

# MySQL 설치

# MySQL의 특징

## 1. SQL이란 무엇인가?

- 1) 비탐색적(NON NAVIGATIONAL)입니다. 이것은 SQL을 통하여 원하는 데이터를 DB에 알려주기만 하면 되므로, DB가 해당 데이터를 얻는 방법을 지정할 필요가 없습니다. 예를 들어, DB는 데이터를 검색하기 위하여 인덱스(INDEX)를 사용할지 여부를 결정합니다.
- 2) SQL은 비절차적(NON-PROCEDURAL) 언어입니다. 절차적 언어는 프로그래밍 언어입니다. SQL은 프로그래밍 언어가 아니므로 반복 수행하거나 확장 IF-THEN-ELSE 단계를 통하여 한 번에 하나의 레코드를 검색하지 않습니다. SQL은 테이블의 일련의 레코드(RECORD)들을 처리하며 해당 데이터에 대한 절차적 논리를 수행하는 프로그래밍 언어 안에 포함되어 해당 언어의 기능을 활용할 수 있습니다

## 2. MySQL의 특징

- 1) SQL에 기반을 둔 관계형 DBMS 중 하나
- 2) Oracle, IBM, Infomix 등의 데이터베이스는 고가이지만 MySQL 데이터베이스는 무료(배포판)
- 3) 리눅스, 유닉스, 윈도우 등 거의 모든 운영체제에서 사용가능
- 4) 처리 속도가 상당히 빠르고 대용량에 데이터 처리 용이
- 5) 설치 방법이 쉽고 초보자도 익히기 쉬움
- 6) 보안성이 우수

# DBMS 개요

## ❖ 데이터베이스의 정의와 특징

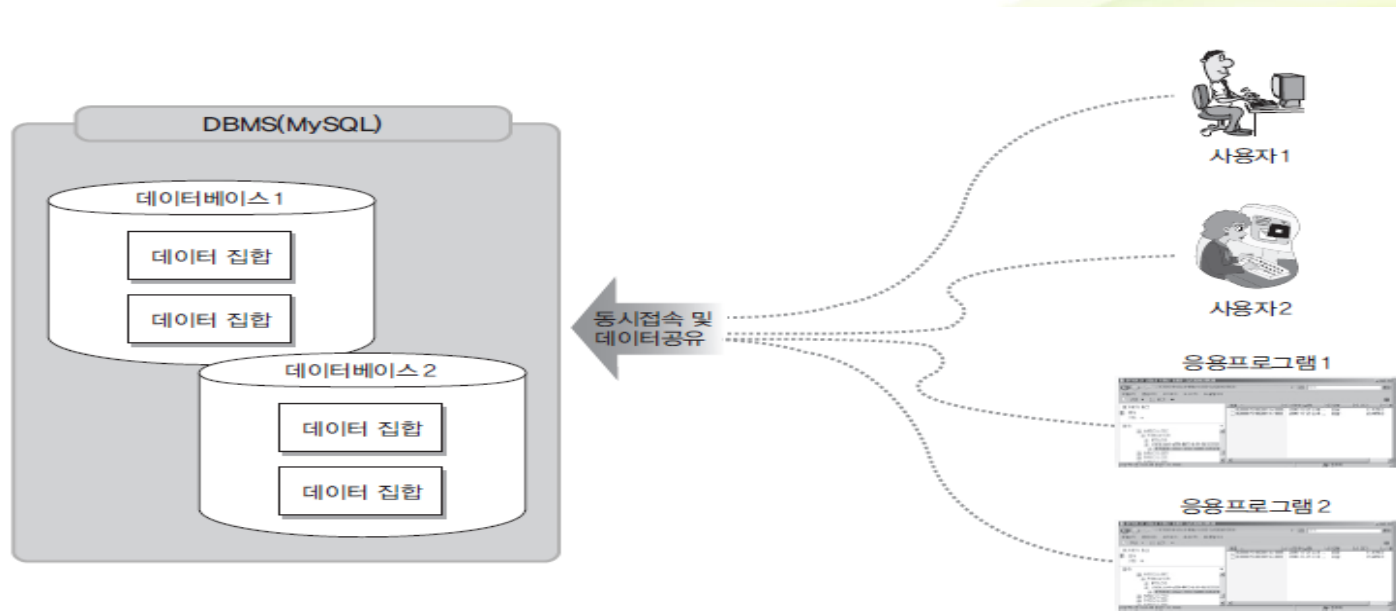
### ■ 데이터베이스

- ‘데이터의 집합’
- 여러 명의 사용자나 응용프로그램이 공유하는 데이터들
- 동시에 접근 가능해야
- ‘데이터의 저장 공간’ 자체

### ■ DBMS

- 데이터베이스를 관리·운영하는 역할

## ❖ DBMS 개념도



# DBMS 개요

## ❖ DB/DBMS의 특징

### ■ 데이터의 무결성 (Integrity)

- 데이터베이스 안의 데이터는 오류가 없어야.
- 제약 조건(Constrain)이라는 특성을 가짐

### ■ 데이터의 독립성

- 데이터베이스 크기 변경하거나 데이터 파일의 저장소 변경
  - 기존에 작성된 응용프로그램은 전혀 영향을 받지 않아야

### ■ 보안

- 데이터베이스 안의 데이터에 데이터를 소유한 사람이나 데이터에 접근이 허가된 사람만 접근할 수 있어야 접근할 때도 사용자의 계정에 따라서 다른 권한 가짐

## ❖ DB/DBMS의 특징

### ■ 데이터 중복의 최소화

- 동일한 데이터가 여러 개 중복되어 저장되는 것 방지

### ■ 응용프로그램 제작 및 수정이 쉬워짐

- 통일된 방식으로 응용프로그램 작성 가능
- 유지보수 또한 쉬워 짐

### ■ 데이터의 안전성 향상

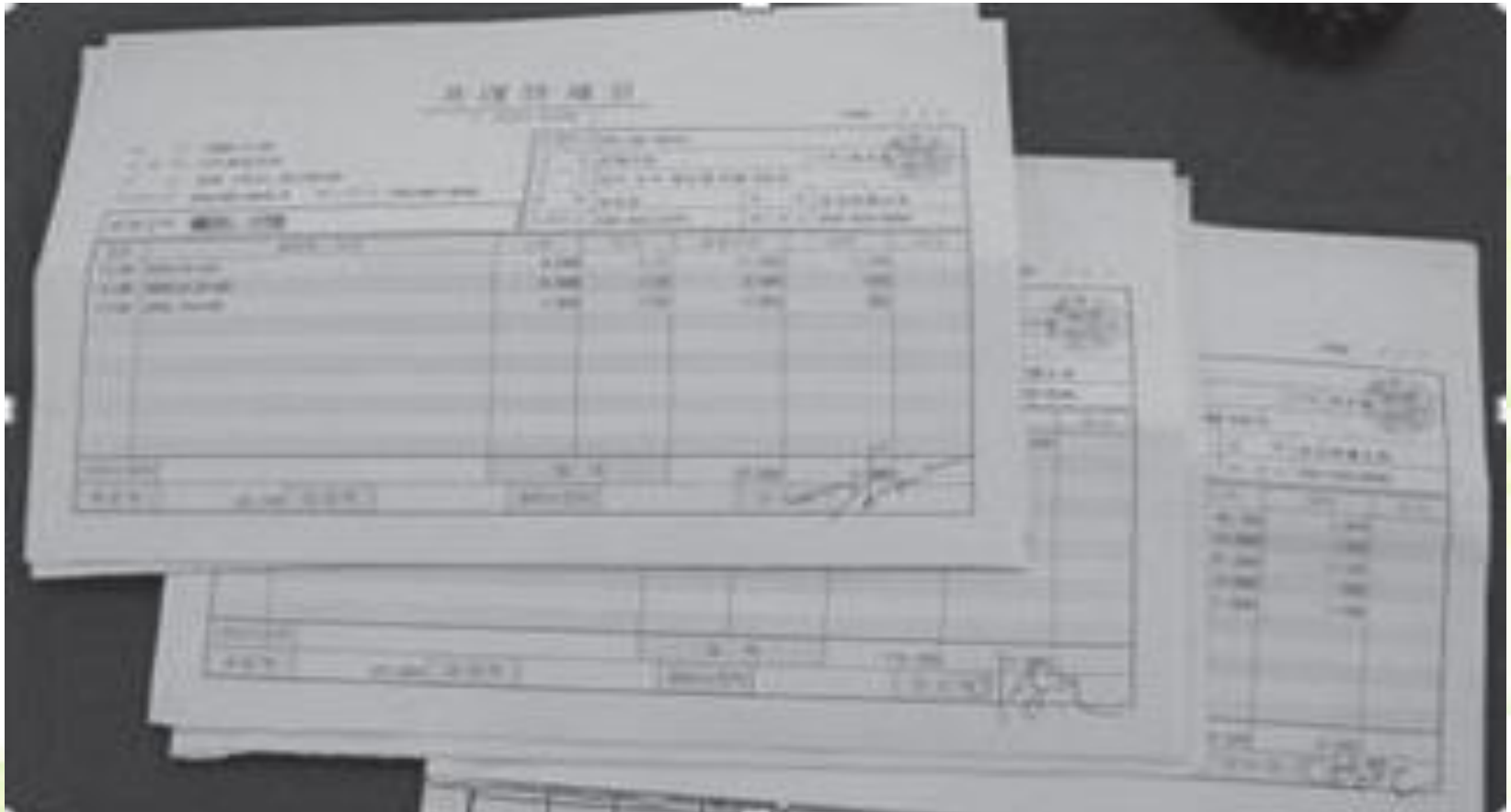
- 대부분의 DBMS가 제공하는 백업·복원 기능 이용
- 데이터가 깨지는 문제가 발생할 경우 원상으로 복원 , 복구하는 방법이 명확해짐

# DBMS 개요

## ❖ 데이터베이스의 발전

### ■ 오프라인 관리

- 종이에 연필로 기록해 장부로 관리

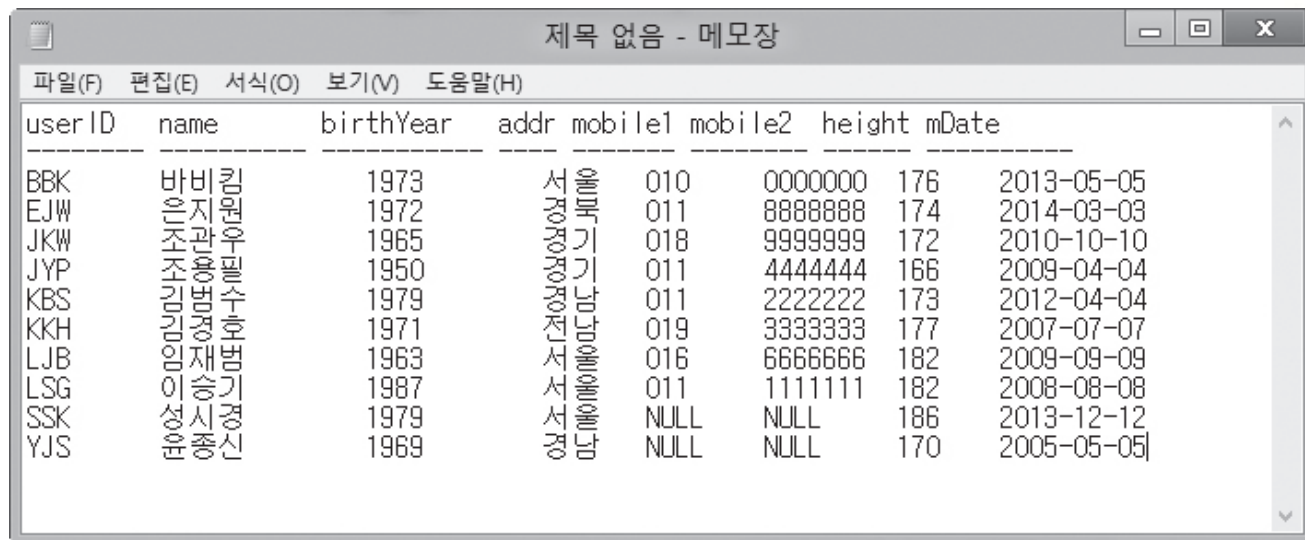


# DBMS 개요

## ❖ 데이터베이스의 발전

### ■ 파일시스템 사용

- 컴퓨터 파일에 기록/저장 - 메모장, 엑셀 활용
- 컴퓨터에 저장된 파일의 내용은 읽고, 쓰기가 편한 약속된 형태의 구조 사용
- 데이터의 양이 많아지면 데이터 중복으로 인한 불일치 위험



The screenshot shows a Windows Notepad window with the title '제목 없음 - 메모장'. The menu bar includes '파일(F)', '편집(E)', '서식(O)', '보기(V)', and '도움말(H)'. The text content is a table with 8 columns: userID, name, birthYear, addr, mobile1, mobile2, height, and mDate. The data is as follows:

userID	name	birthYear	addr	mobile1	mobile2	height	mDate
BBK	바비킴	1973	서울	010	0000000	176	2013-05-05
EJW	은지원	1972	경북	011	8888888	174	2014-03-03
JKW	조관우	1965	경기	018	9999999	172	2010-10-10
JYP	조용필	1950	경기	011	4444444	166	2009-04-04
KBS	김범수	1979	경남	011	2222222	173	2012-04-04
KKH	김경호	1971	전남	019	3333333	177	2007-07-07
LJB	임재범	1963	서울	016	6666666	182	2009-09-09
LSG	이승기	1987	서울	011	1111111	182	2008-08-08
SSK	성시경	1979	서울	NULL	NULL	186	2013-12-12
YJS	윤종신	1969	경남	NULL	NULL	170	2005-05-05

# DBMS 개요

## ❖ 데이터베이스의 발전

### ■ 데이터베이스 관리시스템

- 파일시스템의 단점 보완
- 대량의 데이터를 보다 효율적으로 관리하고 운영하기 위해 사용
- DBMS - DataBase Management System
- 데이터의 집합인 '데이터베이스' 를 잘 관리하고 운영하기 위한 시스템 또는 소프트웨어

### ■ SQL( Structured Query Language)

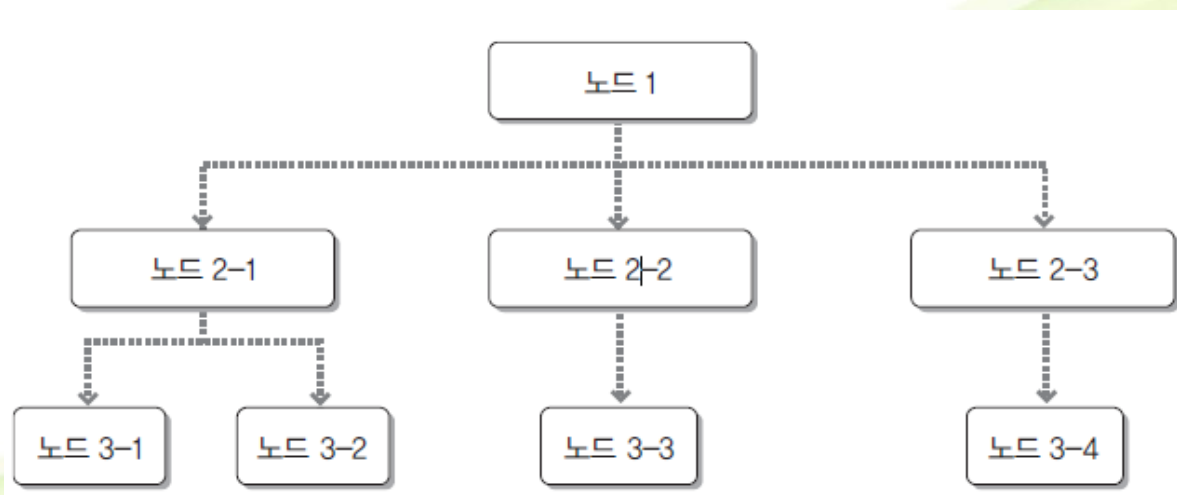
- DBMS에 데이터 구축/관리/활용 위해서 사용되는 언어
- DBMS를 통해 중요한 정보들을 입력, 관리, 추출

# DBMS 개요

## ❖ DBMS 분류

### ■ 계층형 DBMS

- 처음으로 나온 DBMS 개념 - 1960년대에 시작
- 각 계층은 트리Tree 형태, 1:N 관계
- 문제점
  - 처음 구축한 이후 그 구조를 변경하기가 상당히 까다로움
  - 주어진 상태에서의 검색은 상당히 빠름
  - 접근 유연성 부족해서 임의의 검색에는 어려움



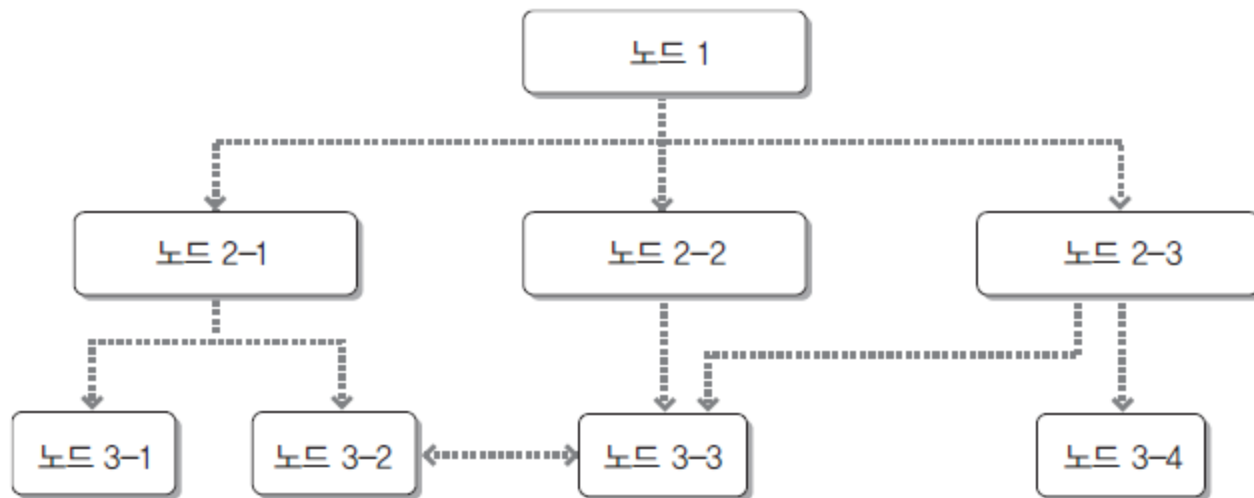


# DBMS 개요

## ❖ DBMS 분류

### ■ 망형 DBMS

- 계층형 DBMS의 문제점을 개선하기 위해 1970년대에 시작
- 1:1, 1:N, N:M(다대다) 관계 지원 - 효과적이고 빠른 데이터 추출
- 복잡한 내부 포인터 사용
  - 프로그래머가 이 모든 구조를 이해해야만 프로그램의 작성 가능

