply> 클릭



2-5 테이블 생성 완료

2-6 테이블 생성 창 닫기



[실습 2-2] 테이블 생성하기

교재 57~61p 참고

3 제품 테이블(productTBL) 만들기

3-1 제품 테이블 생성

Query 1 productTBL - Table

Table Name: productTBL Schema: shopdb

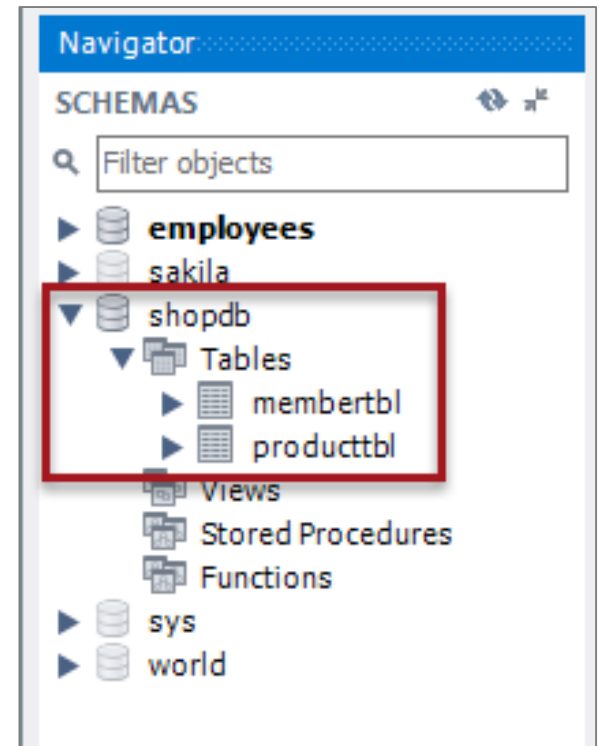
Charset/Collation: Default Charset Default Collation Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
productName	CHAR(4)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
cost	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
makeDate	DATE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
company	CHAR(5)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
amount	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

4 생성한 테이블 확인하기

4-1 테이블 2개 생성 확인

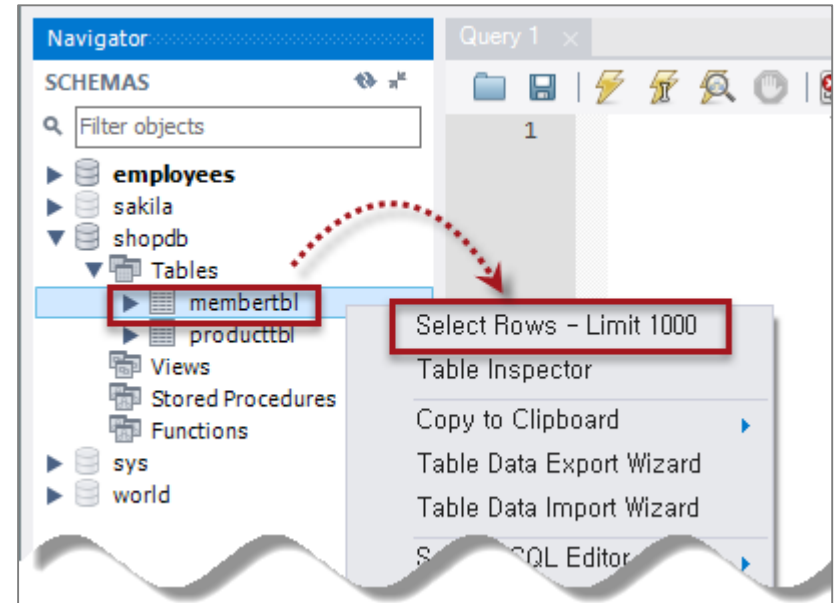


[실습 2-3] 행 데이터 입력하기

교재 61~63p 참고

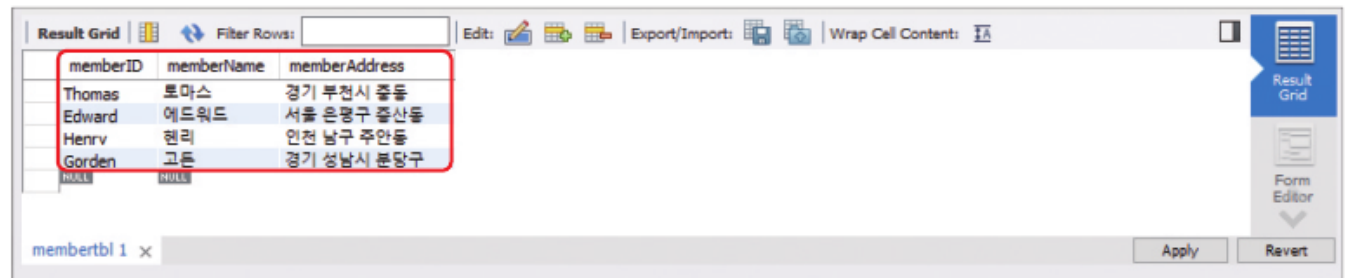
1 회원 테이블의 데이터 입력하기

1-1 [Select Rows - Limit 1000] 선택



1-2 데이터 입력

1-3 창 닫기



[실습 2-3] 행 데이터 입력하기

교재 61~63p 참고

2 제품 테이블의 데이터 입력하기

2-1 데이터 입력

2-2 창 닫기

Result Grid			Filter Rows:	<input type="text"/>	Edit:				Export/Import:		
	productName	cost	makeDate	company	amount						
	냉장고	10	2019-07-01	삼성	17						
	컴퓨터	20	2020-03-01	LG	3						
	모니터	5	2021-09-01	롯데	22						
	NULL	NULL	NULL	NULL							

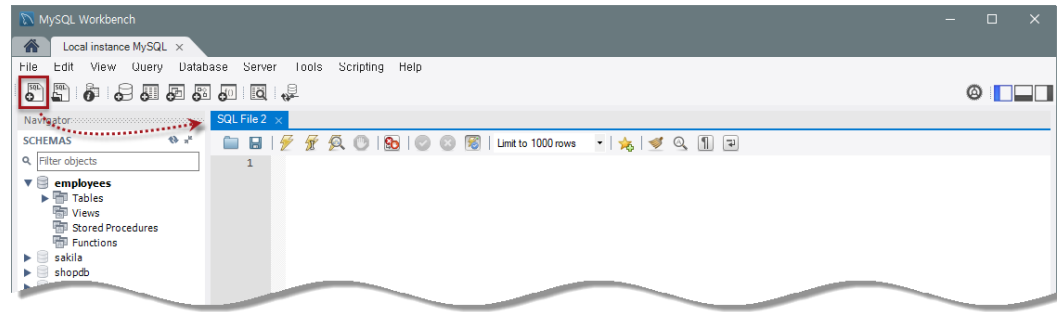
[실습 2-4] SQL 문 작성하기

교재 63~68p 참고

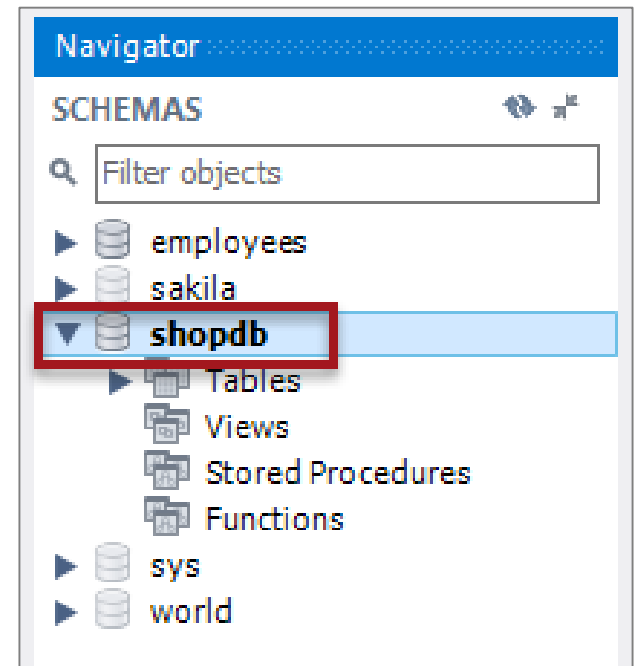
1 쿼리 창 열기

1-1 열려 있는 쿼리 창이 있으면 모두 닫기

1-2 새 쿼리 창 열기



1-3 사용할 데이터베이스 선택



[실습 2-4] SQL 문 작성하기

교재 63~68p 참고

2 SELECT 문 작성하기

2-1 회원 테이블의 모든 데이터 조회

```
SELECT * FROM memberTBL;
```

The screenshot shows the MySQL Workbench interface. The 'SQL File 3*' tab is active, displaying the query `SELECT * FROM memberTBL;`. The 'Result Grid' at the bottom shows the data retrieved from the `memberTBL` table in the `shopdb` schema. The table has three columns: `memberID`, `memberName`, and `memberAddress`. The data includes members Edward, Gordon, Henry, and Thomas. The 'Object Info' pane at the bottom left shows the `memberTBL` table selected.

memberID	memberName	memberAddress
Edward	에드워드	서울 은평구 증산동
Gordon	고든	경기 성남시 분당구
Henry	헨리	인천 남구 주안동
Thomas	토마스	경기 부천시 중동

[실습 2-4] SQL 문 작성하기

교재 63~68p 참고

2-2 회원 테이블의 이름과 주소만 출력

```
SELECT memberName, memberAddress FROM memberTBL;
```

The screenshot shows a SQL IDE interface. At the top, a tab labeled 'SQL File 3*' is open. Below the toolbar, a list on the left shows line numbers 1 and 2. Line 2 contains the SQL query: `SELECT memberName, memberAddress FROM memberTBL;`, which is highlighted with a blue background and a red underline. Below the query editor, the 'Result Grid' is displayed, showing a table with two columns: 'memberName' and 'memberAddress'. The table contains four rows of data. To the right of the table, there are buttons for 'Result Grid' and 'Form Editor'. At the bottom left, a tab labeled 'memberTBL 3' is visible. At the bottom right, there is a 'Read Only' status indicator.

memberName	memberAddress
에드워드	서울 은평구 증산동
고든	경기 성남시 분당구
헨리	인천 남구 주안동
토마스	경기 부천시 중동

[실습 2-4] SQL 문 작성하기

교재 63~68p 참고

2-3 '토마스'에 대한 정보만 추출

```
SELECT * FROM memberTBL WHERE memberName = '토마스';
```

The screenshot shows a SQL IDE window titled "SQL File 3* x". The query editor contains two SQL statements:

```
1  
2 • SELECT memberName, memberAddress FROM memberTBL;  
3 • SELECT * FROM memberTBL WHERE memberName = '토마스' ;
```

The second statement is highlighted with a red box. Below the query editor, the "Result Grid" is displayed, showing the results of the query:

memberID	memberName	memberAddress
Thomas	토마스	경기 부천시 중동
NULL	NULL	NULL

At the bottom of the IDE, the file explorer shows two files: "memberTBL 4" and "memberTBL 5 x". Red dashed arrows point from the first and second lines of the SQL query to these files respectively.