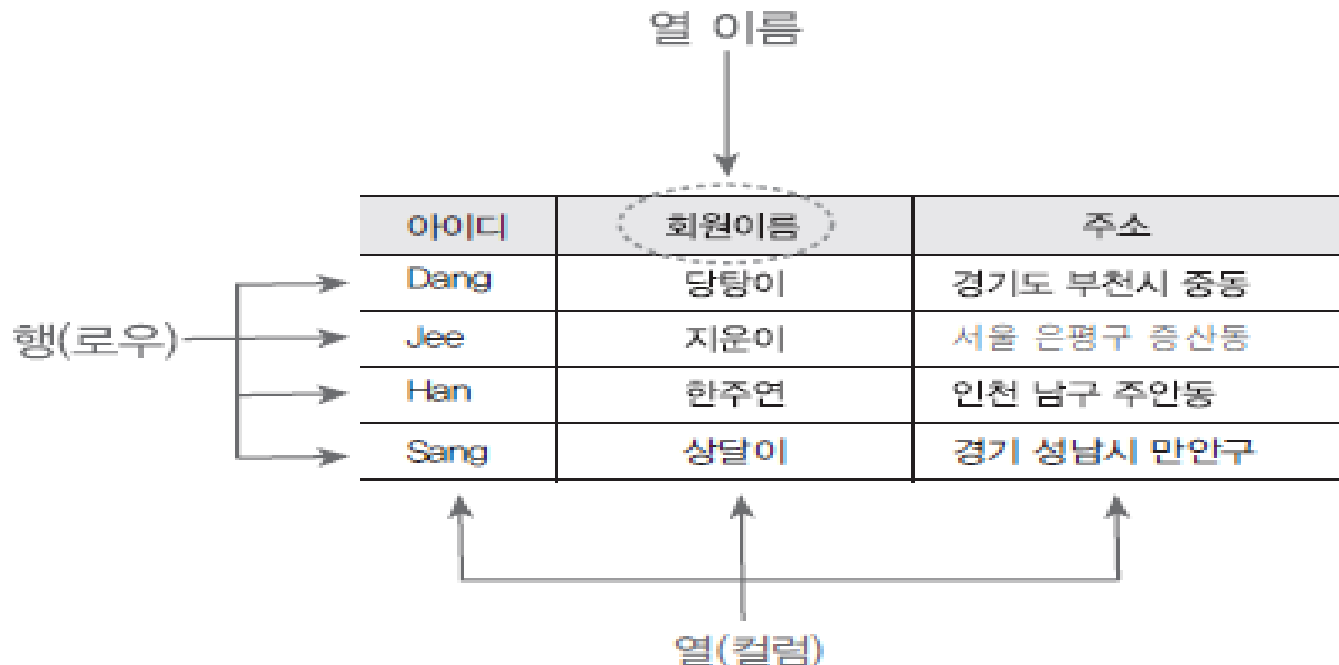


# DBMS 개요

## ❖ DBMS 분류

### ■ 관계형 DBMS (Relational DBMS)

- 1969년 E.F.Codd라는 학자가 수학 모델에 근거해 고안
- 데이터베이스는 테이블Table이라 불리는 최소 단위로 구성
- 이 테이블은 하나 이상의 열로 구성



# DBMS 개요

## ❖ 관계형 DBMS (Relational DBMS)의 장단점

### ■ 장점

- 다른 DBMS에 비해 업무가 변화될 경우 쉽게 변화에 순응
- 유지보수 측면에서도 편리
- 대용량 데이터의 관리와 데이터 무결성Integration보장

### ■ 단점

- 시스템 자원을 많이 차지해 시스템이 전반적으로 느려지는 것
  - 하드웨어 발전되어 해결

## ❖ SQL 개요

### ■ SQL (Structured Query Language)

- 관계형 데이터베이스에서 사용되는 언어, ‘에스큐엘’ 또는 ‘시퀄’

### ■ DBMS 제작 회사와 독립적

### ■ 다른 시스템으로 이식성이 좋음

### ■ 표준이 계속 발전중

### ■ 대화식 언어

### ■ 분산형 클라이언트/서버 구조

# MySQL의 에디션 및 기능 비교

## ❖ 상용 에디션

- Standard, Enterprise, Cluster CGE
- 비용이나 기능 면 비교
  - Standard < Enterprise < Cluster CGE

## ❖ 무료 에디션

- Community
- Enterprise 버전과 기능상 차이는 거의 없음
- 사용 허가에 대한 라이선스 차이

# MySQL 설치

# MySQL 설치 전 준비사항

## ❖ 소프트웨어 요구사항

### ■ MySQL Community 설치 위한 하드웨어

- Windows가 설치된 머신

### ■ 현재 최신버전은 8.버전

### ■ MySQL Community 설치 위한 소프트웨어

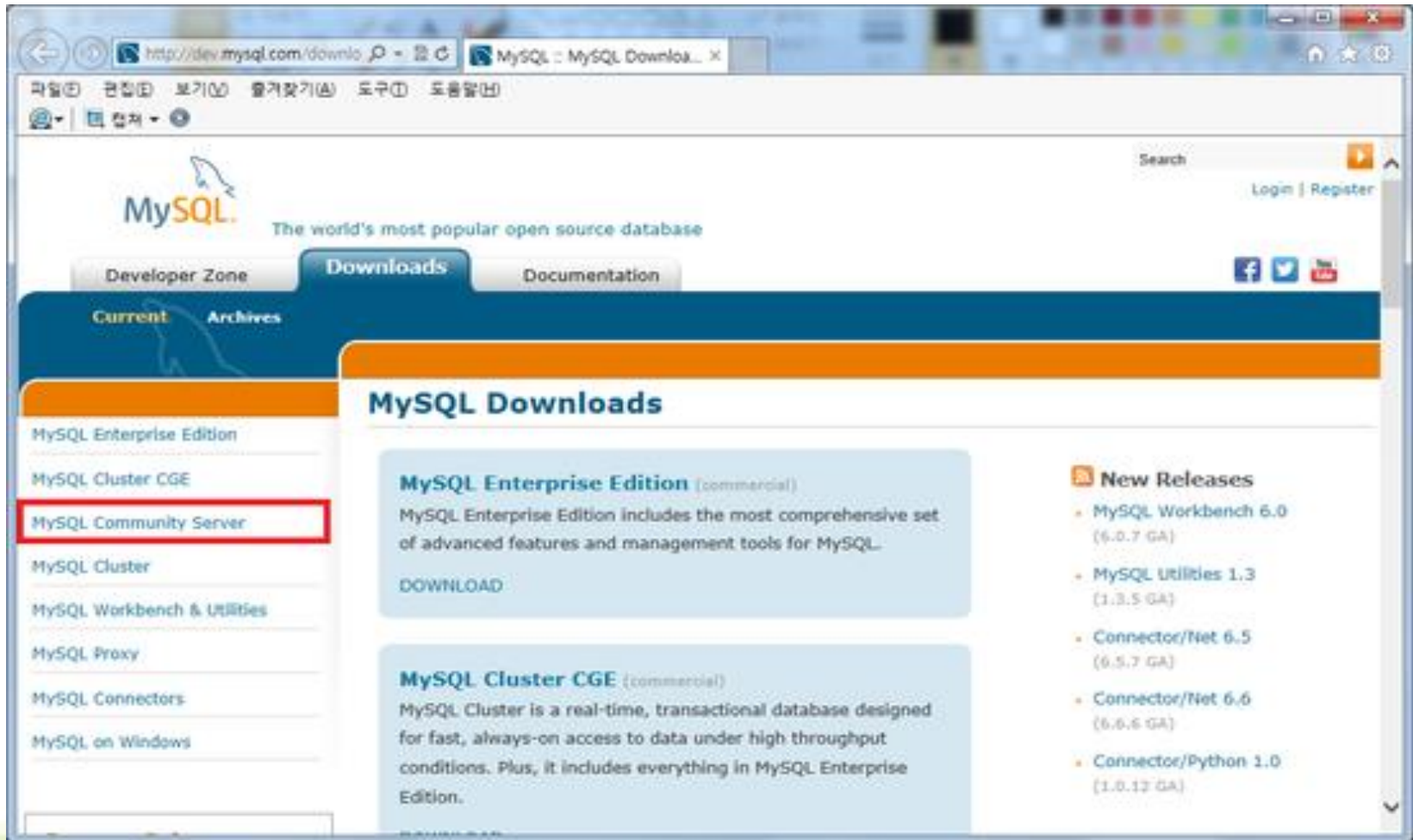
- Windows 7 이상, Windows Server 2008 R2 이상의 버전에서만 설치

서버 운영체제(x64)	PC 운영체제(x64, x86)
Windows Server 2016	Windows 10
Windows Server 2012 R2	Windows 8.1
Windows Server 2012	Windows 8
Windows Server 2008 R2(SP1)	Windows 7(SP1)

- 부가적인 기능을 사용하기 위해서 추가 소프트웨어 설치

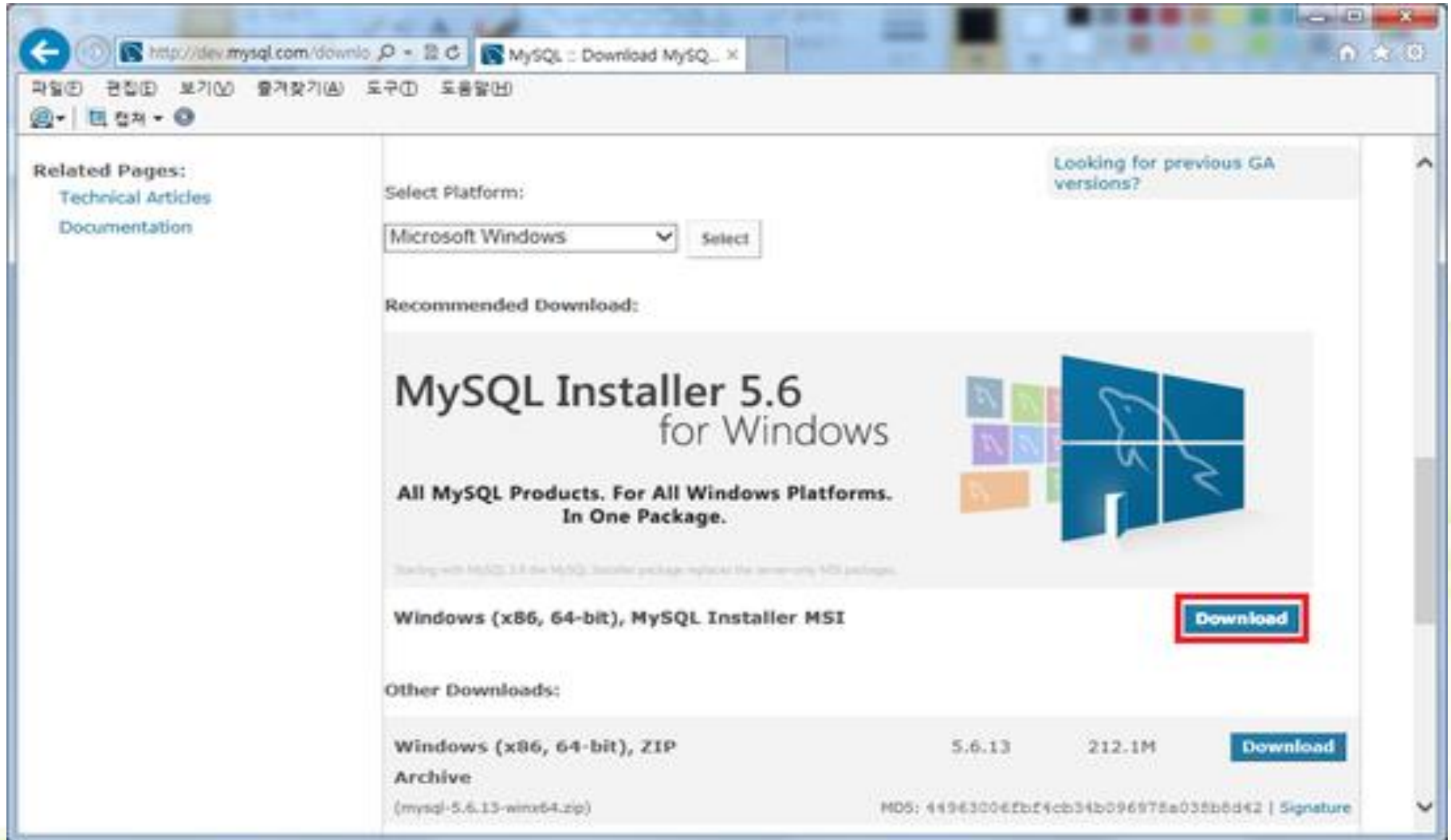
# MySQL 설치

- ❖ <http://dev.mysql.com/downloads/> 사이트에 가서 왼쪽의 MySQL Community Server 을 선택합니다.



# MySQL 설치

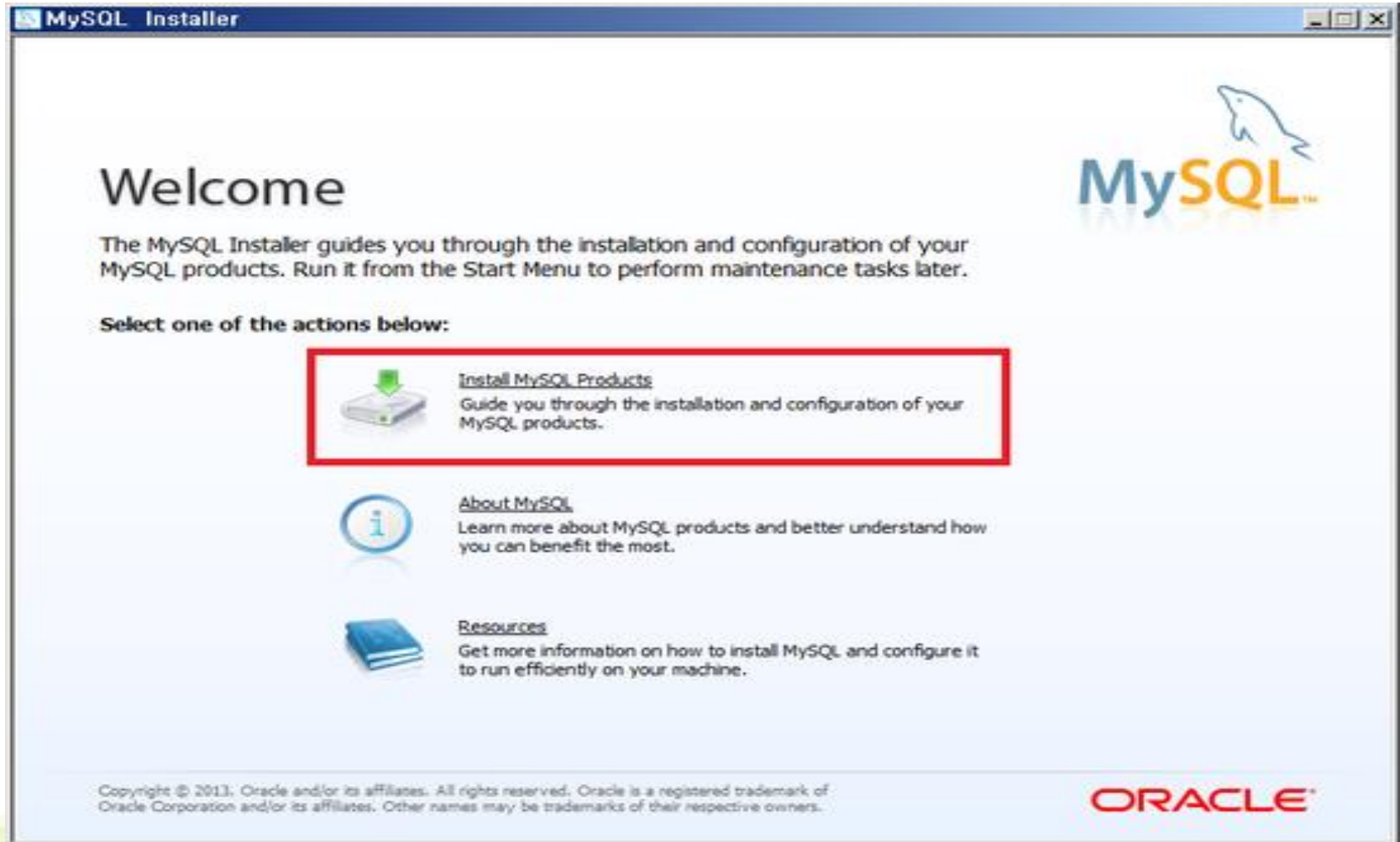
- ❖ MSI 버전은 닷넷 프레임워크 4.50이 설치되어 있어야 합니다
- ❖ Mysql 8.\* 버전은 4.5.20이상





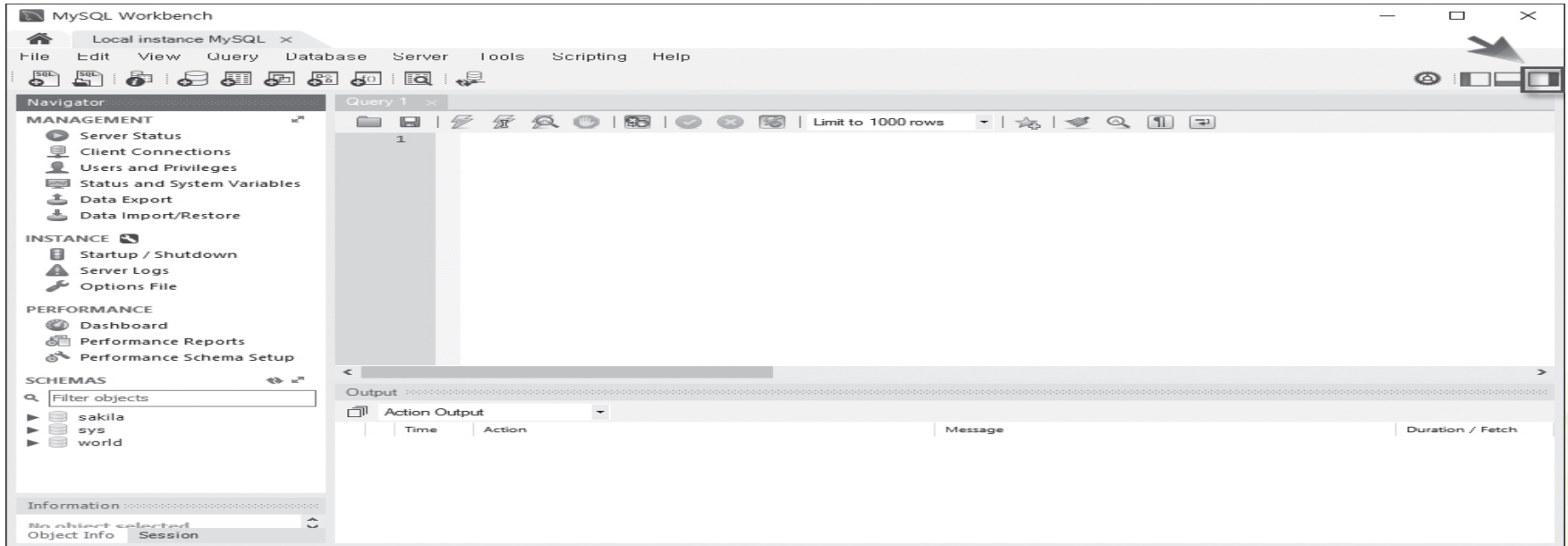
# MySQL 설치

- ❖ 다운받은 설치파일을 실행해서 Install MySQL Products 를 선택합니다.