[실습 2-4] SQL 문 작성하기

교재 63~68p 참고

2-4 두 번째 쿼리 부분만 실행

SQL File 3*

1
2 • SELECT memberName, memberAddress FROM memberTBL;
3 • SELECT * FROM memberTBL WHERE memberName = '토마스';

Result Grid

memberID	memberName	memberAddress
Thomas	토마스	경기 부천시 중동

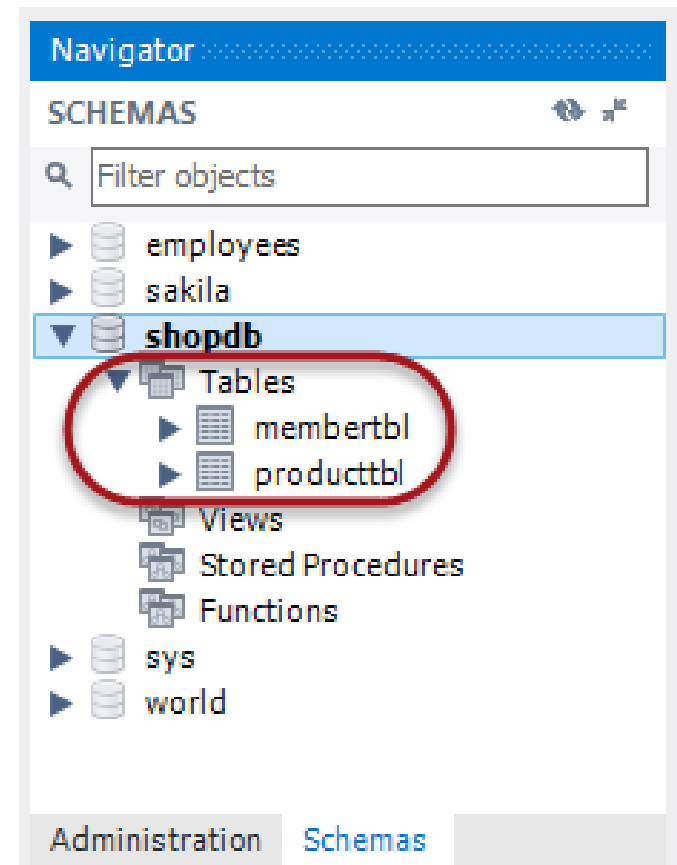
memberTBL 6

3 SQL 문으로 새로운 테이블 생성하기

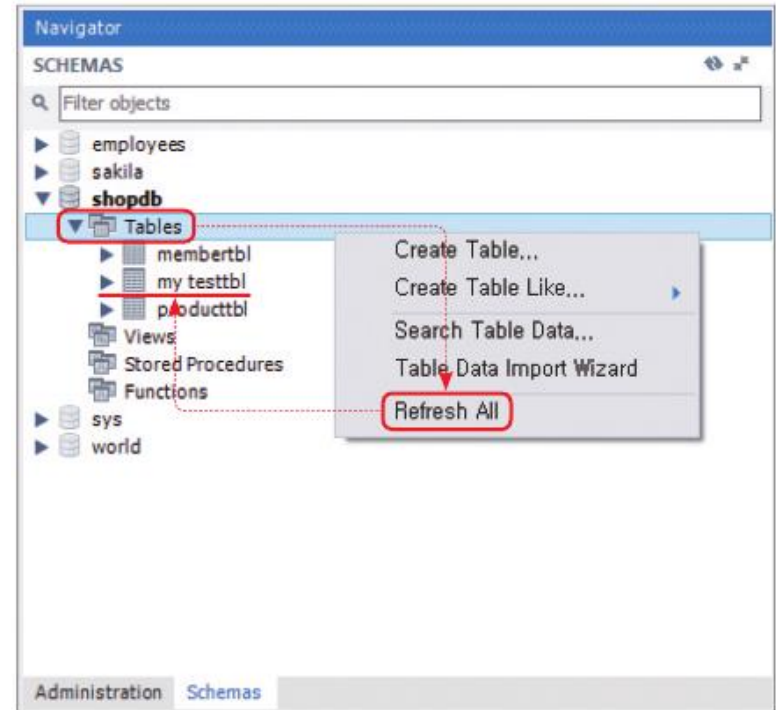
3-1 간단한 테이블을 생성하는 SQL 문 실행

```
CREATE TABLE `my testTBL` (id INT);
```