# MySQL 설치

## ❖ Work Bench



MySQL의 실행 파일 경로 Path에 추가

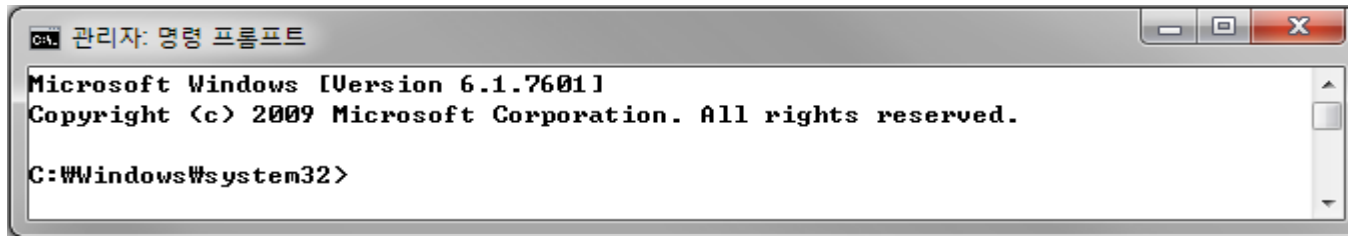
내 컴퓨터 - 속성 - 고급시스템설정 - 고급 - 환경변수

Path를 선택하고 커서를 맨 뒤로 이동

세미콜론(;)을 추가하고 ;C:\Program Files\MySQL\MySQL Server 버전\bin" Path 추가

# MySQL의 시작

## 명령 프롬프트의 실행



```
관리자: 명령 프롬프트
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

### MySQL 접속 명령 1

```
C:\W> mysql -u 계정 -p비밀번호
mysql> use 데이터베이스명
```

### MySQL 접속 명령 2

```
C:\W> mysql -u 계정 -h hostname -p비밀번호 데이터베이스명
```

**계정 : kdhong, 비밀번호:1234, DB명:kdhong\_db**

```
C:\W> mysql -ukdhong -p1234 kdhong_db
```

### MySQL 접속 종료



```
관리자: 명령 프롬프트
mysql> quit
Bye
C:\W>
```

# 유틸리티 사용법

# MySQL Workbench 사용 방법

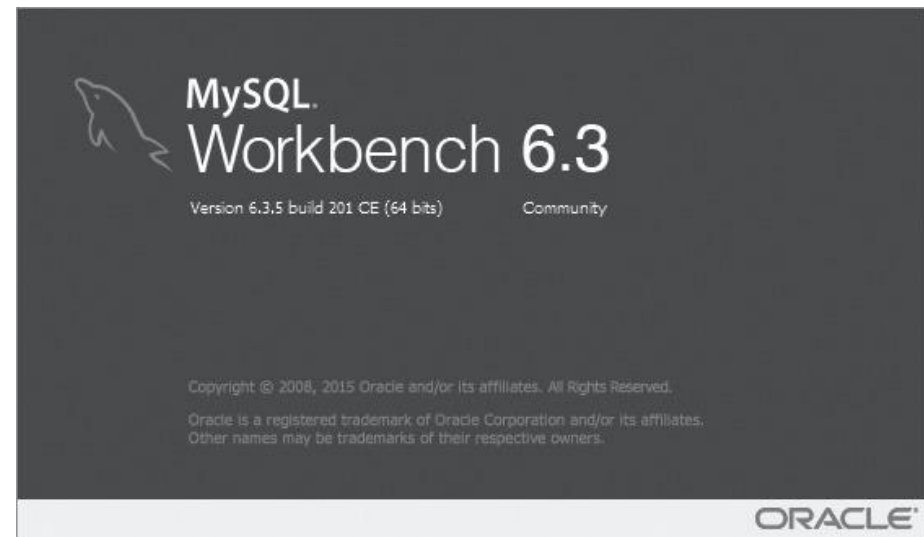
## ❖ MySQL Workbench의 발전과정

- 2002년에 만들어진 DBDesigner4 제품
  - MySQL의 비주얼 툴로 사용
- 2003년에 MySQL GUI Tools Bundle로 통합
  - 2005년에 MySQL Workbench 프리뷰버전으로 변경되어 발표
  - 2007년부터 본격적으로 개발되고 버전이 업그레이드
- MySQL 5.0 버전부터 본격적으로 MySQL의 GUI 툴로 제공
  - Workbench 5.0 버전은 Windows용으로만 제공
  - 5.1 버전에서 다른 운영체제도 지원
  - 2015년에 6.3 버전 발표

# MySQL Workbench 사용 방법

## ❖ Workbench의 주요한 기능

- 데이터베이스 연결 기능
- 인스턴스 관리
- 위저드를 이용한 MySQL의 동작
- 통합된 기능의 SQL 편집기
- 데이터베이스 모델링 기능
- 포워드/리버스 엔지니어링 기능
- 데이터베이스 인스턴스 시작/종료
- 데이터베이스 내보내기/가져오기
- 데이터베이스 계정 관리



## ❖ MySQL Workbench의 버전과 실행

- Windows [시작] >> [모든 앱] >> [MySQL] >> [MySQL Workbench 6.3 CE]
  - MySQL Community 5.7.10에는 MySQL Workbench 6.3 버전 포함

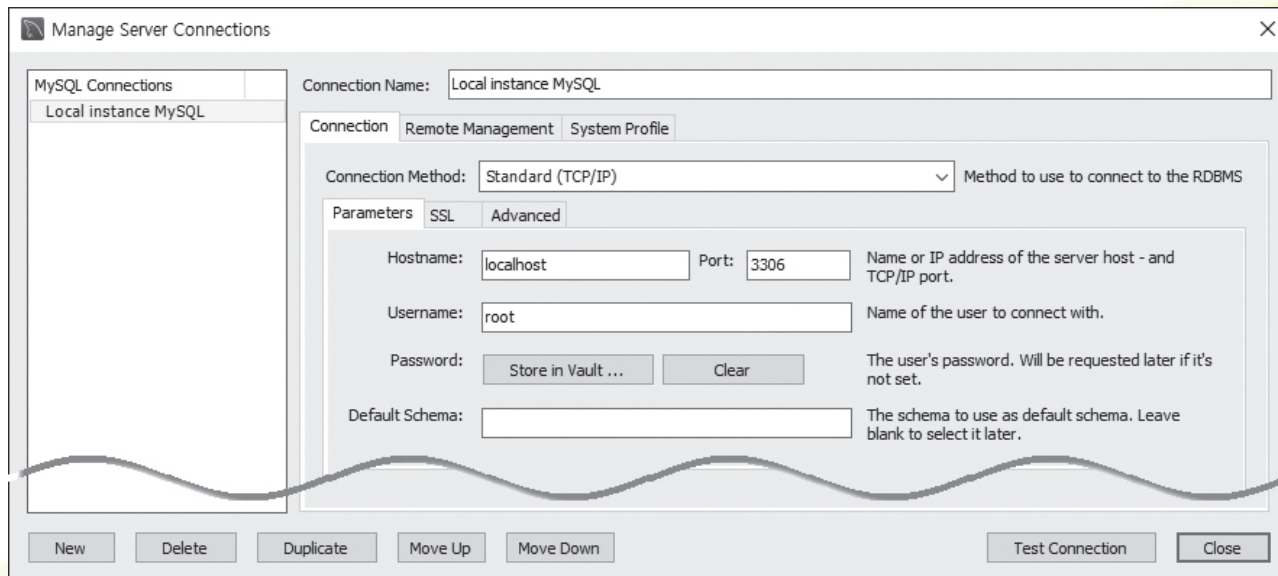
# MySQL Workbench 사용 방법

## ❖ [MySQL Connections] 창

### ■ Workbench 실행

- [MySQL Connections] 창

- 접속될 서버와 사용자, 포트를 선택한 후 접속 시도
- MySQL에 등록된 사용자만 접속
- 접속하는 서버 등록 시 여러 개 등록 가능
- **Connection Name:** 접속하는 이름





# MySQL Workbench 사용 방법

## ❖ [Connection] 탭

### ■ Connection Method

- Standard(TCP/IP), Local Socket/Pipe, Standard TCP/IP over SSH, MySQL Fabric Management Node 등 4가지 중에 선택 가능
- 대부분 Standard (TCP/IP) 사용

### ■ [Parameters] 탭

- Hostname
- Localhost = 127.0.0.1 = 자신의 컴퓨터 = MySQL이 설치된 컴퓨터
- 접속할 컴퓨터가 외부에 있다면 접속할 서버 컴퓨터의 IP주소 입력
- Port
  - 접속할 MySQL 포트 번호
  - 특별한 경우가 아니면 330
  - 0

## ❖ [Connection] 탭

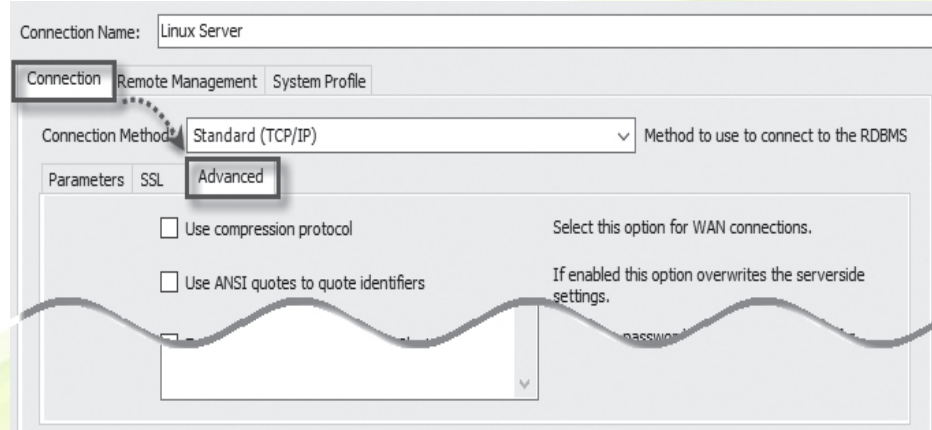
### ■ [SSL] 탭

- SSL (Secure Socket Layer)
  - 보안을 위한 암호 규약

- 서버와 클라이언트가 통신할 때 암호화 통해 비밀 유지 + 보안 강화

### ■ [Advanced] 탭

- 프로토콜의 압축, 인증 방식 등을 설정





# MySQL Workbench 사용 방법

## ❖ [Connection] 탭

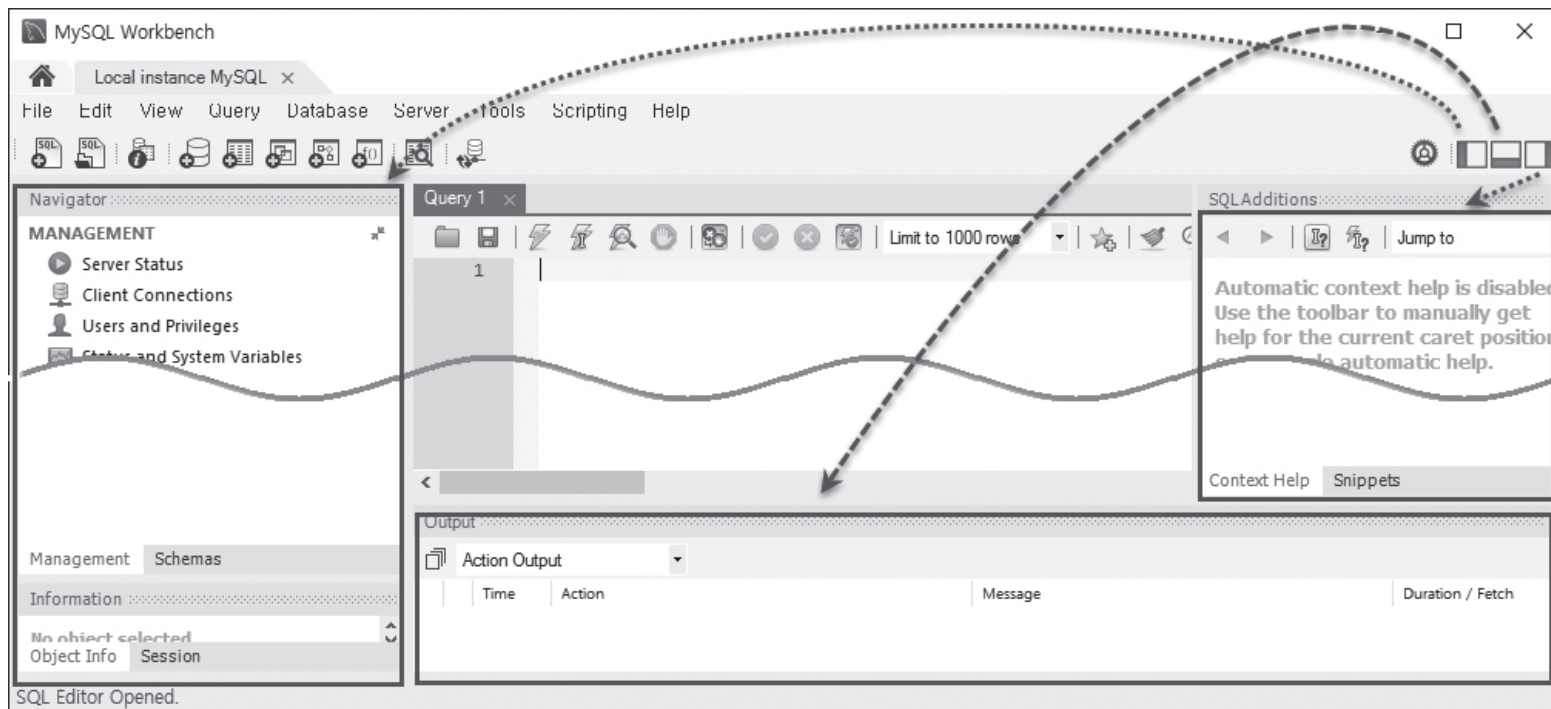
### ■ [System Profile] 탭

- 접속할 서버의 OS 종류 및 MySQL 설정 파일의 경로 등을 설정
- [Remote Management]에서 'Native Windows remote management'나 'SSH login based management'가 선택되어 있어야 활성화
- [System Type]은 FreeBSD, Linux, MacOS X, OpenSolaris, Windows 등 5가지 중 선택
  - [Installation Type]을 선택 가능
- [Configuration File]
  - **MySQL의 설정 파일이 경로와 함께 지정**
- [Configuration File Section]
  - **서버의 서비스 이름 지정**
- [MySQL Management]
  - **MySQL 서비스를 시작하거나 중지하는 시스템 명령어**

# MySQL Workbench 사용 방법

## ❖ MySQL Workbench의 화면 구성

- 3개의 패널과 쿼리 창으로 구성
- 내비게이터, Output, SQL Additions



# MySQL Workbench 사용 방법

## ❖ MySQL Workbench의 화면 구성

### ■ 내비게이터 (Navigator)

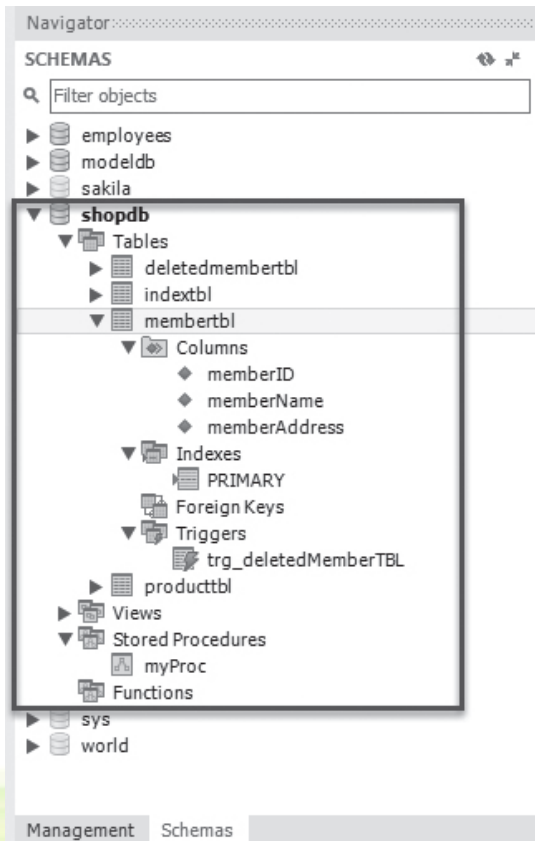
- MySQL의 관리 및 운영을 위한 강력한 도구
- MySQL 명령문이나 SQL문을 모르더라도 대부분의 작업 수행 가능
- 내비게이터의 역할
  - **[Schemas] 탭**
    - » 데이터베이스(=스키마) 생성 및 삭제
    - » 데이터베이스 개체(테이블, 뷰, 인덱스, 저장 프로시저, 함수 등)를 생성하고 관리
    - » 데이터베이스의 속성 조회
  - **[Management] 탭**
    - » MANAGEMENT
    - » INSTANCE
    - » PERFORMANCE

# MySQL Workbench 사용 방법

## ❖ MySQL Workbench의 화면 구성

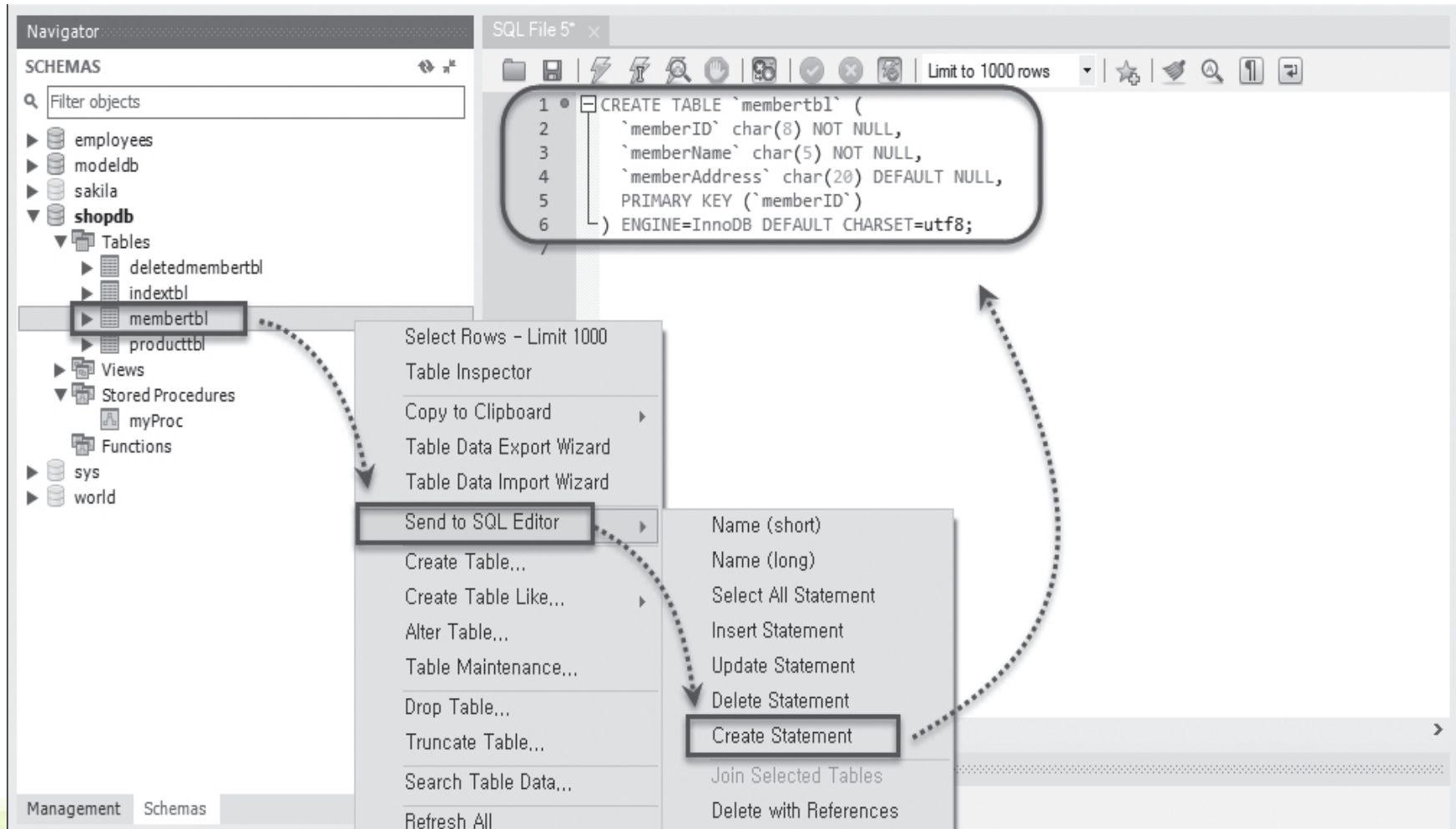
### ■ 내비게이터 (Navigator)

- [Navigator]의 [Schemas]는 트리 형태
- 각각의 항목은 '▶' 기호 클릭해 확장 가능



# MySQL Workbench 사용 방법

- ❖ 내비게이터의 [Schemas] 탭 이용해 SQL문 자동 생성
  - 테이블 생성 이외에도 뷰와 다른 구문을 다룰 수 있음



# MySQL Workbench 사용 방법

## ❖ 내비게이터의 [Management] 탭 이용해 MySQL 관리

### ■ [MANAGEMENT] 부분

- [Server Status]
  - 현재 접속된 서버의 상태 파악 가능
  - 현재 서버의 가동 상태, 포트, 환경 파일의 경로, 메모리 상태나 CPU 사용 상태
- [Client Connections]
  - 연결된 클라이언트의 현재 상태가 휴면(Sleep) 인지 여부 확인
  - 필요하다면 해당 연결에서 마우스 오른쪽 버튼을 클릭한 후 [Kill Connection(s)]으로 연결을 강제로 끊을 수 있음
- [Users and Privileges]에서 MySQL 사용자 관리
- [Status and System Variables]
  - MySQL 서버에 설정된 시스템 변수들 확인 / 변경
- [Data Export] 및 [Data Import/Restore]
  - 백업 및 복원과 관련된 부분 (3장에서 이미 다룬 내용)

# MySQL Workbench 사용 방법

## ❖ 내비게이터의 [Management] 탭 이용해 MySQL 관리

### ■ [INSTANCE] 부분

- [Startup/Shutdown]
  - MySQL 서버의 현재 작동 상태 확인
  - MySQL 서버의 종지와 시작 설정
- [Server Logs] 서버에 기록된 오류, 경고, 방화벽 등의 로그 확인
- [Options File]
  - MySQL의 핵심 설정 파일인 my.ini 파일
  - 파일 설정 내용을 GUI 모드로 보여줌

## ❖ 내비게이터의 [Management] 탭 이용해 MySQL 관리

### ■ [PERFORMANCE] 부분

- [Dashboard]
  - 네트워크, MySQL 서버, InnoDB의 상태를 그래픽으로 보여줌
- [Performance Reports]
  - 입출력이 오래 걸린 파일, 비용이 많이 든 쿼리문, 데이터베이스 통계 등의 항목들 조회
  - 결과 내보내기
- [Performance Schema Setup]
  - 성능에 대한 설정
  - 오른쪽 위 <Show Advanced>나 <Hide Advanced> 클릭
    - » 세부적인 설정 가능



# MySQL Workbench 사용 방법

## ❖ 쿼리 창 (Query Editor)

- ‘쿼리 문장(SQL 구문)을 입력/실행하는 텍스트 에디터

### ■ 쿼리 창 사용 방법

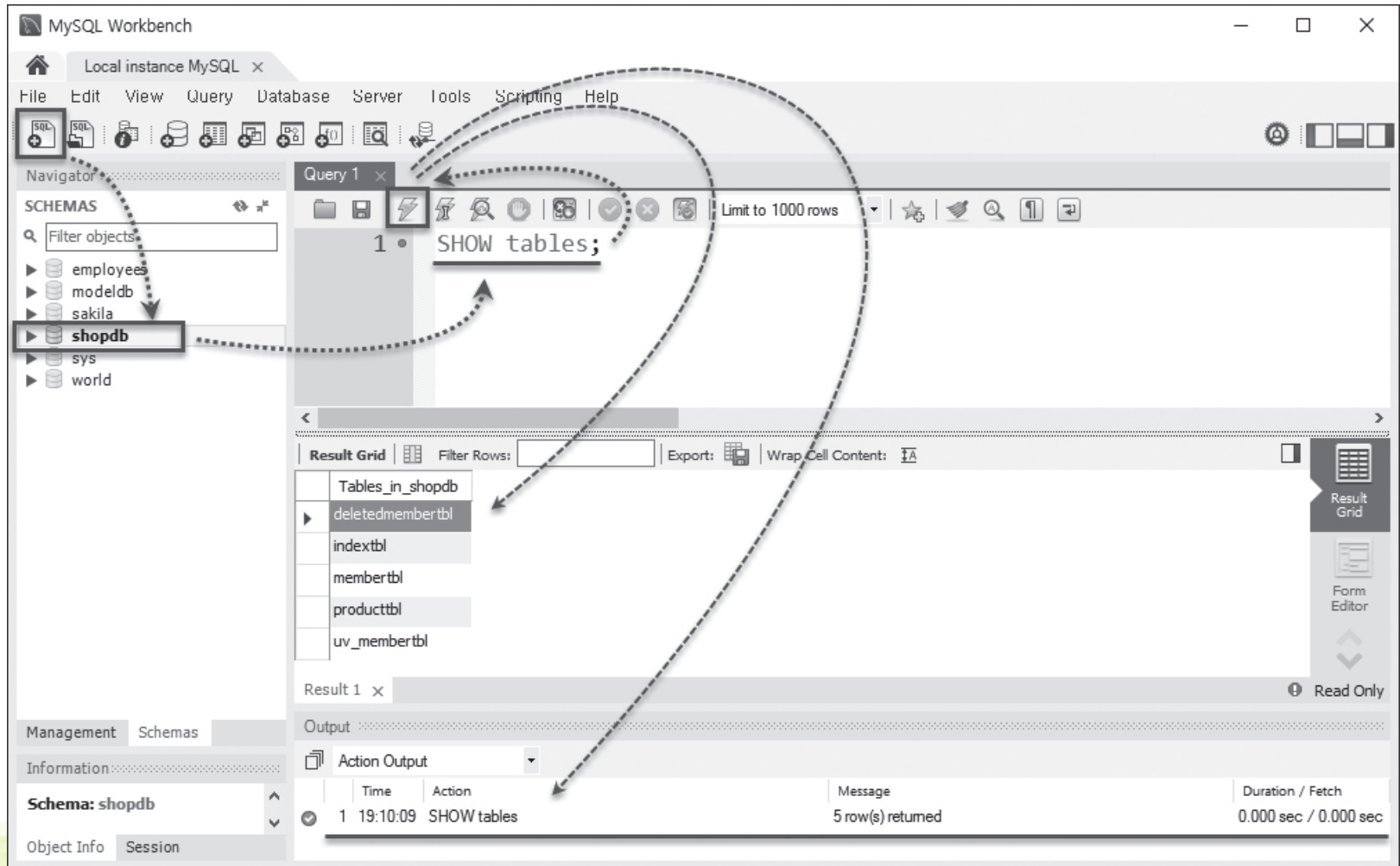
- Workbench의 상단 제일 왼쪽의 <Create a new SQL tab for executing queries> 아이콘 클릭 or Workbench 메뉴의 [File] >> [New Query Tab]을 클릭해 쿼리 창 열기
- 작업할 데이터베이스를 [Schemas] 탭에서 더블 클릭해 선택
- SQL문 문법에 맞게 입력
- SQL 구문에 이상이 없다면 툴바의 <Execute the selected portion~~> 아이콘을 클릭하거나 Ctrl + Shift + Insert 눌러서 SQL 문장 실행
- 아래쪽의 결과 창을 통해 결과 확인
  - 성공된 결과 또는 오류 메시지 확인



# MySQL Workbench 사용 방법

## ❖ 쿼리 창 (Query Editor)

- 한 번 열린 쿼리 창은 계속해서 SQL 입력해 사용 가능



# MySQL Workbench 사용 방법

## ❖ 쿼리 창 (Query Editor)의 다른 기능들

- 개체 드래그 해 자동 완성 기능
- 예약어 대문자나 소문자로 변경하기
- SQL 코드나 설명의 주석처리 방법
- 여러 개의 SQL 실행하기
  - 모든 SQL 다 실행할지, 일부만 드래그 선택해 실행할지 결정 중요함
- 결과를 다양한 방식으로 필터링 + 파일 형태 저장 가능
- 실행되는 SQL 문 실행 계획 확인해보기

# 사용자 관리하기

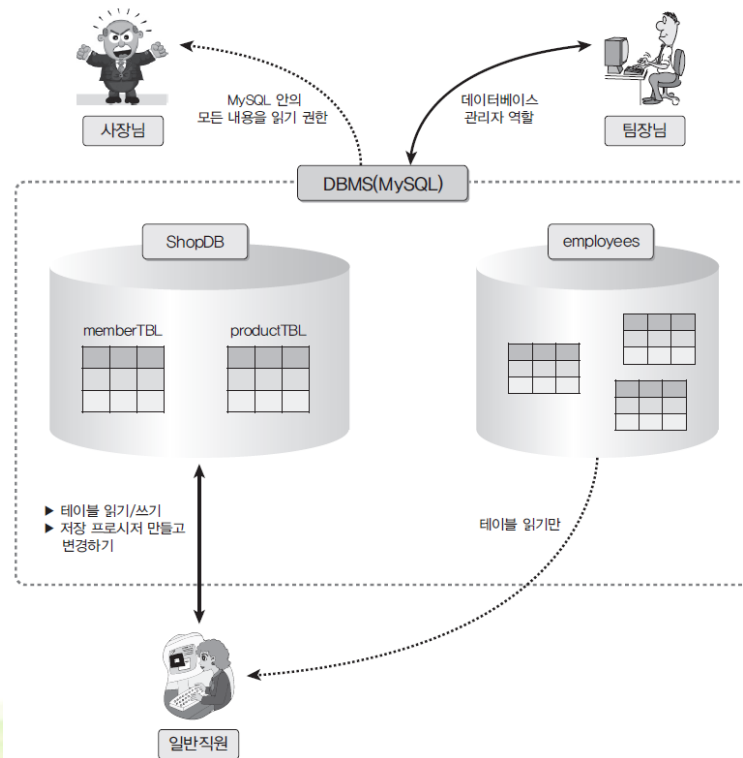
## ❖ DB 사용자 관리의 필요성

### ■ 현재까지 사용 방법

- MySQL 관리자인 root로 접속해 사용

### ■ 실무에서의 문제

- MySQL 데이터베이스를 다양한 사용자나 응용프로그램에 접속해 사용



# 사용자 관리하기

## ❖ DB 사용자 관리 개념

### ■ 권한 (Privileges)

- 단편적인 개념
- Ex) SELECT 권한, INSERT 권한, CREATE 권한

### ■ 역할 (Role)

- 권한의 집합 개념
- Ex) DBA 역할은 SELECT 권한 등 모든 권한 포함

# 사용자 관리하기

## ❖ MySQL의 사용자 및 역할/권한 관리

### ■ Director

- 데이터베이스 관리자(DBA)의 역할 부여
- Workbench 실행하고 [Local instance MySQL]을 클릭해서 접속
  - **사용자를 생성하는 권한은 root에게만 있음**
- [Navigator]의 [Management] 탭 → [Users and Privileges] 클릭
- [Users and Privileges] 창에서 왼쪽 아래 <Add Account> 클릭한 후 [Login] 탭의 [Login Name]에 'director' 입력
  - **비밀번호 입력하고 <Apply> 클릭 → director 사용자 등록 확인**
- [Account Limits] 탭으로 쿼리 한계 설정 → 0은 제한 없음
- [Administrative Roles] - MySQL 자체에 대한 권한 설정

# 사용자 관리하기

## ❖ MySQL의 사용자 및 역할/권한 관리

- MySQL의 모든 데이터에 읽기 (Select) 권한 부여
- 계정 등록방법은 앞의 경우와 같음
- [Administrative Roles] 탭 클릭
  - MySQL의 모든 데이터를 읽을 수 있게 계획되어 있음
  - [Global Privileges] 중에서 <SELECT> 체크
  - 왼쪽 Role 중에 <Custom>이 자동으로 체크

# 사용자 관리하기

## ❖ MySQL의 사용자 및 역할/권한 관리

### ■ 사용자

- ShopDB 데이터베이스의 모든 테이블에 대해 읽기(Select) , 쓰기 (Insert) , Update, Delete 권한 부여
- 스토어드 프로시저 등을 생성(Create Routine) 하고 수정( Alter Routine) 할 수 있는 권한 부여
- employees 데이터베이스의 테이블에 대해서는 읽기 권한만 부여
- 데이터 베이스에 대한 권한 부여 - [Schema Privileges] 탭 사용

#### – Shopdb

- » Object Rights에서는 <SELECT>, <INSERT>, <UPDATE>, <DELETE> 체크
- » DDL Rights에서는 <CREATE ROUTINE>과 <ALTER ROUTINE> 체크

#### – employees – SELECT 권한만 부여

# 데이터 타입



# MySQL 데이터 타입

## ◎ CHAR (M)

CHAR 데이터 타입은 고정된 길이의 문자열을 나타내는데 사용된다. 하나의 CHAR 문자열은 1-255 자 범위의 문자를 저장할 수 있다.

ex. `car_model CHAR(10);`

## ◎ VARCHAR (M)

VARCHAR 데이터 타입은 가변적인 길이의 문자열을 저장하므로 CHAR 보다는 좀더 융통성 있는 데이터 타입이다. VARCHAR 문자열은 1-255 자 범위의 문자를 저장할 수 있다.

(CHAR 는 포함된 데이터의 크기에는 관계없이 이미 지정된 가변적인 전체의 길이를 저장하는 반면에 VARCHAR 는 오직 들어가는 데이터의 양만을 저장하므로 데이터베이스 파일의 크기를 줄 일수 있다.)

ex. `car_model VARCHAR(10);`

## ◎ INT (M) [Unsigned]

INT 데이터타입은 -2147483648 에서 2147483647 사이의 정수를 저장한다.

"unsigned" 옵션과 함께 0부터 4294967295 범위의 정수를 나타낼 수도 있다.

ex. `light_years INT;`

ex. `light_years INT unsigned;`

# MySQL 데이터 타입

## ◎ FLOAT [(M,D)]

FLOAT는 다소 정확한 숫자의 표시가 필요할 때 사용되어지며, 작은 양의 소수점 숫자를 나타낸다.

ex. rainfall FLOAT (4,2);

이것은 소수점 값이 될 수 있는 일년간 평균강수량을 나타낼 수 있다. 좀 더 명확하게 말하면 FLOAT (4,2) 는 4개의 저장할 수 있는 최대 자리 수와 2개의 소수점 이하의 자리 수를 가리킨다.

주의) FLOAT 는 어림잡은 수이기 때문에 MySQL 내에 포함 된 정수의 값이 아닌 데이터 타입으로 돈의 값을 나타낼 때에는 DECIMAL을 사용하는 것이 더 현명한 방법이다.

## ◎ DATE

날짜와 관련된 정보를 저장한다. 디폴트 형식은 'YYYY-MM-DD' 이며, '0000-00-00' 에서 '9999-12-31'까지의 범위를 갖는다.

ex. the\_date DATE;

## ◎ TEXT / BLOB

text 와 blob 데이터 타입은 255 - 65535 자의 문자열을 저장할 때 사용된다. 기사와 같은 것을 저장하기에 유용하다. 그러나 VARCHAR 와 CHAR처럼 딱 잘라 비교할 수는 없다. 단지 BLOB 와 TEXT 사이에 차이점이 있다면 BLOB은 변하기 쉬운 경우에 비유할 수 있고, TEXT는 영향을 받지 않는 무감각한 경우에 비유할 수 있다

# MySQL 데이터 타입

## ◎ SET

지정된 값으로부터 어떤 주어진 값을 선택하는, 정해진 문자열의 데이터 타입으로 그것은 하나의 값이 될 수도 있고 여러개의 값을 가질 수도 있다. 64개의 값까지 지정할 수 있다.

ex. transport SET ("truck", "wagon") NOT NULL;

## ◎ ENUM

SET 데이터 타입과 비슷한 특징을 갖는 정해진 문자열의 데이터 타입이지만 선택할 수 있는 값이 하나로만 정해져 있다는 점이 다르다. 한 공간의 바이트만을 가지므로 테이블내의 시간과 공간을 절약할 수 있다.

ex. transport ENUM ("truck", "wagon") NOT NULL;

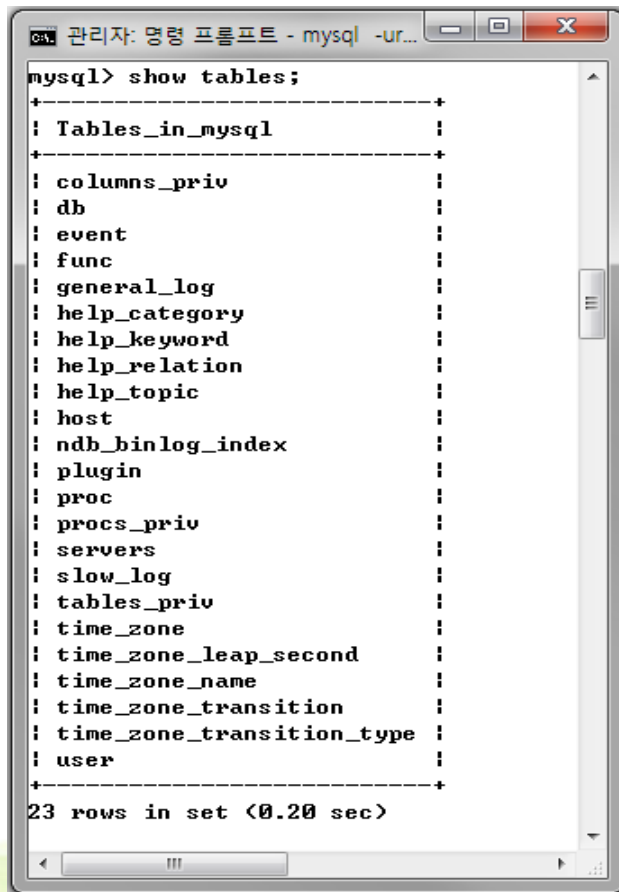
# 테이블 생성 및 권한



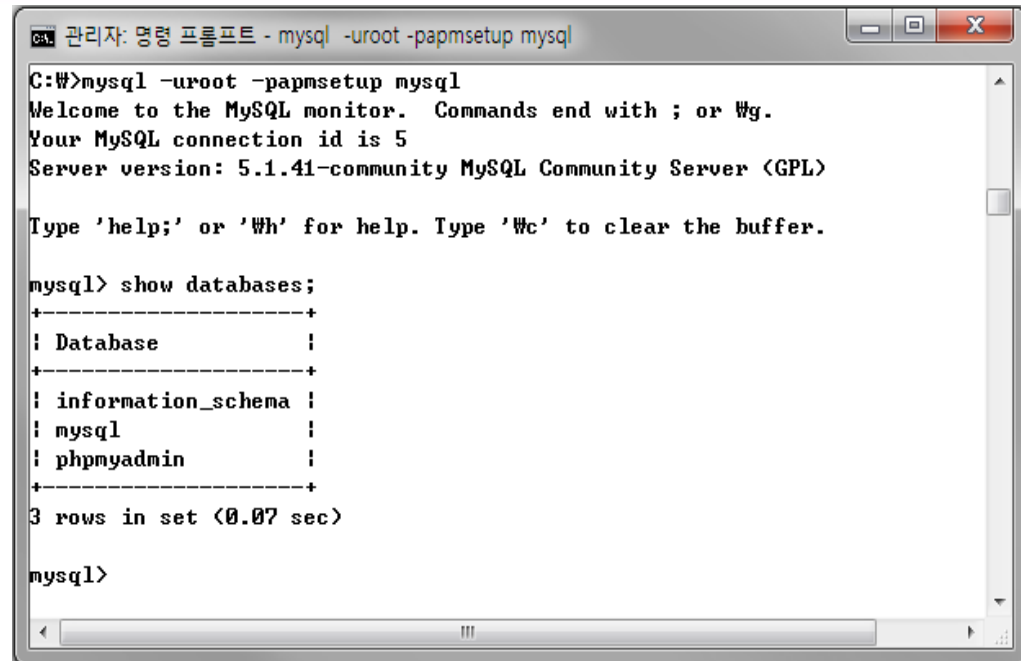
# MySQL의 시작

- ❖ 데이터베이스에 관리자 계정으로 접속
- ❖ 존재하는 데이터베이스 목록보기  
`mysql> show databases;`

관리자 계정으로 접속