3-2 [Navigator]에서 방금 생성한 'my testTBL' 확인



3-3 [Navigator]에서 [Refresh All]을 선택한 후 확인



4 테이블 삭제하기

4-1 DROP TABLE 문을 사용하여 테이블 삭제

```
DROP TABLE `my TestTBL`;
```


3-2 인덱스

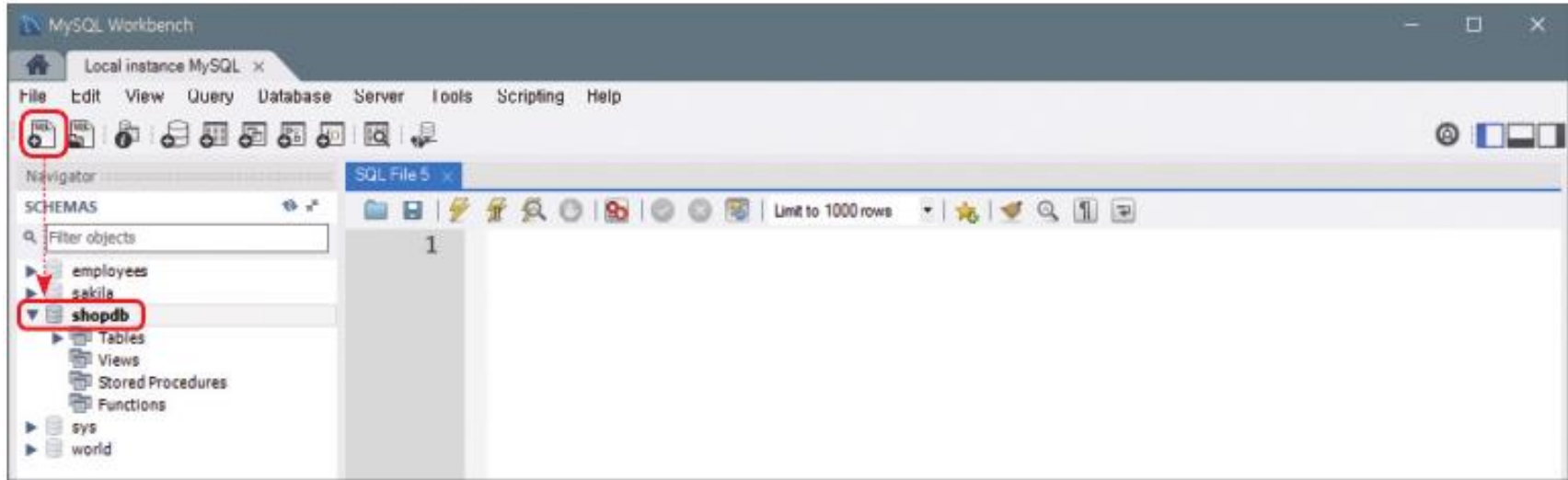
- 인덱스(index)
 - 실무에서 사용하는 데이터는 많게는 수천만, 수억 건 이상에 달하므로 인덱스 없이 전체 데이터를 찾는다는 것은 굉장히 부담스러운(시간이 오래 걸리는) 일
 - 인덱스는 책의 뒷부분에 실리는 '찾아보기(색인)'와 같음
 - 인덱스는 테이블의 열 단위에 생성됨

[실습 2-5] 인덱스 사용하기

교재 70~73p 참고

1 적정량의 데이터가 있는 테이블 생성하기

1-1 현재 데이터베이스를 ShopDB로 변경



1-2 500건의 데이터가 있는 indexTBL 생성

```
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;  
SELECT * FROM indexTBL;
```

Result Grid

	first_name	last_name	hire_date
	Georgi	Facello	1986-06-26
	Bezalel	Simmel	1985-11-21
	Parto	Bamford	1986-08-28
	Chirstian	Koblick	1986-12-01
	Kvoichi	Maliniak	1989-09-12
	Anneke	Preusis	1989-06-02
	Tzvetan	Zielinski	1989-02-10

indexTBL 3 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 1	10:51:03	CREATE TABLE indexTBL (first_name varchar(14), last_name va...	0 row(s) affected	0.125 sec
✓ 2	10:51:03	INSERT INTO indexTBL SELECT first_name, last_name, hire_da...	500 row(s) affected Records: 500 Duplicates: 0 Warnings: 0	0.031 sec
✓ 3	10:51:04	SELECT * FROM indexTBL LIMIT 0, 1000	500 row(s) returned	0.015 sec / 0.000 sec

[실습 2-5] 인덱스 사용하기

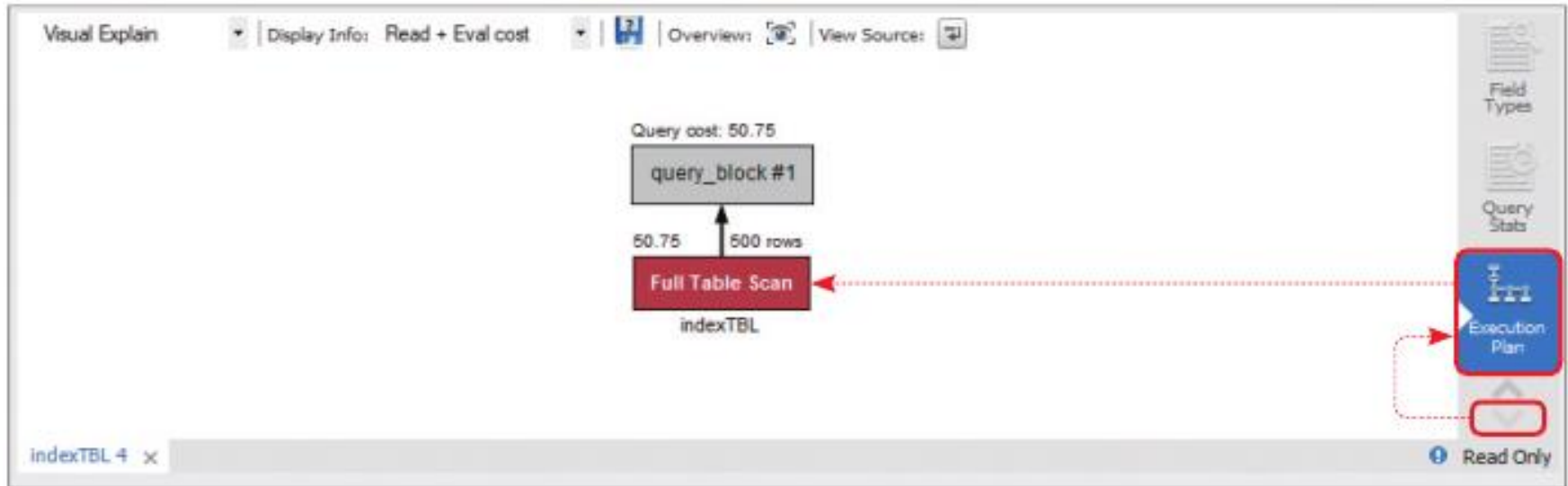
교재 70~73p 참고

2 인덱스가 없는 상태에서 쿼리 작동 확인하기

2-1 indexTBL에서 이름이 'Mary'인 사람을 조회

```
SELECT * FROM indexTBL WHERE first_name = 'Mary';
```

2-2 실행 계획 확인(인덱스를 사용하지 않고 테이블 전체를 검색(scan)함)



3 인덱스 생성 후 쿼리 작동 확인하기

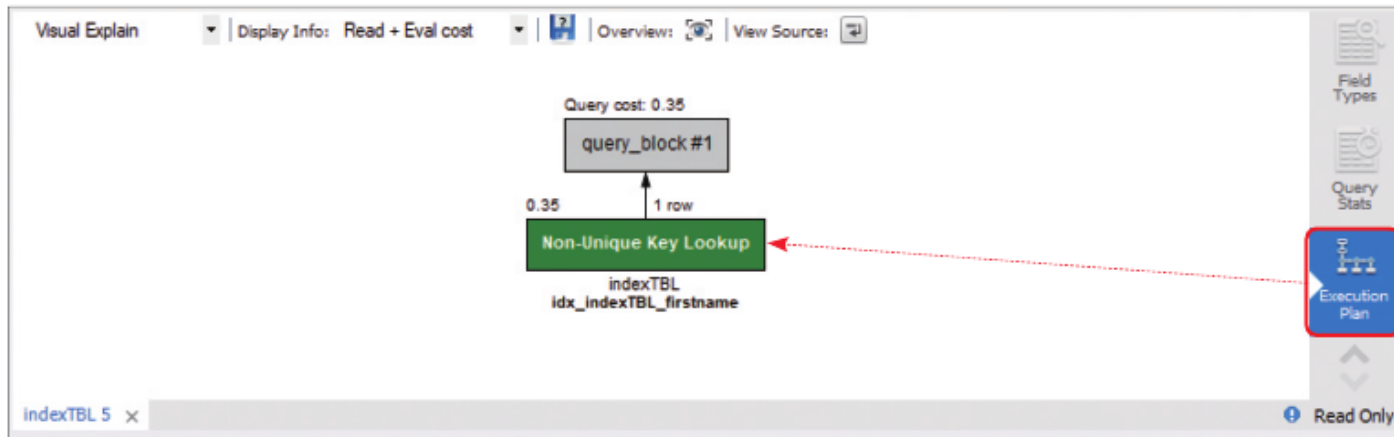
3-1 테이블(indexTBL)의 이름(first_name) 열에 인덱스 생성

```
CREATE INDEX idx_indexTBL_firstname ON indexTBL(first_name);
```

3-2 다시 검색

```
SELECT * FROM indexTBL WHERE first_name = 'Mary';
```

3-3 결과는 동일하게 1건 출력(인덱스 생성 전과 후의 내부적 작동은 큰 차이가 있음)



3-4 결론적으로 인덱스를 생성하기 전인 2-1의 쿼리는 책의 찾아보기가 없는 상태에서 특정 단어를 검색하는 것(책의 전체 페이지를 찾아보는 것)과 같고, 인덱스를 생성한 후인 3-2의 쿼리는 책의 찾아보기가 있을 때 먼저 찾아보기에서 특정 단어를 찾아보고 그 페이지를 펴서 검색하는 것과 같음

3-3 뷰

- 뷰(view)
 - 가상의 테이블
 - 실체가 없고 진짜 테이블에 연결(link)된 개념
 - 뷰를 SELECT 문으로 조회하면 진짜 테이블의 데이터를 조회하는 것과 동일한 결과가 나옴