```
관리자: 명령 프롬프트 - mysql -ur...
mysql> show tables;
+-----+
| Tables_in_mysql |
+-----+
| columns_priv    |
| db              |
| event           |
| func            |
| general_log     |
| help_category   |
| help_keyword    |
| help_relation   |
| help_topic      |
| host            |
| ndb_binlog_index|
| plugin          |
| proc            |
| procs_priv      |
| servers         |
| slow_log        |
| tables_priv     |
| time_zone       |
| time_zone_leap_second|
| time_zone_name  |
| time_zone_transition|
| time_zone_transition_type|
| user            |
+-----+
23 rows in set (0.20 sec)
```



```
관리자: 명령 프롬프트 - mysql -uroot -papmsetup mysql
C:\W>mysql -uroot -papmsetup mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.1.41-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\w' to clear the buffer.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql        |
| phpmyadmin    |
+-----+
3 rows in set (0.07 sec)

mysql>
```

mysql 데이터베이스의 테이블

테이블 목록보기

`mysql> show tables;`

# MySQL의 시작

## 데이터베이스 생성 명령

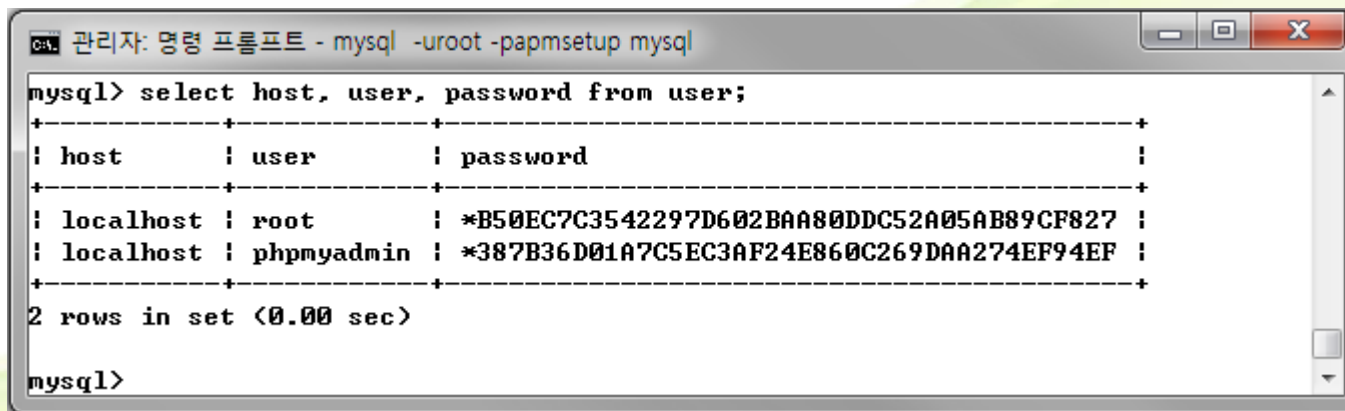
```
mysql> create database 데이터베이스명;
```

**CREATE DATABASE 데이터베이스명 DEFAULT CHARACTER SET utf8  
COLLATE utf8\_general\_ci;  
ALTER DATABASE 데이터베이스\_이름 DEFAULT CHARACTER  
SET utf8 COLLATE utf8\_general\_ci;**

## 테이블 구조 출력 명령

```
mysql> desc 테이블명;
```

user 테이블에 등록된 계정 목록 확인



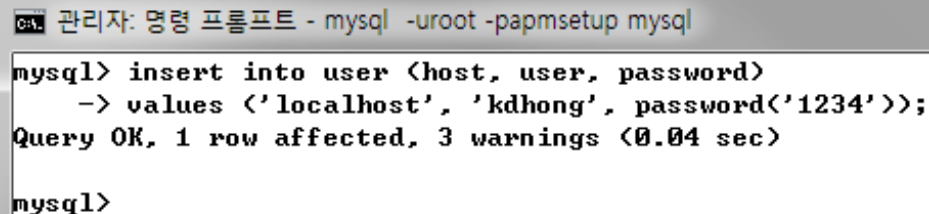
```
관리자: 명령 프롬프트 - mysql -uroot -pappmsetup mysql
mysql> select host, user, password from user;
+-----+-----+-----+
| host      | user      | password                                     |
+-----+-----+-----+
| localhost | root      | *B50EC7C3542297D602BAA80DDC52A05AB89CF827 |
| localhost | phpmyadmin | *387B36D01A7C5EC3AF24E860C269DAA274EF94EF |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

# MySQL의 시작

## 필드에 새로운 데이터 입력

```
mysql> insert into 테이블명 (필드1, 필드2, 필드3) values  
      (필드1_값, 필드2_값, 필드3_값);
```

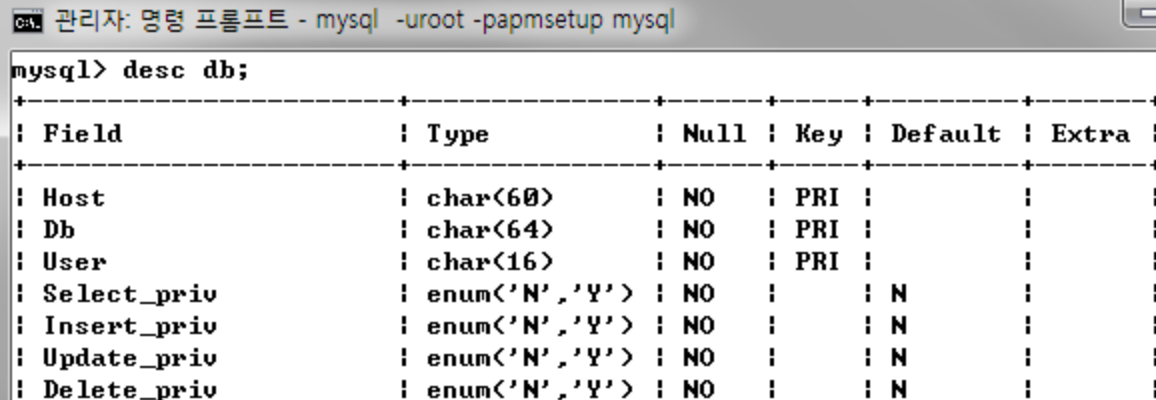


```
C:\> 관리자: 명령 프롬프트 - mysql -uroot -pappmsetup mysql  
mysql> insert into user (host, user, password)  
      -> values ('localhost', 'kdhong', password('1234'));  
Query OK, 1 row affected, 3 warnings (0.04 sec)  
  
mysql>
```

user 테이블에 계정(kdhong)과 비밀번호(1234) 등록

user 테이블에 계정 등록

```
mysql> insert into user (host, user, password)  
      -> values ('localhost', 'kdhong', password('1234'));
```

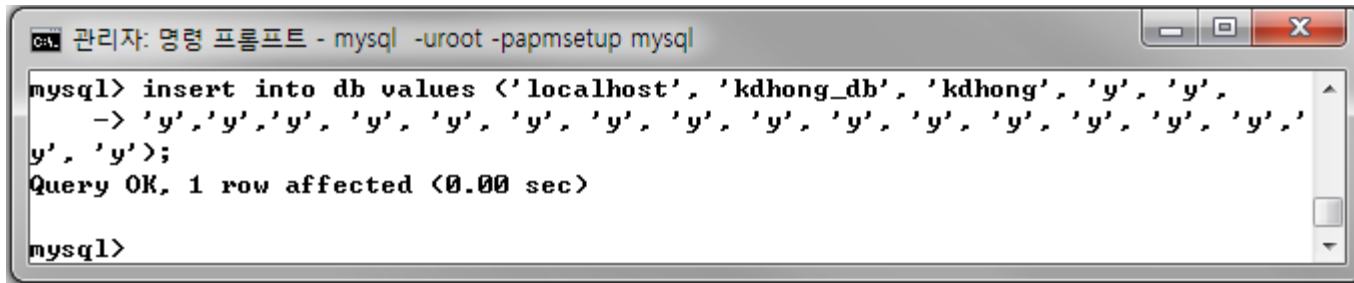


```
C:\> 관리자: 명령 프롬프트 - mysql -uroot -pappmsetup mysql  
mysql> desc db;  
+-----+-----+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| Host | char(60) | NO | PRI | | |  
| Db | char(64) | NO | PRI | | |  
| User | char(16) | NO | PRI | | |  
| Select_priv | enum('N','Y') | NO | | N | |  
| Insert_priv | enum('N','Y') | NO | | N | |  
| Update_priv | enum('N','Y') | NO | | N | |  
| Delete_priv | enum('N','Y') | NO | | N | |
```

테이블 구조 파악  
mysql> desc db;



# MySQL의 시작



```
C:\> 관리자: 명령 프롬프트 - mysql -uroot -pampsetup mysql

mysql> insert into db values ('localhost', 'kdhong_db', 'kdhong', 'y', 'y',
-> 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y',
y', 'y');
Query OK, 1 row affected (0.00 sec)

mysql>
```

데이터베이스 사용 권한 설정

db 테이블에 사용 권한 설정

```
mysql> insert into db values ('localhost','kdhong_db',
kdhong', 'y', 'y', 'y', 'y', 'y', 'y', y',
'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y',
'y', 'y', 'y', 'y');
```

user, db 테이블의 변경된 내용 적용

```
mysql> flush privileges;
```



# MySQL의 시작

## 데이터베이스 접속 명령

```
mysql -u계정 -p비밀번호 데이터베이스명
```

## 데이터베이스 생성 명령

```
create database 데이터베이스명;
```

## 데이터베이스 목록 출력 명령

```
show databases;
```

## 데이터베이스 삭제 명령

```
drop database 데이터베이스명;
```

**MySql을 설치하고 root 계정으로 접속 한 후 데이터베이스를 생성**  
**grant all privileges on 데이터베이스이름.\* to 계정@'%' identified by '비밀번호'**

❖ %대신에 ip를 기재하면 특정 ip에서만 접속이 허용됩니다.

# Database사용

## ❖ USE 구문

- SELECT문 학습 위해 사용할 데이터베이스 지정
- 지정해 놓은 후 특별히 다시 USE문 사용하거나 다른 DB를 사용하겠다고 명시하지 않는 이상 모든 SQL문은 지정 DB에서 수행

```
USE 데이터베이스_이름;
```

- Workbench 에서 직접 선택해서 사용도 가능
  - [Navigator]의 [Schemas] 탭
    - employees 데이터베이스를 더블 클릭하면 진한 글자 전환
    - 왼쪽 아래 ‘Active schema changed to employees’ 메시지

# 데이터베이스 관련 명령

## DCL(Data Control Language)

### ① 계정 생성

```
create user 'jspexam'@'localhost' identified by 'jsppw';  
create user 'jspexam'@'%' identified by 'jsppw';
```

### ② 권한 부여

grant 권한 on DB명.table명 to 유저명 [identified by ‘암호’][with grant option]

ex) grant all privileges on \*.\* to kim identified by ‘1234’ with grant option;

```
create user 'root'@'%' identified by 'mysql';  
grant all privileges on *.* to 'root'@'%' with grant option;  
grant all on *.* to 'kim'@'localhost';  
grant all on test.* to 'park'@'%';
```

### ③ 권한박탈

revoke 권한 on DB명 from 유저명;

ex) revoke all on test.\* from park;

# 데이터베이스 관련 명령

## ④ mysql DB

mysql데이터베이스 권한에 관한 규정

- . user table : mysql권한에 대한 정보규정
- . db : db권한에 대한 정보 저장(root제외), host이름 %는 어디서나 접속 가능
- . tables-priv : 특정 table접속 가능권한
- . columns-priv : 특정 column 접속 가능 권한
- . host : 특정 computer 접속 가능 권한
- . root password 변경  
update user set password=password('1234') where user='root';
- . User삭제 delete from user where user='lee';

## ⑤ 권한 적용

flush privileges;

# MySQL의 데이터 형식

## ❖ MySQL에서 지원하는 데이터 형식의 종류

### ■ 숫자 데이터 형식

데이터 형식	바이트 수	숫자 범위	설명
BIT(N)	N/8		1~64bit를 표현. b'0000' 형식으로 표현
TINYINT	1	-128~127	정수
★SMALLINT	2	-32,768~32,767	정수
MEDIUMINT	3	-8,388,608~8,388,607	정수
★INT INTEGER	4	약 -21억~+21억	정수
★BIGINT	8	약 -900경~+900경	정수
★FLOAT	4	-3.40E+38~-1.17E-38	소수점 아래 7자리까지 표현
★DOUBLE REAL	8	-1.22E-308~1.79E+308	소수점 아래 15자리까지 표현
★DECIMAL(m, [d]) NUMERIC(m, [d])	5~17	$-10^{38}+1 \sim +10^{38}-1$	전체 자릿수(m)와 소수점 이하 자릿수(d)를 가진 숫자형 예) decimal(5, 2)은 전체 자릿수를 5자리로 하되, 그 중 소수점 이하를 2자리로 하겠다는 의미

# MySQL의 데이터 형식

## ❖ MySQL에서 지원하는 데이터 형식의 종류

### ■ 문자 데이터 형식

데이터 형식		바이트 수	설명
★CHAR(n)		1~255	고정길이 문자형. n을 1부터 255까지 지정. character의 약자 그냥 CHAR만 쓰면 CHAR(1)과 동일
★VARCHAR(n)		1~65535	가변길이 문자형. n을 사용하면 1부터 65535 까지 지정. Variable character의 약자
BINARY(n)		1~255	고정길이의 이진 데이터 값
VARBINARY(n)		1~255	가변길이의 이진 데이터 값
TEXT 형식	TINYTEXT	1~255	255 크기의 TEXT 데이터 값
	TEXT	1~65535	N 크기의 TEXT 데이터 값
	MEDIUMTEXT	1~16777215	16777215 크기의 TEXT 데이터 값
	★LONGTEXT	1~4294967295	최대 4GB 크기의 TEXT 데이터 값
BLOB 형식	TINYBLOB	1~255	255 크기의 BLOB 데이터 값
	BLOB	1~65535	N 크기의 BLOB 데이터 값
	MEDIUMBLOB	1~16777215	16777215 크기의 BLOB 데이터 값
	★LONGBLOB	1~4294967295	최대 4GB 크기의 BLOB 데이터 값
ENUM(값들...)		1 또는 2	최대 65535개의 열거형 데이터 값
SET(값들...)		1, 2, 3, 4, 8	최대 64개의 서로 다른 데이터 값

# MySQL의 데이터 형식

## ❖ MySQL에서 지원하는 데이터 형식의 종류

- 날짜와 시간 데이터 형식

데이터 형식	바이트 수	설명
★DATE	3	날짜는 1001-01-01~9999-12-31까지 저장되며 날짜 형식만 사용 'YYYY-MM-DD' 형식으로 사용됨
TIME	3	-838:59:59.000000~838:59:59.000000까지 저장되며 'HH:MM:SS' 형식으로 사용
★DATETIME	8	날짜는 1001-01-01 00:00:00~9999-12-31 23:59:59까지 저장되며 형식은 'YYYY-MM-DD HH:MM:SS' 형식으로 사용
TIMESTAMP	4	날짜는 1001-01-01 00:00:00~9999-12-31 23:59:59까지 저장되며 형식은 'YYYY-MM-DD HH:MM:SS' 형식으로 사용. time_zone 시스템 변수와 관련이 있으며 UTC 시간대로 변환하여 저장
YEAR	1	1901~2155까지 저장. 'YYYY' 형식으로 사용

	DATE			TIME			DATETIME
▶	2020-10-19		▶	12:35:29		▶	2020-10-19 12:35:29

# MySQL의 데이터 형식

## ❖ MySQL에서 지원하는 데이터 형식의 종류

### ■ 기타 데이터 형식

- JSON 데이터 형식은 MySQL 5.7.8 이후부터 지원

데이터 형식	바이트 수	설명
GEOMETRY	N/A	공간 데이터 형식으로 선, 점 및 다각형 같은 공간 데이터 개체를 저장하고 조작
JSON	8	JSON(JavaScript Object Notation) 문서를 저장

### ■ LONGTEXT, LONGBLOB

- LOB (Large Object, 대량의 데이터) 을 저장
- LONGTEXT, LONGBLOB 데이터 형식 지원
- 지원되는 데이터 크기는 약 4GB의 파일을 하나의 데이터로 저장 가능



# 테이블

## ❖ 제약 조건

### ■ 제약 조건 (Constraint) 이란?

- 데이터의 무결성을 지키기 위한 제한된 조건 의미
- 특정 데이터를 입력할 때 어떠한 조건을 만족했을 때에 입력되도록 제약
  - Ex) 동일한 주문등록 번호로 회원 중복 가입하지 못함
- 데이터 무결성을 위한 제약조건
  - PRIMARY KEY 제약 조건
  - FOREIGN KEY 제약 조건
  - UNIQUE 제약 조건
  - DEFAULT 정의
  - NULL 값 허용

# 테이블

## ❖ 데이터 무결성 위한 제약 조건

### ■ PRIMARY KEY 제약 조건

- ‘기본 키 Primary Key’ 의 개념
  - 테이블에 존재하는 많은 행의 데이터를 구분할 수 있는 식별자
  - 중복되어서도 안되며 비어져서도 안됨
  - Ex) 회원 테이블의 회원 아이디, 학생 테이블의 학번
- 기본 키로 생성한 것은 자동으로 클러스터형 인덱스 생성
- 테이블에서는 기본 키를 하나 이상의 열에 설정 가능
- 기본 키 생성 방법

```
CREATE TABLE userTbl
( userID char(8) NOT NULL PRIMARY KEY,
  name nvarchar(10) NOT NULL,
  --- 중간 생략 ---
```

# 테이블

## ❖ 데이터 무결성 위한 제약 조건

### ■ 외래 키 제약 조건

- 두 테이블 사이의 관계 선언 : 데이터의 무결성을 보장해 주는 역할
- 외래 키 관계를 설정하면 하나의 테이블이 다른 테이블에 의존
- 설정된 외래 키 제약 조건은 SHOW INDEX FROM buyTbl문으로 확인
- ON DELETE CASCADE / ON UPDATE CASCADE
- 기존 테이블의 데이터가 변경되었을 때 외래 키 테이블도 자동 적용되도록 설정

### ■ UNIQUE 제약 조건

- '중복되지 않는 유일한 값'을 입력해야 하는 조건
- PRIMARY KEY와 거의 비슷하며 차이점은 UNIQUE는 NULL 값 허용
  - NULL은 여러 개가 입력되어도 상관 없음
  - Ex) 회원 테이블의 예를 든다면 주로 Email 주소 Unique로 설정

## ❖ 데이터 무결성 위한 제약 조건

### ■ DEFAULT 정의

- 값 입력하지 않았을 때 자동으로 입력되는 기본 값 정의하는 방법

### ■ Null 값 허용

- NULL 값을 허용하려면 NULL을, 허용하지 않으려면 NOT NULL 사용
  - PRIMARY KEY가 설정된 열에는 생략하면 자동으로 NOT NULL

# 테이블 생성

## 테이블(The Bigger Picture: Tables)

```
CREATE TABLE sawon (num int not null primary key, name VARCHAR (15),  
phone INT, age int, sex enum('man','woman') not null default 'woman');
```

테이블이 성공적으로 만들어지면 다음과 같은 화면이 출력될 것이다.

Query OK, 0 rows affected (0.10 sec)

주의(1) : 서로 다른 두 개의 테이블은 같은 이름을 가질 수 없다.

주의(2) : 각각의 데이터 공간은 컬럼(column)이라는 개체의 속성으로 나타낸다.

1) 컬럼(Column)의 특징 :

하나의 이름이 숫자로만 구성될 수 없다. 하나의 이름은 숫자로 시작할 수 있다.

하나의 이름은 64자까지 가능하다.

2) 테이블의 다른 옵션들 :

. 기본 키(Primary Key) :

다른 속성들로부터 하나의 레코드를 유일하게 식별하는데 사용되어지며, 두 개의 레코드는 똑같은 기본 키를 가질 수 없다. 이것은 두 개의 레코드가 중복되어지는 실수를 없애려고 할 때 아주 적합하게 사용된다.

. auto\_Increment :

이 함수(function)를 가진 컬럼(column)은 레코드에 삽입될 때 자동적으로 한 개씩 값이 (previous + 1) 증가한다. data가 'NULL'값 일 때도 컬럼(column)에 자동적으로 한 개씩 값이 증가하여 삽입된다.

. NOT NULL :

NULL값(값이 없는)을 절대로 허용하지 않는 컬럼(column)을 의미한다. 숫자는 0 문자는 null이 들어감, 날짜 0000-00-00

ex) soc\_sec\_number INT PRIMARY KEY;

Ex) ID\_NUMBER INT AUTO\_INCREMENT;

# 테이블 생성

## . auto\_Increment :

이 함수(function)를 가진 컬럼(column)은 레코드에 삽입될 때 자동적으로 한 개씩 값이 (previous + 1) 증가한다. data가 'NULL'값 일 때도 컬럼(column)에 자동적으로 한 개씩 값이 증가하여 삽입된다.

## . NOT NULL :

NULL값(값이 없는)을 절대로 허용하지 않는 컬럼(column)을 의미한다.

숫자는 0 문자는 null이 들어감, 날자 0000-00-00

### 데이터베이스 테이블 생성 명령

```
mysql> create table 테이블명(  
    필드명1 타입 컬럼제약조건,  
    필드명2 타입,  
    필드명3 타입,  
    .....  
    제약 조건);
```

주소록 테이블(friend) 만들기

```
create table friend (  
    num int not null, name char(10),  
    address char(80),tel char(20),  
    email char(20), primary key(num));
```

# 테이블 생성

- ❖ 제약조건: unique, primary key, not null, default 값, check 조건, foreign key
- ❖ create table School(code int not null primary key,  
name varchar(15));
- ❖ create table Junior(stu\_id int not null auto\_increment primary key,  
name varchar(15), school\_code int,  
foreign key(school\_code) references School(code)  
on delete cascade  
on update cascade);
- ❖ create table Senior(stu\_id int not null auto\_increment primary key,  
name varchar(15), school\_code int,  
foreign key(school\_code) references School(code)  
on delete set null on update cascade);

## 데이터베이스 테이블 관련 명령

## 테이블 변경하기 (Alter table)

① 오피스

- 새로운 열 추가하기, 열의 속성 바꾸기, 열 삭제하기, 테이블 이름 바꾸기

② alter table 테이블 변경명령;

Example: 테이블 이름 변경하기 (Rename the table)

**변경명령** : rename 새로운 테이블명; add 필드형 자료형 옵션

drop 필드명;      change 필드명 새로운 필드명 옵션;

modify 필드명 자료형 옵션;      add primary key (필드명);

drop primary key;      add index 인덱스명 (필드명);

drop index 인덱스명;

## 확인

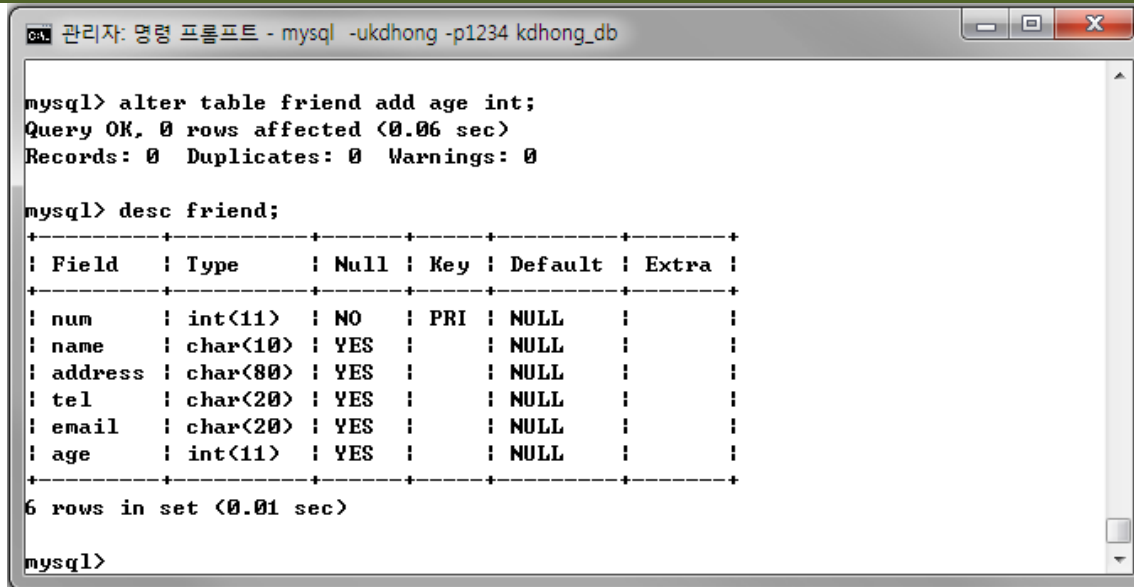
desc table명;

```
show index from table명;
```

# 데이터베이스 테이블 관련 명령

## 데이터베이스 테이블의 필드 추가 명령

alter table 테이블명 add 새로운 필드명 필드타입 [first 또는 after 필드명];



```
관리자: 명령 프롬프트 - mysql -ukdhong -p1234 kdhong_db

mysql> alter table friend add age int;
Query OK, 0 rows affected (0.06 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> desc friend;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| num   | int(11) | NO   | PRI | NULL    |       |
| name  | char(10) | YES  |     | NULL    |       |
| address | char(80) | YES  |     | NULL    |       |
| tel   | char(20) | YES  |     | NULL    |       |
| email | char(20) | YES  |     | NULL    |       |
| age   | int(11) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql>
```

## 새로운 필드 추가 명령

ex) 앞서 만든 friend 테이블 나이 필드를 정수형으로 추가

1. mysql> alter table friend add age int;
2. mysql> desc friend;



# SQL 기본

# 샘플 데이터

❖ 오라클 scott/tiger 계정의 emp, dept, salgrade 테이블을 생성하고 데이터 가져옴 **SHOW VARIABLES LIKE 'char%';**

## 1) 테이블 생성

```
create table dept (deptno int not null primary key,  
  dname varchar(30), loc varchar(30));
```

```
create table emp (empno int not null primarykey,  
  eame varchar(20), job varchar(30), mgr int, hiredate date,  
  sal float(7,2), comm float(7,2), deptno int);
```

```
create table salgrade (grade int not null primary key,  
  losal int, hisal int);
```

```
alter table dept convert to charset utf8;
```

```
alter table emp convert to charset utf8;
```

## 2) Load

```
set global local_infile=1;
```

```
load data [local] infile '파일네임' replace into table emp;
```

```
load data [local] infile '파일네임' into table emp;
```

```
load data [local] infile 'c:/gov/mysql/sawon.csv' into table emp  
character set euckr fields terminated by ',' lines terminated by '\r\n';
```

# Data BACKUP및 복구(mysql8.\*)

## 1. 데이터 위치 보기

```
show variables like "secure_file_priv";  
SELECT @@GLOBAL.secure_file_priv;
```

## 2. 데이터를 해당 위치로 변경

C:\ProgramData\MySQL\MySQL Server 8.0\Uploads

## 3. Load명령(tab)

```
load data infile  
'C:/ProgramData/MySQL/MySQL Server  
8.0/Uploads/emp.txt'  
into table emp character set euckr;
```

## 4. Load명령(csv)

```
load data infile  
'C:/ProgramData/MySQL/MySQL Server  
8.0/Uploads/emp2.csv'  
into table emp2 character set euckr fields terminated by  
' ,' lines terminated by '\r\n';
```

# 데이터의 변경을 위한 SQL문

## ❖ 데이터의 삽입 : INSERT

### ■ INSERT문의 기본

- 테이블 이름 다음에 나오는 열 생략 가능
  - 생략할 경우에 VALUE 다음에 나오는 값들의 순서 및 개수가 테이블이 정의된 열 순서 및 개수와 동일해야 함

### ■ 자동으로 증가하는 AUTO\_INCREMENT

- INSERT에서는 해당 열이 없다고 생각하고 입력
  - INSERT문에서 NULL 값 지정하면 자동으로 값 입력
- 1부터 증가하는 값 자동 입력
- 적용할 열이 PRIMARY KEY 또는 UNIQUE 일 때만 사용가능
- 데이터 형은 숫자 형식만 사용 가능

## ❖ 데이터의 삽입 : INSERT

### ■ 대량의 샘플 데이터 생성

- INSERT INTO... SELECT 구문 사용
- 다른 테이블의 데이터를 가져와 대량으로 입력하는 효과
- SELECT문의 열의 개수 = INSERT 할 테이블의 열의 개수

# 데이터의 변경을 위한 SQL문

## ❖ 데이터의 수정 : UPDATE

- 기존에 입력되어 있는 값 변경하는 구문

```
UPDATE 테이블이름  
SET 열1=값1, 열2=값2 ...  
WHERE 조건 ;
```

- WHERE절 생략 가능하나 테이블의 전체 행의 내용 변경

## ❖ 데이터의 삭제 : DELETE FROM

- 행 단위로 데이터 삭제하는 구문
- DELETE FROM 테이블이름 WHERE 조건;
- 테이블을 삭제하는 경우의 속도 비교
  - DML문인 DELETE는 트랜잭션 로그 기록 작업 때문에 삭제 느림
  - DDL문인 DROP과 TRUNCATE문은 트랜잭션 없어 빠름

# DML(Data Manipulation Language)

## ① 데이터 삽입

```
INSERT INTO PROJ VALUES ('MA2114;',',','B01',',',NULL,CURRENT DATE, NULL);
```

INSERT 문을 사용하면 테이블에 데이터 행을 추가할 수 있습니다.

이런 작업은 여러 가지 방법으로 수행할 수 있습니다. PROJECT 테이블에서 컬럼이 만들어진 것과 같은 순서로 각 컬럼에 대한 값과 컬럼 값을 지정해야 합니다.

PROJNO 컬럼에는 MA2114, PROJNAME 컬럼에는 공백, DEPTNO 컬럼에는 B01, RESPEMP 컬럼에는 공백, PRSTAFF 컬럼에는 NULL 값, PRSTDATE 컬럼에는 현재 날짜 값, PRENDATE 컬럼에는 NULL 값이 들어갑니다. 화면의 아래쪽에는 PROJ 테이블에 이 값들을 삽입한 결과가 보입니다. PRSTAFF와 PRENDATE 컬럼의 대신에 주목하십시오. 이 대신은 INSERT 문에 지정한 NULL 값을 나타냅니다.

```
INSERT INTO PROJ (DEPTNO, PROJNO, PROJNAME, RESPEMP PRSTDAT) VALUES  
( 'B01','MA2114;',',', ',,CURRENT DATE)
```

또는 컬럼 이름을 괄호로 묶어서 지정하는 INSERT 문을 사용할 수도 있습니다. 화면에 이 방법이 나타나 있습니다. 이 구문 형식을 사용하면 모든 컬럼을 지정할 필요가 없습니다. 값이 주어지지 않은 NULL 지정이 가능한 컬럼은 자동으로 NULL로 지정됩니다. 값이 해당 컬럼에 대응하면 컬럼을 특정 순서로 지정하지 않아도 됩니다.

# SELECT문

## ❖SELECT의 구문 형식

```
SELECT select_expr  
  [FROM table_references]  
  [WHERE where_condition]  
  [GROUP BY {col_name | expr | position}]  
  [HAVING where_condition]  
  [ORDER BY {col_name | expr | position}]
```



```
SELECT 열이름  
FROM 테이블이름  
WHERE 조건
```

# SELECT문

## ❖ SELECT와 FROM

### ■ SELECT \*

- 선택된 DB가 employees 라면 다음 두 쿼리는 동일

```
SELECT * FROM employees.titles;  
SELECT * FROM titles;
```

### ■ SELECT 열 이름

- 테이블에서 필요로 하는 열만 가져오기 가능
- 여러 개의 열을 가져오고 싶을 때는 콤마로 구분
- 열 이름의 순서는 출력하고 싶은 순서대로 배열 가능



# SELECT문

## ❖ DB, TABLE, 열의 이름이 확실하지 않을 때 조회하는 법

- 현재 서버에 어떤 DB가 있는지 보기
  - SHOW DATABASES;
- 현재 서버에 어떤 TABLE이 있는지 보기
  - 데이터베이스 employees 에 있는 테이블 정보 조회
    - **SHOW TABLE STATUS;**
  - 테이블 이름만 간단히 보기
    - **SHOW TABLES;**
- employees 테이블의 열이 무엇이 있는지 확인
  - DESCRIBE employees;
  - DESC employees;

## ❖ 문제가 생긴 DB 초기화하기

### ■ DB가 존재한다면 지우고 다시 만들기

```
DROP DATABASE IF EXISTS sqlDB; -- 만약 sqlDB가 존재하면 우선 삭제한다.  
CREATE DATABASE sqlDB;
```

- 계속 사용할 쿼리는 SQL 파일로 저장해서 재사용 가능하게 만들기
- 파일 내용을 불러다 쓰기 전에 모든 쿼리 창을 닫도록 함

# SELECT문

## ❖ 특정 조건의 데이터만 조회 - <SELECT FROM WHERE>

### ■ 기본적인 WHERE절

- 조회하는 결과에 특정한 조건 줘서 원하는 데이터만 보고 싶을 때 사용
- SELECT 필드이름 FROM 테이블이름 WHERE 조건식;
- 조건이 없을 경우 테이블의 크기가 클수록 찾는 시간과 노력이 증가

### ■ 관계 연산자의 사용

- '...했거나', '... 또는' - OR 연산자
- '...하고', '...면서', '... 그리고' - AND 연산자
- 조건 연산자(=, <, >, <=, >=, <>, != 등)와 관계 연산자(NOT, AND, OR 등)의 조합으로 알맞은 데이터를 효율적으로 추출

# 기본 SELECT

SELECT 문에서 SELECT, FROM, WHERE, 그 다음 ORDER BY와 같은 정해진

## 1) SELECT 문에서 모든 컬럼 가져오기

`select * from table명;`

`SELECT * FROM personal;`

`Select pno,pname,pay from personal;`

테이블의 모든 컬럼을 선택하는 간단한 방법은 SELECT \*를 지정하는 것입니다.

## 2) 행 제어하기

`SELECT * from personal where dno=10;`

`Select pno,pname,pay from personal where dno=20;`

`Select * from personal where job=' salesman' ;`

## 3) 비교 연산자

WHERE 절에서는 등호 연산자 외에도 기타 비교 연산자를 사용할 수 있습니다.

`>, <, >=, <=, =, != >= 100과 JOB <>'manager'식으로 사용합니다`

# 기본 SELECT

## 4) NULL특성 과 NOT NULL WITH DEFAULT

NULL 특성은 관계형 이론의 일부입니다.

컬럼이 NOT NULL인 경우에는 값을 가지고 있어야 하며, 숫자 컬럼의 경우는 0, 문자 컬럼의 경우는 공백이 올 수 있습니다. NULL 지정이 가능한 컬럼에 값을 제공할 필요는 없습니다. NULL 지정이 가능한 것으로 지정된 컬럼에는 해당 필드가 값을 가지고 있는지 여부를 나타내는 1바이트 FLAG가 추가됩니다.

NULL로 플래그가 지정된 필드는 알 수 없는 값을 포함하고 있습니다. 0은 숫자 필드에서, 공백은 문자 데이터에서 의미 있는 값으로 사용되기 때문에 이 알 수 없는 값은 0이나 공백은 아닙니다. NULL 값은 문자 그대로 “알 수 없는” 값입니다.

컬럼을 NOT NULL WITH DEFAULT로 지정할 수도 있습니다. 이 컬럼은 값이 필요하며, 값을 지정하지 않으면 시스템이 기본값을 제공합니다.

- 숫자 데이터의 경우 기본값은 0입니다.
- 문자 데이터의 기본값은 고정 길이의 경우 공백(BLANK)이며 가변 길이의 경우 제로 길이(ZERO LENGTH)입니다.

NULL 값을 알 수 없기 때문에 이 값을 검색하려면 WHERE 절에서 특수 구문을 사용해야 합니다.

```
Select * from personal where bonus is null;
```

# 기본 SELECT

## 5) 다중 조건식

**select \* from personal where dno > 20 and pno > 1112;**

**select \* from personal where dno > 20 or pno > 1112;**

**select \* from personal where dno > 20 && pno > 1112;**

**select \* from personal where dno > 20 || pno > 1112;**

## 6) IN

**WHERE job IN( 'sales' , 'manager' , 'clerk' )**

## 7) BETWEEN

**where dno between 20 and 30; >= 20 그리고 <= 30와 같습니다.**

**문자 값의 범위를 선택할 수도 있습니다.**

## 8) 부분 검색

**SQL은 LIKE 키워드와 함께 사용되는 % 및 \_와 같은 특수 기호를 사용**

**select \* from personal where job like 's%';**

**Select \* from personal where job like '%r';**

**Select \* from dno like '3\_';**

## 9) 부정

**WHERE NOT LIKE 'G%'; Where not between**

# 기본 SELECT

## 2. 테이블 행의 결과

### 1) ORDER BY

ORDER BY 절은 컬럼 값을 기준으로 결과를 정렬합니다.  
`select * from personal order by bonus;`

`select * from personal order by bonus, dno;`

하나 이상의 컬럼에 ORDER BY를 수행할 수 있습니다. 지정된 첫 번째 컬럼에 따라 결과가 정렬, 이 결과가 동일하면 ORDER BY의 두 번째 컬럼에 따라 정렬  
이 예제에서 결과는 JOB을 기준으로 오름차순으로 정렬됩니다. 오름차순은 기본 값입니다. 다른 방법으로 ASC를 지정할 수 있습니다.

`Select * from personal order by dno desc;`

이 예제에서 볼 수 있는 것처럼 먼저 YEARS DESC를 지정하여 결과의 순서를 변경할 수 있습니다.

### 2) DISTINCT

`select dno from personal;`

중복 값을 제거하고 회사 내의 다른 부서만 보려면 SELECT 바로 뒤에 키워드 DISTINCT를 지정합니다.

`SELECT DISTINCT Dno FROM personql;`

# 기본 SELECT

## 3) 표현식을 이용한 값

SELECT 문에서 수학 계산을 사용할 수 있습니다. +(더하기), -(빼기), \*(곱하기), /(나누기)를 사용합니다.

2개의 컬럼을 더할 수도 있습니다. 이 예제에서는 부서의 각 직원별 급여와 커미션을 알아보려고 합니다. 또한 총 급여를 알기 위해 커미션이 추가된 급여를 보고 싶을 수도 있습니다.

```
select pname,pay,pay + bonus from personal;
```

※. Null + 값 = null

## 4) AS

AS 절을 사용하여 계산된 컬럼에 이름을 지정하고 다른 컬럼에 더 읽기 쉬운 이름을 지정할 수 있습니다.

```
SELECT pname,pay,pay+bonus as income from personal;
```

```
SELECT pname,pay,pay+bonus as 'month income' from  
personal;
```

```
SELECT pname,pay,pay+bonus as income from personal order  
by 1;
```

```
SELECT pname,pay,pay+bonus as 'month income' from  
personal order by 'month income' ;
```



# 컬럼함수 / GROUP BY

## 1) 컬럼 함수란 무엇인가?

이 함수들은 SUM, AVG, MIN, MAX, COUNT입니다. 컬럼 함수는 컬럼 값들을 단일 값으로 요약하거나 집계.