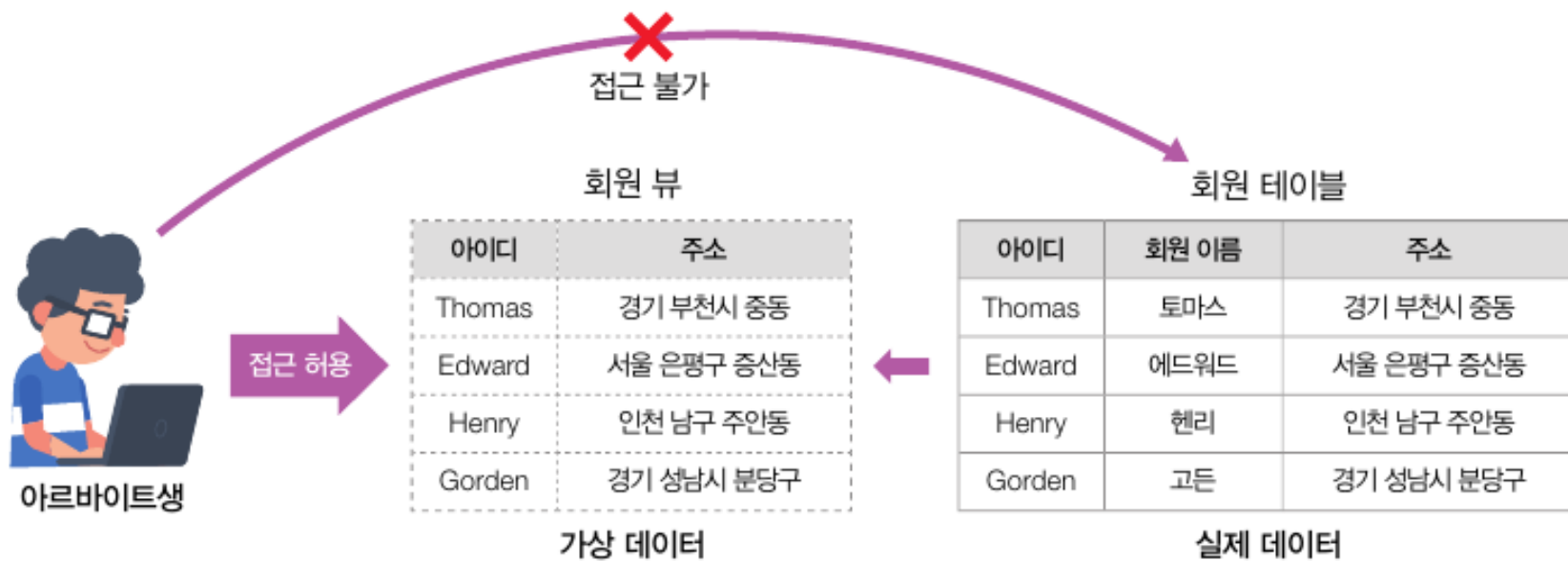


그림 2-36 뷰의 사용 예

[실습 2-6] 기본적인 뷰 사용법 알아보기

교재 74~76p 참고

1 현재 데이터베이스를 ShopDB로 변경하기

1-1 현재 데이터베이스를 ShopDB로 변경

2 뷰 생성하기

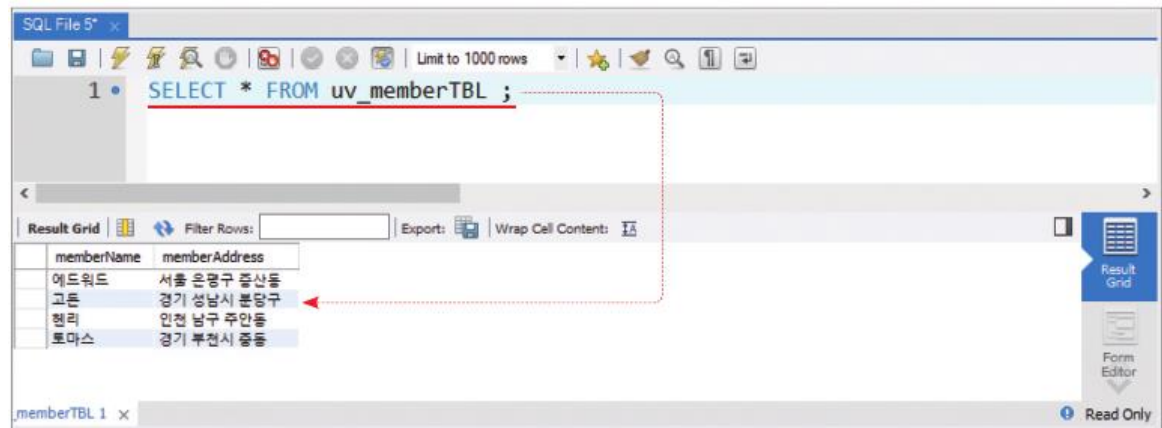
2-1 회원 이름과 주소만 있는 뷰 생성

```
CREATE VIEW uv_memberTBL
AS
SELECT memberName, memberAddress FROM memberTBL;
```