## 2) 컬럼 함수 사용

```
SELECT SUM(pay), AVG(pay), MIN(pay), MAX(pay) FROM personal;  
AVG(pay + bonus)을 사용 가능
```

## 3) COUNT(\*) 컬럼 함수

```
select count(*), count(distinct dno) from personal;  
SELECT COUNT(DISTINCT dno), COUNT(*), AVG(pay) FROM  
personal WHERE pay > 18000;
```

## 4) GROUP BY

```
Select sum(pay) from personal;  
Select dno,sum(pay) from personal group by dno;  
Select dno,sum(pay) from personal where pay > 1500 group by dno;  
SELECT DEPT, SUM(pay) AS TOTAL_SALARY, SUM(bonus) AS  
TOTAL_COMM FROM STAFF WHERE JOB <> 'manager' GROUP BY  
dno;
```



# 컬럼함수 / GROUP BY

**SELECT에 지정된 모든 항목은 일관성을 지녀야 합니다. 즉, 모든 세부 열만을 선택하거나 집계값 만을 선택해야 합니다.**

**세부 열과 집계값을 혼용하는 경우 모든 세부 열은 GROUP BY 문에 의해 포함되어야 합니다. 이 SELECT 문에서 GROUP BY를 실행할 때 그 결과에 유의하십시오.**

**SELECT DEPT, SUM(SALARY), SUM(COMM.) FROM STAFF WHERE  
JOB <> 'MGR' GROUP BY DEPT;**

**SELECT dno, JOB FROM STAFF WHERE JOB <> 'MGR' GROUP BY  
dno;**

**이 SELECT 문은 실행되지 않고 오류가 발생합니다.**

**모든 데이터를 집계하려면 GROUP BY 절에 모든 세부 사항을 지정**

**이 SELECT 문은 GROUP BY DEPT, JOB S를 사용하는 경우에만 실행됩니다**

# SELECT문

## ❖ GROUP BY 및 HAVING 그리고 집계 함수

- WHERE와 비슷한 개념으로 조건 제한
- 집계 함수에 대해서 조건 제한하는 편리한 개념
- **HAVING절은 꼭 GROUP BY절 다음에 나와야 함 !!!**
- SELECT DEPT, SUM(pay) FROM personal GROUP BY dno;
- SELECT DEPT, SUM(pay) FROM personal GROUP BY dno  
HAVING SUM(pay) > 65000

## ■ ROLLUP

- 총합 또는 중간합계가 필요할 경우 사용'
- GROUP BY절과 함께 WITH ROLLUP문 사용
- Ex) 분류(groupName) 별로 합계 및
- 그 총합 구하기

	num	groupName	비용	
▶	1	NULL	60	
	10	NULL	60	
	12	NULL	60	
	NULL	NULL	180	소합계
	7	서적	75	
	8	서적	30	
	11	서적	15	
	NULL	서적	120	소합계
	5	의류	150	
	9	의류	50	
	NULL	의류	200	소합계
	2	전자	1000	
	3	전자	200	
	4	전자	1000	
	6	전자	800	
	NULL	전자	3000	소합계
	NULL	NULL	3500	총합계

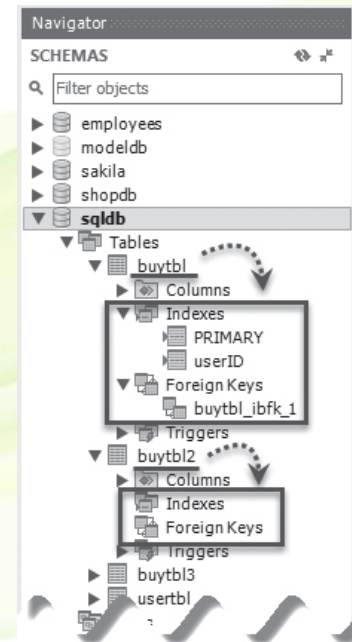
# SELECT문

❖ 원하는 순서대로 정렬하여 출력 : **ORDER BY**

■ 테이블을 복사하는 **CREATE TABLE ... SELECT**

- 테이블을 복사해서 사용할 경우 주로 사용
- CREATE TABLE 새로운테이블 (SELECT 복사할 열 FROM 기존테이블)
- 지정된 일부 열만 테이블로 복사하는 것도 가능
- PK나 FK 같은 제약 조건은 복사되지 않음

– **Workbench의 [Navigator]에서 확인 가능**



# Query

## 처리 순서

- |          |          |
|----------|----------|
| ① SELECT | FROM     |
| ② WHERE  | GROUP BY |
| ③ HAVING | ORDER BY |

## 4. SACLAR함수

### 1) 스칼라 함수의 소개

집계 값이 아닌 단일 값을 산출합니다. 스칼라 함수는 중첩될 수 있습니다.  
즉 내부에 다른 스칼라 함수를 포함할 수 있습니다.  
또한 스칼라 함수에 컬럼 함수를 포함할 수 있으며 그 반대의 경우도 가능합니다.

#### . 문자열 추출하기 - SUBSTR

```
SELECT SUBSTR(job, 1, 4) FROM personal;
```

```
SELECT * FROM personal WHERE SUBSTR(job,1,1) = 'm' ;
```

#### . 문자열 결합하기 - CONCAT

```
select concat(pname,' job is ',job) from personal;
```

# 숫자 관련 함수

- ▶ ABS(숫자) : 절대값 출력.

`select abs(123);`

- ▶ CEILING(숫자) : 값보다 큰 정수 중 가장 작은 수.

--양수일 경우는 소숫점 자리에서 무조건 반올림(4.0과 같은 0 값은 제외)

--음수일 경우는 소숫점 자리를 무조건 버림

`select ceiling(4.0);    select ceiling(4.1);    select ceil(4.9);`

- ▶ FLOOR(숫자) : 값보다 작은 정수 중 가장 큰 수[실수를 무조건 버림].

--음수일 경우는 [.0/.00/.000/...] 을 제외하고 무조건 소숫점을 버리고 반내림(?)

`select floor(4.0);    select floor(4.1);    select floor(4.9);    select floor(-4.6789);`

- ▶ ROUND(숫자, 자릿수) : 숫자를 소수점 이하 자릿수에서 반올림

--자릿수를 생략하면 소숫점이 5 이상일 때 반올림/자릿수를 지정하면 지정된 자릿수에서 반올림

`select round(4.5);    select round(4.55);`

`select round(-4.5);    select round(4.556);`

`select round(4.556,0);    select round(4.556,1);`

`select round(4.556,2);    select round(45.556,-1);    select round(455.556,-2);`

# 숫자 관련 함수

- ▶ TRUNCATE(숫자, 자릿수) : 숫자를 소수점 이하 자릿수에서 버림.
  - ➔ 소숫점 이전으로 정하면 소숫점이하는 버리고 나머지 값은 0 값으로 처리  
/ 예) truncate(9999, -3) --> 9000
  - ➔ 소숫점이하로 정하며, 해당숫자가 자릿수보다 소숫점이 모자랄경우 0 값 대치  
/ 예) truncate(999, 3) --> 999.000
  - 음수일 경우는 해당자릿수에서 소숫점을 버리면서 무조건 반올림  
==> (자릿수 숫자에서 이후 숫자가 0 일 경우는 제외 / 예) -4.0, 0 / -400, -2 / -4.1230, 4  
==> 음수도 소숫점이하로 정하며, 자릿수보다 소숫점이 모자랄경우 0 값 대치
  - ➔ 소숫점 이전으로 정하면 소숫점이하는 버리고 나머지 값은 역시 0 값으로 처리
- ▶ POW(X, Y) 또는 POWER(X, Y) : X의 Y승  
--소숫점이 있는 경우도 실행, 단 음수는 양수로 승처리  
select pow(-2.5, 2);    select pow(1.5, 2);
- ▶ MOD (분자, 분모) : 분자를 분모로 나눈 나머지를 구한다. (연산자 %와 같음)  
select mod(12, 5);    ==> 2            select 12%5;            ==> 2
- ▶ GREATEST(숫자1, 숫자2, 숫자3...) : 주어진 수 중 제일 큰 수 리턴.  
select greatest(100, 101, 90);
- ▶ LEAST(숫자1, 숫자2, 숫자3...) : 주어진 수 중 제일 작은 수 리턴.  
select least(100, 101, 90);
- ▶ INTERVAL(a, b, c, d, ...) : a(숫자)의 위치 반환  
--두 번째 이후는 오름차순 정렬이 되어야 함  
INTERVAL(5, 2, 4, 6, 8) ==> 2 (5는 4와 6사이에 존재, 4~6사이의 위치가 앞에서 2번째)  
select interval(4, 1, 2, 3, 5, 6);



# 문자 관련 함수

- ▶ ASCII(문자) : 문자의 아스키 코드값 리턴.

```
SELECT ASCII('문자');      select ascii('A');
```

- ▶ CONCAT('문자열1','문자열2','문자열3'...) : 문자열들을 이어준다.

```
select concat('ASP','PHP','SQL',' WEB STUDY');
```

- ▶ INSERT('문자열','시작위치','길이','새로운문자열') : 문자열의 시작위치부터 길이만큼 새로운 문자열로 대치

'시작위치' 와 '길이'는 문자열이 아니므로 작은따옴표로 굳이 묶어주지 않아도 된다.

```
select insert('MySql web study','7','3','offline');
```

```
select insert('MySql web study',7,3,'offline');
```

- ▶ REPLACE('문자열','기존문자열','바뀔문자열') : 문자열 중 기존문자열을 바뀔 문자열로 바꾼다.

```
select replace('MySql web study','web','offline');
```

- ▶ INSTR('문자열','찾는문자열') : 문자열 중 찾는 문자열의 위치값을 출력 -  
> 값이 존재하지 않으면 0값 리턴

```
select instr('MySql web study','s');
```

```
select instr('MySql web study','S');
```

# 문자 관련 함수

- ▶ LEFT('문자열',개수) : 문자열 중 왼쪽에서 개수만큼을 추출.  
select left('MySql web study',5);      select left('MySql web study','5');
- ▶ RIGHT('문자열',개수) : 문자열 중 오른쪽에서 개수만큼을 추출.  
select right('MySql web study',5);      select right('MySql web study','5');
- ▶ MID('문자열',시작위치,개수) : 문자열 중 시작위치부터 개수만큼 출력  
select mid('MySql web study',7,3);      select mid('MySql web study','7','3');
- ▶ SUBSTRING('문자열',시작위치,개수) : 문자열 중 시작위치부터 개수만큼 출력  
select substring('Mysql web study',11,5);  
select substring('Mysql web study','11','5');
- ▶ LTRIM('문자열') : 문자열 중 왼쪽의 공백을 없앤다.  
select ltrim('      web study');
- ▶ RTRIM('문자열') : 문자열 중 오른쪽의 공백을 없앤다.  
select rtrim('web study     ');
- ▶ TRIM('문자열') : 양쪽 모두의 공백을 없앤다.  
select trim('      web study     ');
- ▶ LCASE('문자열') 또는 LOWER('문자열') : 소문자로 바꾼다.  
select lcase('MYSQL');      select lower('MySQL');
- ▶ UCASE('문자열') 또는 UPPER('문자열') : 대문자로 바꾼다.  
select ucase('mySql');      select upper('mysql');
- ▶ REVERSE('문자열') : 문자열을 반대로 나열한다.  
예) REVERSE('abcde') ==> edcba  
select reverse('lqSyM');



# 날짜 관련 함수

- ▶ **NOW() 또는 SYSDATE() 또는 CURRENT\_TIMESTAMP() 현재 날짜와 시간 출력**

- ※ 함수의 상황이 숫자인지 문자열인지에 따라

YYYYMMDDHHMMSS 또는 'YYYY-MM-DD HH:MM:SS' 형식으로 반환한다.

예) `select now();` ==> '2001-05-07 09:10:10'

`select now() + 0;` ==> 20010507091010

- ▶ **CURDATE() 또는 CURRENT\_DATE() 현재 날짜 출력**

- ※ 함수의 상황이 숫자인지 문자열인지에 따라

YYYYMMDD 또는 'YYYY-MM-DD' 형식으로 반환한다.

예) `select curdate();` ==> '2001-05-07'

`select curdate() + 0;` ==> 20010507

- ▶ **CURTIME() 또는 CURRENT\_TIME() 현재 시간 출력**

- ※ 함수의 상황이 숫자인지 문자열인지에 따라 HHMMSS 또는 'HH:MM:SS' 형식

예) `select curtime();` ==> '09:10:10'

`select curtime() + 0;` ==> 091010

- ▶ **DATE\_ADD(날짜, INTERVAL 기준값) 날짜에서 기준값 만큼 더한다.**

- ※ 기준값 : YEAR, MONTH, DAY, HOUR, MINUTE, SECOND

예) `select date_add(now(), interval 2 day);` ==> 오늘보다 2일 후의 날짜와 시간

`select date_add(curdate(), interval 2 day);` ==> 오늘보다 2일 후의 날짜 출력.

# 날짜 관련 함수

▶ **DATE\_SUB(날짜, INTERVAL 기준값)** 날짜에서 기준값 만큼 뺀다.

※ 기준값 : YEAR, MONTH, DAY, HOUR, MINUTE, SECOND

`select date_sub(now(), interval 2 day);` ==> 오늘보다 2일 전의 날짜와 시간 출력.

`select date_sub(curdate(), interval 2 day);` ==> 오늘보다 2일 전의 날짜 출력.

▶ **YEAR(날짜)** : 날짜의 연도 출력.

`select year('20000101');`      `select year(20000101);`

`select year('2000-01-01');`      `select year(now());`

`select year(curdate());`      `select year(date_add(now(), interval 2 year));`

`select year(date_sub(curdate(), interval 2 year));`

▶ **MONTH(날짜)** : 날짜의 월 출력.

`select month('20001231');`      `select month(20001231);`

`select month('2000-12-31');`      `select month(now());`

`select month(curdate());`      `select month(date_add(now(), interval 2`

`month));`

`select month(date_sub(curdate(), interval 2 month));`

▶ **MONTHNAME(날짜)** : 날짜의 월을 영어로 출력.

`select monthname(20021221);`      `select monthname('20000721');`

`select monthname('2000-08-10');`      `select monthname(now());`

`select monthname(curdate());`

`select monthname(date_add(now(), interval 17 month));`

`select monthname(date_sub(curdate(), interval 11 month));`

# 날짜 관련 함수

## ▶ DAYNAME(날짜) : 날짜의 요일일 영어로 출력.

```
select dayname(20000121);      select dayname('20010123');
select dayname('2001-06-22');  select dayname(now());
select dayname(curdate());
select dayname(date_add(now(),interval 21 day));
select dayname(date_sub(curdate(),interval 333 day));
```

## ▶ DAYOFMONTH(날짜) : 날짜의 월별 일자 출력.

```
select dayofmonth(20030112);    select dayofmonth('20011231');
select dayofmonth('2001-12-23'); select dayofmonth(now());
select dayofmonth(curdate());
select dayofmonth(date_add(now(),interval 56 day));
select dayofmonth(date_sub(curdate(),interval 33 day));
```

## ▶ DAYOFWEEK(날짜) : 날짜의 주별 일자

출력(월요일(2),화요일(3)...일요일(1))

```
select dayofweek(20011209);      select dayofweek('20001212');
select dayofweek('2003-03-21');  select dayofweek(now());
select dayofweek(curdate());
select dayofweek(date_add(now(),interval 23 day));
select dayofweek(date_sub(curdate(),interval 31 day));
```

# 날짜 관련 함수

- ▶ **WEEKDAY(날짜)** : 날짜의 주별 일자 출력(월요일(0), 화요일(1)...일요일(6))  
select weekday(20000101);      select weekday('20030223');  
select weekday('2002-10-26'); select weekday(now());  
select weekday(curdate());      select weekday(date\_add(now(),interval 23 day));  
select weekday(date\_sub(curdate(),interval 33 day));
- ▶ **DAYOFYEAR(날짜)** : 일년을 기준으로 한 날짜까지의 날 수.  
select dayofyear(20020724);      select dayofyear('20001231');  
select dayofyear('2002-01-01'); select dayofyear(now());  
select dayofyear(curdate());      select  
dayofyear(date\_add(curdate(),interval 44 year));  
select dayofyear(date\_sub(now(),interval 25 month));  
select dayofyear(date\_add(now(),interval 55 day));  
select dayofyear(date\_sub(curdate(),interval 777 hour));  
select dayofyear(date\_add(now(),interval 999999 minute));
- ▶ **WEEK(날짜)** : 일년 중 몇 번째 주.  
select week(now());      select week(date\_sub(curdate(),interval 12 month));
- ▶ **FROM\_DAYS(날 수)**  
--00년 00월 00일부터 날 수 만큼 경과한 날의 날짜 출력.  
※ 날 수는 366 이상을 입력 그 이하는 무조건 '0000-00-00' 으로 출력.  
--또한 9999-12-31 [from\_days(3652424)] 까지의 날짜가 출력가능  
--따라서 날 수는 366 이상 3652424[3652499] 이하가 되어야 한다.  
select from\_days(3652424);      select from\_days('3652499');

# 날짜 관련 함수

## ▶ TO\_DAYS(날짜)

--00 년 00 월 00일 부터 날짜까지의 일자 수 출력. 정확한 날짜범위는 3652424 일 수

```
select to_days(now()) - to_days('1970-10-10');
```

응용 예제2) 살아 온 날수를 이용하여 자신의 나이를 만으로 구하기

```
select (to_days(now())-to_days('1970-10-10'))/365;
```

▶ DATE\_FORMAT(날짜, '형식') : select date\_format(now(), '%Y:%M:%p'); => 2001:May:PM

타입	기호	설명	기호	설명
년도	%Y	4자리 연도	%y	2자리 연도
월	%M	긴 월 이름 (January, ...)	%m	숫자의 월 (01...12)
	%b	짧은 월 이름 (Jan, ...)	%c	숫자의 월 (1...12)
요일	%W	긴 요일 이름 (Sunday, ...)	%a	짧은 요일 이름 (Sun, ...)
일	%D	월 내에서 서수 형식의 일(1th, ...)	%d	월 내의 일자 (01...31)
	%w	숫자의 요일 (0=Sunday, ...)	%e	월 내의 일자 (1...31)
			%j	일년 중의 날수 (001...366)
시	%I	12시간제의 시 (1...12)	%k	12시간제의 시 (0...23)
	%h	12시간제의 시 (01...12)		12시간제의 시 (00...23)
	%l	12시간제의 시 (01...12)		
분	%i	숫자의 분 (00...59)		
초	%S	숫자의 초 (00...59)	%s	숫자의 초 (00...59)
시간	%r	12시간제의 시간 (hh:mm:ss AM 또는 PM)	%T	24시간제의 시간 (hh:mm:ss)
주	%U	일요일을 기준으로 한 주 (0...52)	%u	월요일을 기준으로 한 주 (0...52)
기타	%%	문자 '%'	%p	AM 또는 PM



# 논리관련함수

## 5) 논리관련함수

- ▶ IF(논리식, 참일 때 값, 거짓일 때 값)

Select pno, pname, pay, if(pay >= 1500, ' good' , ' poor' as result from personal;

- ▶ IFNULL(값1, 값2)

값1이 NULL 이면 값2로 대체하고 그렇지 않으면 값1을 출력

select pno, pname, pay, pay + ifnull(bonus, 0) from personal;

## 6) 그밖의 함수

- ▶ limit select pname from personal limit 5;

- ▶ select DATABASE() : 현재의 데이터베이스 이름을 출력한다.

- ▶ mysql5.\* : select password('문자열') : 문자열을 암호화한다.

- ▶ mysql8.\* : select sha('a');

- ▶ FORMAT(숫자, 소수이하자리수) : 숫자를 #, ###, ###.## 형식으로 출력

--임의의 소수점자리수를 생성한다./소숫점을 필요한 만큼 취한다.