3 뷰 조회하기

3-1 아르바이트생의 입장에서 뷰(uv_memberTBL) 조회

```
SELECT * FROM uv_memberTBL;
```



3-4 스토어드 프로시저

- 스토어드 프로시저(stored procedure, 저장 프로시저)
 - SQL 문을 하나로 묶어 편리하게 사용하는 기능

[실습 2-7] 간단한 스토어드 프로시저 만들기

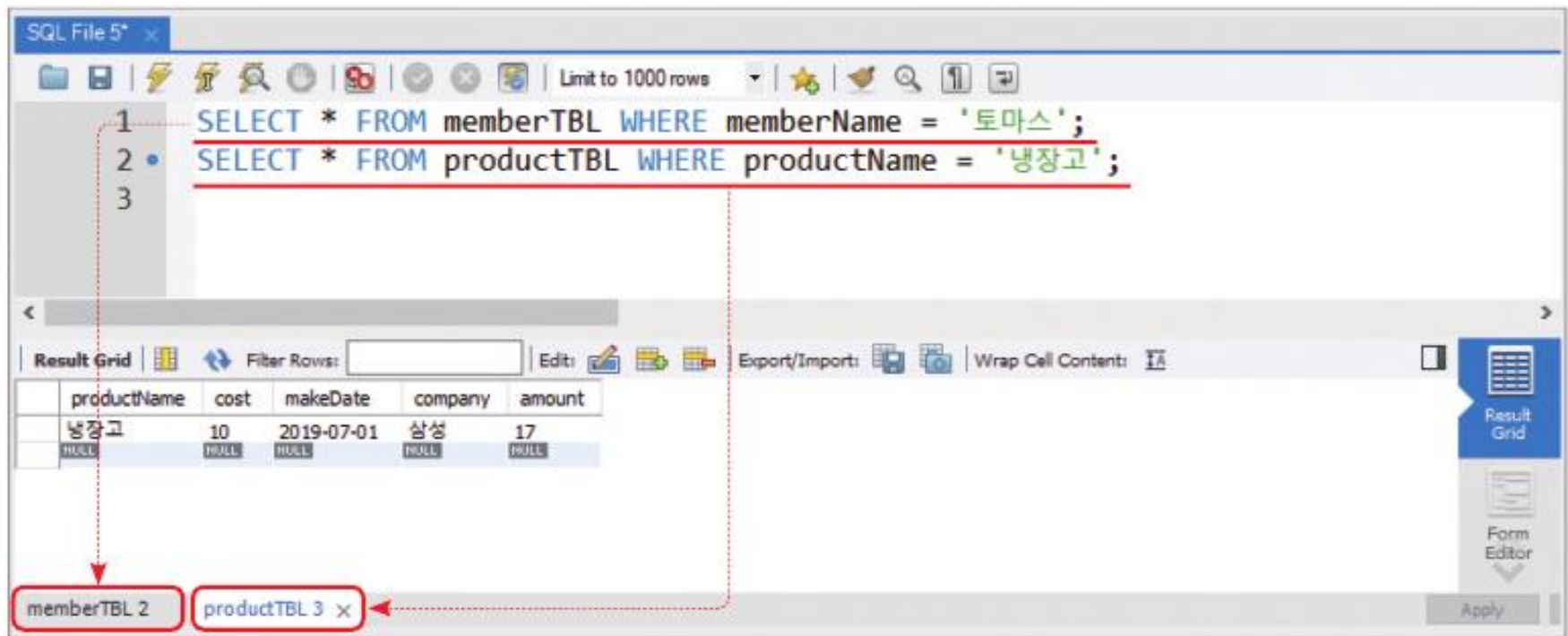
교재 76~77p 참고

1 2개의 쿼리를 각각 실행하기

1-1 현재 데이터베이스가 ShopDB인지 확인

1-2 SQL문 두 줄 입력과 실행

```
SELECT * FROM memberTBL WHERE memberName = '토마스';  
SELECT * FROM productTBL WHERE productName = '냉장고';
```



[실습 2-7] 간단한 스토어드 프로시저 만들기

교재 76~77p 참고

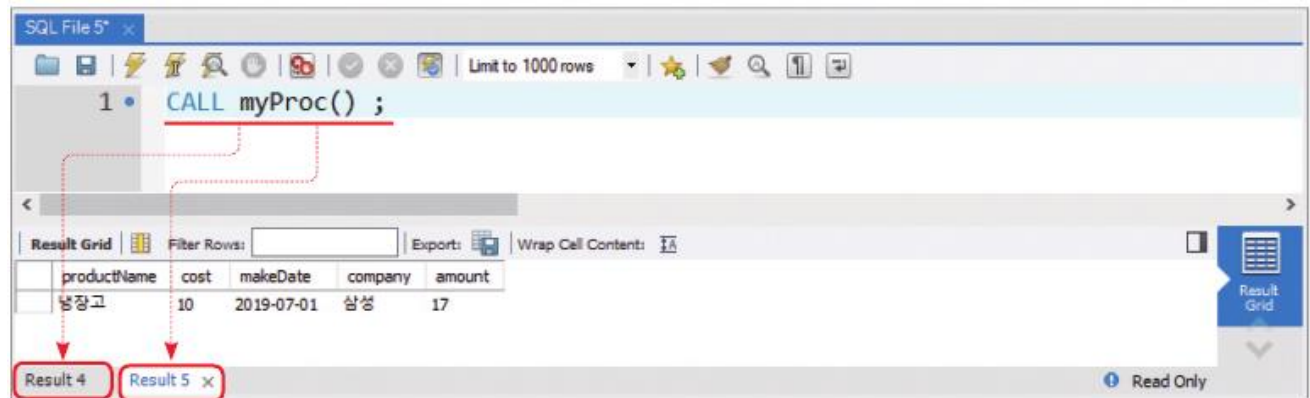
2 2개의 쿼리를 하나의 스토어드 프로시저로 만들기

2-1 myProc()라는 이름의 스토어드 프로시저 생성

```
DELIMITER //  
CREATE PROCEDURE myProc()  
BEGIN  
    SELECT * FROM memberTBL WHERE memberName = '토마스';  
    SELECT * FROM productTBL WHERE productName = '냉장고';  
END //  
DELIMITER ;
```

2-2 스토어드 프로시저 실행

```
CALL myProc();
```



3-5 트리거

- 트리거(trigger)
 - 테이블에 부착되어 테이블에 INSERT(삽입), UPDATE(수정), DELETE(삭제) 작업이 발생하면 실행되는 코드
 - 회원 탈퇴 시 간단히 회원 테이블(memberTBL)에서 토마스의 정보를 삭제하면(토마스의 행 데이터를 지우면) 되지만, 이렇게 하면 토마스가 회원 탈퇴를 한 사람인지 나중에 알 길이 없음
 - 트리거를 작성하면 회원 테이블(memberTBL)에서 삭제 작업이 일어날 때마다 다른 곳에 그 데이터를 '자동으로' 저장하여 편리함

[실습 2-8] 가장 일반적으로 사용되는 트리거의 용도 알아보기

교재 78~80p 참고

1 데이터를 삽입, 수정, 삭제하는 SQL 문 작성하기

1-1 현재 데이터베이스가 ShopDB인지 확인

1-2 새로운 회원 'Soccer/홍민/서울시 서대문구 북가좌동' 삽입

```
INSERT INTO memberTBL VALUES ('Soccer', '홍민', '서울시 서대문구 북가좌동');
```

1-3 '홍민'인 회원의 주소를 '서울 강남구 역삼동'으로 수정

```
UPDATE memberTBL SET memberAddress = '서울 강남구 역삼동' WHERE memberName = '홍민';
```

1-4 DELETE 문으로 회원 테이블에서 홍민 정보 삭제

```
DELETE FROM memberTBL WHERE memberName = '홍민';
```

2 다른 테이블에 삭제된 데이터와 삭제된 날짜 기록하기

2-1 삭제된 데이터를 보관할 테이블(deletedMemberTBL) 생성

```
CREATE TABLE deletedMemberTBL
( memberID char(8),
  memberName char(5),
  memberAddress char(20),
  deletedDate date -- 삭제한 날짜
);
```

2-2 삭제된 데이터가 기록되는 트리거 생성

```
DELIMITER //
CREATE TRIGGER trg_deletedMemberTBL -- 트리거 이름
  AFTER DELETE -- 삭제 후에 작동하게 지정
  ON memberTBL -- 트리거를 부착할 테이블
  FOR EACH ROW -- 각 행마다 적용
BEGIN
  -- OLD 테이블의 내용을 백업 테이블에 삽입
  INSERT INTO deletedMemberTBL
    VALUES (OLD.memberID, OLD.memberName, OLD.memberAddress, CURDATE());
END //
DELIMITER ;
```

[실습 2-8] 가장 일반적으로 사용되는 트리거의 용도 알아보기

교재 78~80p 참고

3 회원 테이블의 데이터 삭제 후 삭제된 데이터가 백업 테이블에 들어가는지 확인하기

3-1 회원 테이블에 데이터가 4건 들어 있는지 확인

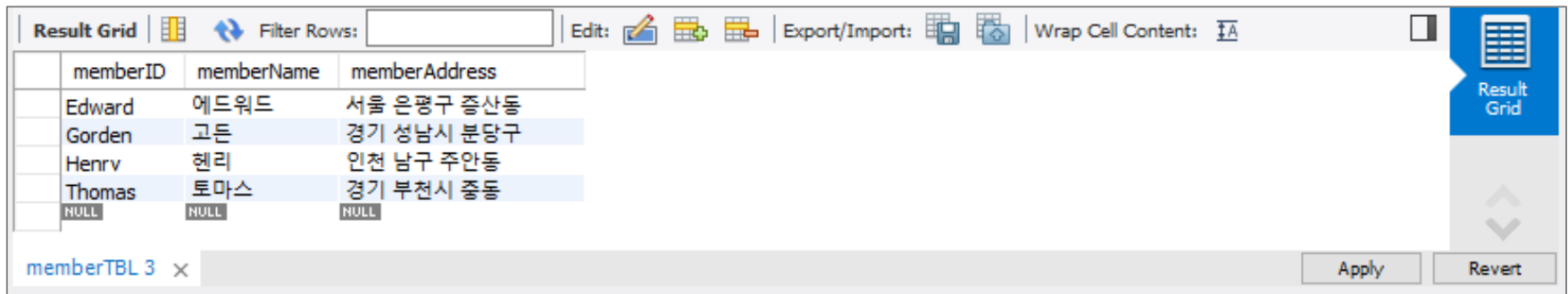
```
SELECT * FROM memberTBL;
```

3-2 회원 테이블에 '홍민'을 삽입한 후 바로 삭제

```
INSERT INTO memberTBL VALUES ('Soccer', '홍민', '서울시 서대문구 북가좌동');  
DELETE FROM memberTBL WHERE memberName = '홍민';
```

3-3 홍민이 회원 테이블에서 삭제되었는지 확인

```
SELECT * FROM memberTBL;
```



memberID	memberName	memberAddress
Edward	에드워드	서울 은평구 종산동
Gorden	고든	경기 성남시 분당구
Henry	헨리	인천 남구 주안동
Thomas	토마스	경기 부천시 중동
NULL	NULL	NULL

3-4 백업 테이블 확인

```
SELECT * FROM deletedMemberTBL;
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
memberID	memberName	memberAddress	deletedDate
Soccer	홍민	서울시 서대문구 북가좌동	2018-09-03

deletedMemberTBL 5 × Read Only

3-5 Workbench 종료



Thank You