--소숫점을 만들어 같은 길이로 불러와서 소숫점을 버리고 출력하는 등에 응용

select format(123, 5);          select format(123.12345600123, 9);    select  
format(123.123, -3);

※ 소숫점이하자리수가 0 이나 음수값은 해당이 안됨

## 연습

업무가 clerk0이나 manager가 아닌 사원, 입사일이 1991년 이후이며 급여가 20000이사인  
사원

업무가 manager0이거나 sales, 사원의 이름 첫 글자가 A~S인 사원중에서 이른 순으로 정렬

# MySQL의 데이터 형식

## ❖ MySQL 내장 함수

### ■ 제어 흐름 함수

- 제어 흐름 함수는 프로그램의 흐름 제어하는 역할
- IF (수식, 참, 거짓)
  - 수식이 참 또는 거짓인지 결과에 따라서 2중 분기
- IFNULL(수식1, 수식2)
  - 수식1이 NULL이 아니면 수식1이 반환
  - 수식1이 NULL이면 수식2가 반환

```
SELECT CASE 10
      WHEN 1 THEN '일'
      WHEN 5 THEN '오'
      WHEN 10 THEN '십'
      ELSE '모름'
END;
```

## ❖ MySQL 내장 함수

### ■ 제어 흐름 함수

- NULLIF(수식1, 수식2)
  - 수식1과 수식2가 같으면 NULL을 반환
  - 다르면 수식1을 반환
- CASE ~ WHEN ~ ELSE ~ END
  - CASE는 내장 함수는 아니며 연산자Operator로 분류
  - 다중 분기에 사용

# MySQL의 데이터 형식

## ❖ MySQL 내장 함수

### ■ 문자열 함수

- TRIM(문자열), TRIM(방향 자를\_문자열 FROM 문자열)
  - **TRIM(문자열)은 문자열의 앞뒤 공백을 모두 없앴**
  - **TRIM(방향 자를\_문자열 FROM 문자열) 에서 방향은 LEADING(앞), BOTH(양쪽), TRAILING(뒤) 으로 표시**
- REPEAT(문자열, 횟수) : 문자열을 횟수만큼 반복
- REPLACE(문자열, 원래 문자열, 바꿀 문자열)
  - **문자열에서 원래 문자열을 찾아서 바꿀 문자열로 바꿈**
- REVERSE(문자열) : 문자열의 순서를 거꾸로 바꿈

### ■ 수학 함수

- ABS(숫자) : 숫자의 절댓값 계산
- ACOS(숫자), ASIN(숫자), ATAN(숫자), ATAN2(숫자1, 숫자2), SIN(숫자), COS(숫자), TAN(숫자) : 삼각 함수와 관련된 함수 제공
- CEILING(숫자), FLOOR(숫자), ROUND(숫자)
  - **올림, 내림, 반올림 계산**
- CONV(숫자, 원래 진수, 변환할 진수)
  - **숫자를 원래 진수에서 변환할 진수로 계산**



# MySQL의 데이터 형식

## ❖ MySQL 내장 함수

### ■ 시스템 정보 함수

- 시스템의 정보를 출력하는 함수 제공
- USER(), DATABASE()
  - 현재 사용자 및 현재 선택된 데이터베이스 출력
- FOUND\_ROWS()
  - 바로 앞의 SELECT문에서 조회된 행의 개수 구함
- VERSION()
  - 현재 MySQL의 버전
- SLEEP(초)
  - 쿼리의 실행을 잠깐 멈춤

# 테이블 조인

## 1) 개념

두 개 이상의 테이블에서 공동 컬럼을 이용하여 조회, parent child관계에서 사용  
Select pname, personal.dno, pay, dname from personal, division  
where personal.dno = division.dno;.

## 2) 카티션 곱

select pname, dno, dname from personal, division;

## 3) 내부조인

select 필드명,... from table1, table2 where 연결조건;

select pname, p.dno, pay, dname from personal p, division d where  
p.dno=d.dno;

## 4) 외부조인

select pname, p.dno, pay, dname from personal p left outer join  
division d on p.dno = d.dno;

select pname, p.dno, pay, dname from personal p right outer join  
division d on p.dno = d.dno;

## 5) 셀프조인

테이블 한 개를 다른 이름으로 join시켜 사용

select w.pno, w.pname, w.pay, m.pname from personal w, personal  
m where w.manager = m.pno;

# Sub Query

## 1) 서브쿼리

서브쿼리는 다른 SELECT 문의 검색 조건에 포함된 하나의 SELECT 문입니다. 내부 서브쿼리가 먼저 해결되고 쿼리에 대한 응답은 외부 SELECT 문으로 전달되는데, 이 외부 SELECT 문을 사용하여 마지막으로 최종 결과를 표시합니다

```
SELECT pname, MAX(pay) from personal;
```

원하는 결과를 이끌어낼 방법이 없는 특정한 경우에 서브쿼리를 이용할 수 있습니다. 회사에서 가장 높은 급여를 받는 사람을 알고 싶다면, 예제와 같은 SELECT 문을 쓸 수 있을 것입니다.

personal에서 가장 높은 급여를 받는 사람의 이름을 알아보려고 합니다. 그런데 이 SELECT 문에서 무언가 잘못되었음을 알게 됩니다. 세부 컬럼과 집계 함수를 혼합한 것입니다. 이 SELECT 문이 작동하도록 하려면 GROUP BY 절이 필요합니다.

```
SELECT pname MAX(pay) FROM personal GROUP BY pname;
```

GROUP BY 절을 추가하자 SELECT 문이 작동합니다. 그러나 여전히 찾으려는 답을 얻지는 못한 상태입니다. 간단한 기본 쿼리 구조를 사용하여 올바른 답을 찾는 유일한 방법은 두 개의 쿼리를 코딩하는 것입니다. 첫 번째 쿼리에서는 가장 높은 급여 값을 얻습니다. 다음 두 번째 쿼리에 이 값을 사용하여 이 급여를 받는 사람의 이름을 찾을 수 있습니다.

```
SELECT pname, pay from personal WHERE SALARY = (SELECT MAX(pay) FROM personal);
```

원하는 답을 얻는 좀더 간단한 방법은 서브쿼리를 사용하여 가장 높은 급여액을 찾고, 그 금액을 넘겨주어 외부 쿼리를 해결하는 데에 사용하는 것입니다.

여기서 몇 개의 구문 규칙에 유의하십시오. 서브쿼리는 WHERE 절 안의 조건 오른쪽에 있으며 괄호로 묶여 있습니다. 서브쿼리의 SELECT 절에서는 단 하나의 컬럼만 지정할 수 있습니다. 또한 서브쿼리의 결과는 표시되지 않고 외부 쿼리로 전달된다는 점에 유의하십시오.

# Query

**SELECT NAME, SALARY FROM STAFF WHERE SALARY >= (SELECT AVG(SALARY) FROM STAFF)**

외부 쿼리로 전달되는 값은 단 하나뿐 : =, <, <=, > 또는 >= 연산자 사용

사원번호가 1121인 사원보다 많은 급여를 받는 사람

**select \* from personal where pay > (select pay from personal where pno=1121);**

martine과 같은 업무에 종사하는 사람, 단 마틴은 제외

**select \* from personal where pname != ' martine' and job = (select job from personal where pame=' martine' );**

sales부서에 근무하는 사람

**select \* from personal where dno=(select dno from division where dname=' sales' );**

평균급여보다 많이 받는 사람

**select \* from personal where pay > (select avg(pay) from personal);**

2) IN 을 이용한 서브 쿼리

**select \* from personal where dno in (select dno from division where position=' seoul' );**

서브쿼리는 여러 부서 번호를 반환합니다. 연산자는 IN으로 바뀌어 다중 값을 수용해야 합니다. 그렇지 않으면 오류를 발생하게 됩니다. (IN을 대체할 수 있는 것 = ANY, some)

# Query

## 3) 여러 개의 값을 이용하는 서브 쿼리 : ALL

**SELECT AVG(pay) FROM personal GROUP BY dno**

모든 부서의 부서별 평균 급여보다 높은 급여를 받는 모든 직원의 이름을 목록으로 만들려고 합니다. 우선, 각 부서 별로 직원의 평균 급여를 찾아야 합니다. 그렇게 하려면 GROUP BY 절을 사용해야 합니다. 외부 쿼리에 사용할 다수의 결과를 얻습니다.

서브쿼리의 결과와 각 직원의 급여를 비교하십시오. 구하는 결과는 부서의 평균 급여보다 높은 급여입니다. 이런 결과를 얻으려면 서브쿼리 전에 '> ALL' 을 사용합니다.

**Select \* from personal where pay > all (select avg(pay) from personal group by dno);**

### 4) 서브쿼리 예

어떤 부서의 평균급여보다 높은 급여를 받는 직원

같은 부서의 평균급여보다 많이 받는 사람

급여가 가장 적은 사람

**select \* from personal where pay= (select min(pay) from personal);**

입사일이 가장 빠른 사람

**select \* from personal where startdate = (select min(startdate) from personal;**

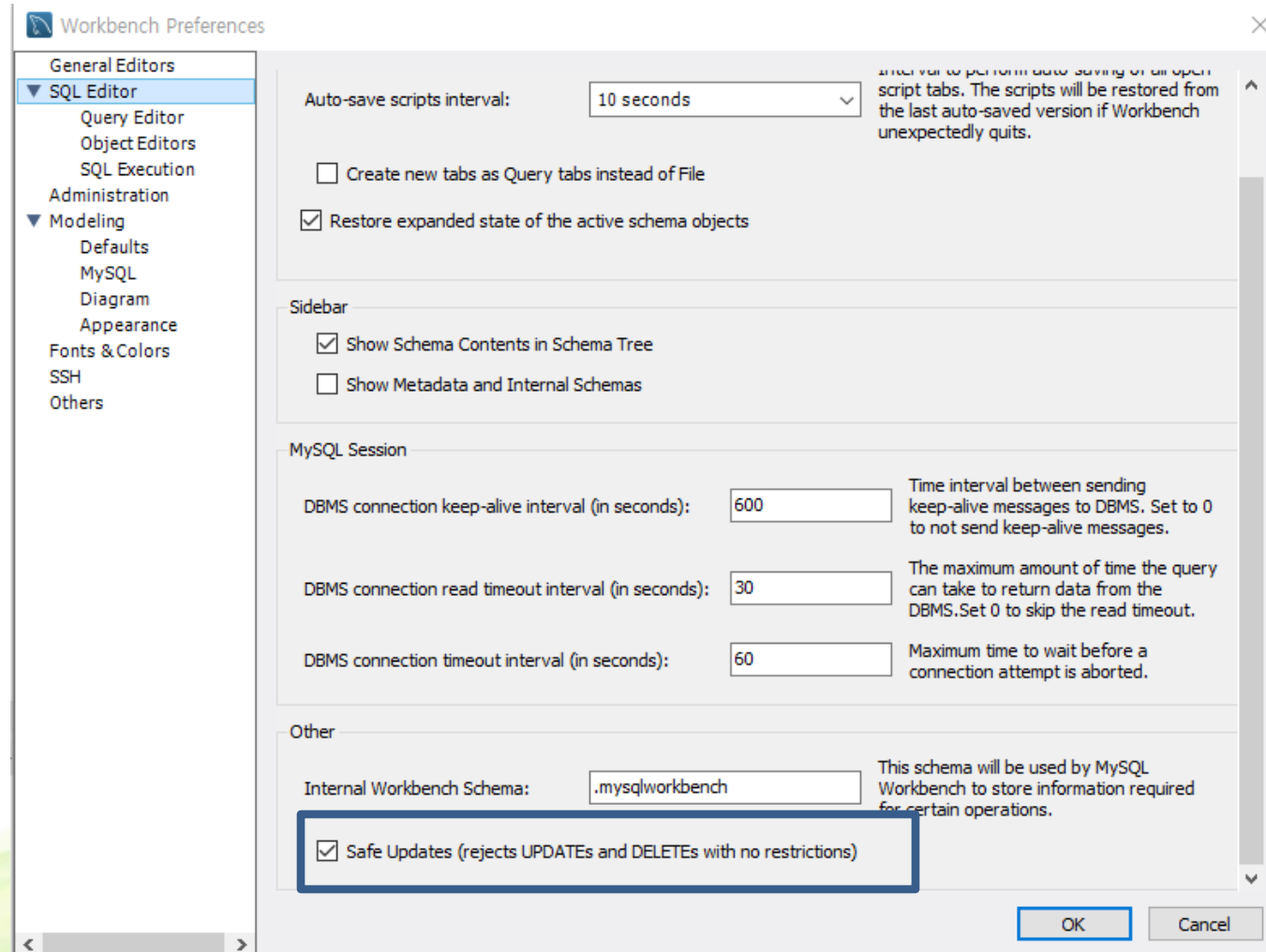
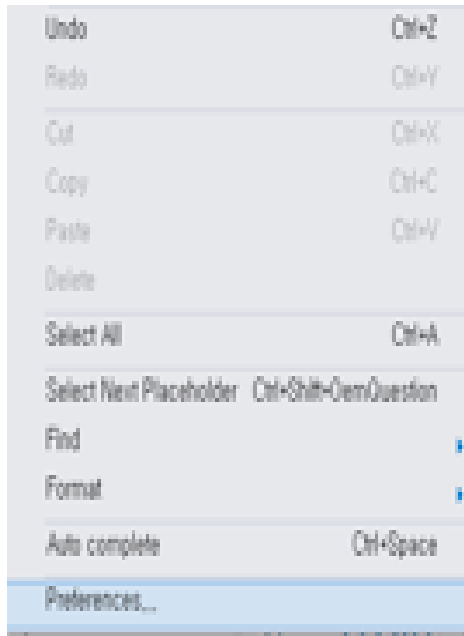
각 부서의 평균 급여중에서 가장 많은 급여를 받는 부서의 번호와 급여 출력

**select dno, avg(pay) from personal group by dno having avg(pay) >= all (select avg(pay) from personal group by dno);**

# Update 또는 delete에 제약 있을 때

edit – preferences

Safe Updates 체크 해제 후에 workbench 다시 시작



# 기타 SQL

업데이트 안될 때 : edit - preference - sqlediter 하단 safe update 체크,해제

## 1) 서브쿼리 활용

- . create table salesman as select \* from personal where job= 'salesman' ;
- . create table sawon as select pno, pname, pay, dno from personal;
- . insert into sawon2 select pno, pname, pay from personal where dno=20;
- . update personal set job=' salesman' where dno = (select dno from division where dname= 'sales' );
- . delete from sawon where dno = (select dno from division where position = 'seoul' );

## 2)TCL(transaction Control Language)

- . Transaction : DML명령을 하나의 처리단위로 만들어 놓은 것
- . Transaction의 시작 : DML명령사용
- . Transaction 적용 : Commit
- . Transaction 취소 : rollback

자동 commit

- . DML, DCL 명령사용, . SQL 정상종료

자동 Rollback

- . SQL 비정상 종료



# Data BACKUP 및 복구(mysql5.\*)

set global local\_infile=1;

## 1) DataBase

]# mysqldump -u 계정 -h 서버 -p DB명 > 파일명

]# mysql -u 계정 -h 서버 -p DB명 < 파일명

## 2) Table

mysqldump -u root -pmysql test personal > a.txt

mysql -u root -pmysql test < a.txt

## 3) load

mysql> load data infile '파일네임' replace into table personal;

load data infile '파일네임' into table personal;

load data infile 'c:/gov/mysql/sawon.csv' into table sawon

CHARACTER SET euckr fields terminated by ',' lines terminated by '\r\n';

SHOW VARIABLES LIKE 'char%';

## 3) DB내용

/usr/local/mysql/var : db가 디렉토리별로 되어 있음

## 4) SQL계정삭제

mysql> delete from user where user = '계정' ;

권한 재적용 mysql db에서 권한 재적용할 때에 사용하는 명령어



# Data BACKUP및 복구(mysql8.\*)

## 1. 데이터 위치 보기

`show variables like "secure_file_priv";`

## 2. 데이터를 해당 위치로 변경

`C:\ProgramData\MySQL\MySQL Server 8.0\Uploads`

## 3. Load명령(tab)

`load data [local] infile`

`'C:/ProgramData/MySQL/MySQL Server  
8.0/Uploads/emp.txt'`

`into table emp character set euckr;`

## 4. Load명령(csv)

`load data [local] infile`

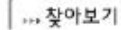
`'C:/ProgramData/MySQL/MySQL Server  
8.0/Uploads/emp2.csv'`

`into table emp2 character set euckr fields terminated by ','  
lines terminated by '\r\n';`

# 인덱스

## ❖인덱스 (Index) 란?

바로 찾아진다

책의 제일 뒷부분

# 인덱스의 개념

## ❖인덱스의 장단점

### ■ 장점

- 검색 속도가 무척 빨라질 수 있음 (단, 항상 그런 것은 아님)
- 그 결과 해당 쿼리의 부하가 줄어들어, 결국 시스템 전체의 성능 향상

### ■ 단점

- 인덱스가 데이터베이스 공간 차지해서 추가적인 공간 필요
  - **대략 데이터베이스 크기의 10% 정도의 추가 공간 필요**
- 처음 인덱스 생성하는데 시간 많이 소요 가능
- 데이터의 변경 작업 (Insert, Update, Delete)이 자주 일어날 경우
  - **오히려 성능이 많이 나빠질 수도 있음**

## ❖인덱스의 종류

### ■ 클러스터형 인덱스 (Clustered Index)

- ‘영어 사전’과 같은 책
- 테이블 당 한 개만 지정 가능
- 행 데이터를 인덱스로 지정한 열에 맞춰 자동 정렬

### ■ 보조 인덱스 (Secondary Index)

- 책 뒤에 <찾아보기>가 있는 일반 책
- 테이블당 여러 개도 생성 가능

# 인덱스의 종류와 자동 생성

## ❖ 자동으로 생성되는 인덱스

### ■ Primary Key 지정

- 자동으로 userID 열에 클러스터형 인덱스 생성
- 테이블 생성 시 제약 조건으로 Primary Key 또는 Unique 사용
  - 인덱스 자동 생성

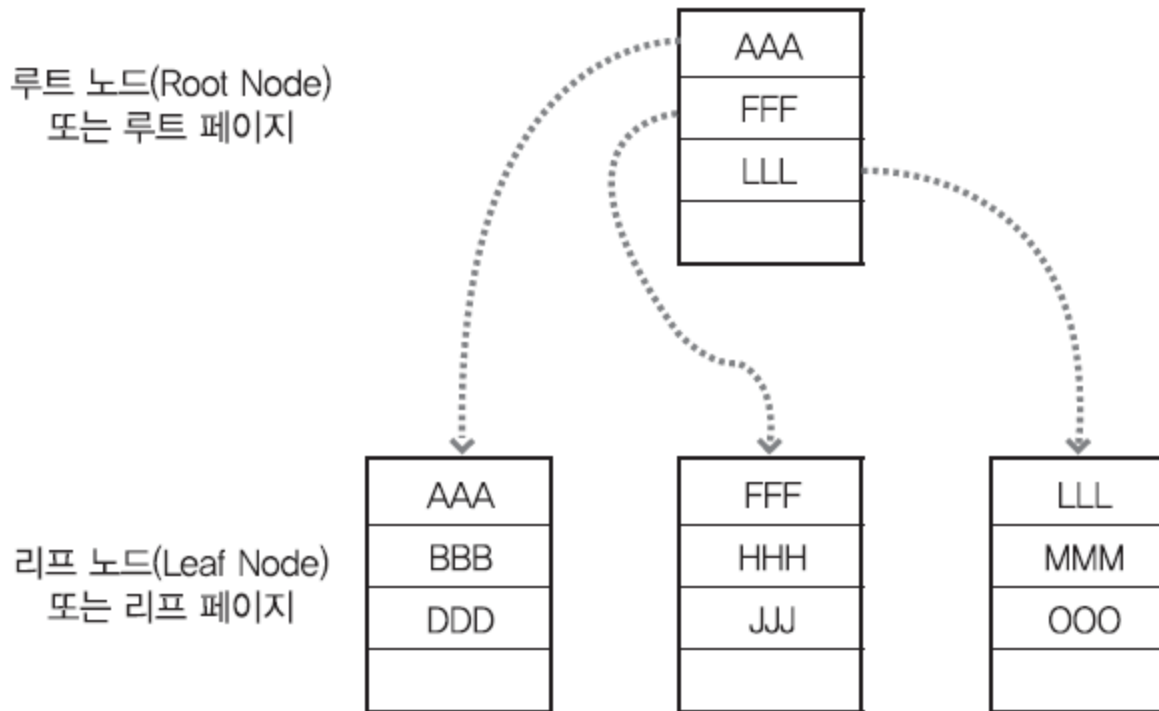
## ❖ 자동으로 생성되는 인덱스

- PRIMARY KEY로 지정한 열은 클러스터형 인덱스 생성
- UNIQUE NOT NULL로 지정한 열은 클러스터형 인덱스 생성
- UNIQUE(또는 UNIQUE NULL)로 지정한 열은 보조 인덱스 생성
- PRIMARY KEY와 UNIQUE NOT NULL이 존재
  - **PRIMARY KEY로 지정한 열에 우선 클러스터형 인덱스 생성**
- PRIMARY KEY로 지정한 열로 데이터가 오름차순 정렬

# 인덱스의 내부 작동

## ❖ B-Tree(Balanced Tree, 균형 트리)

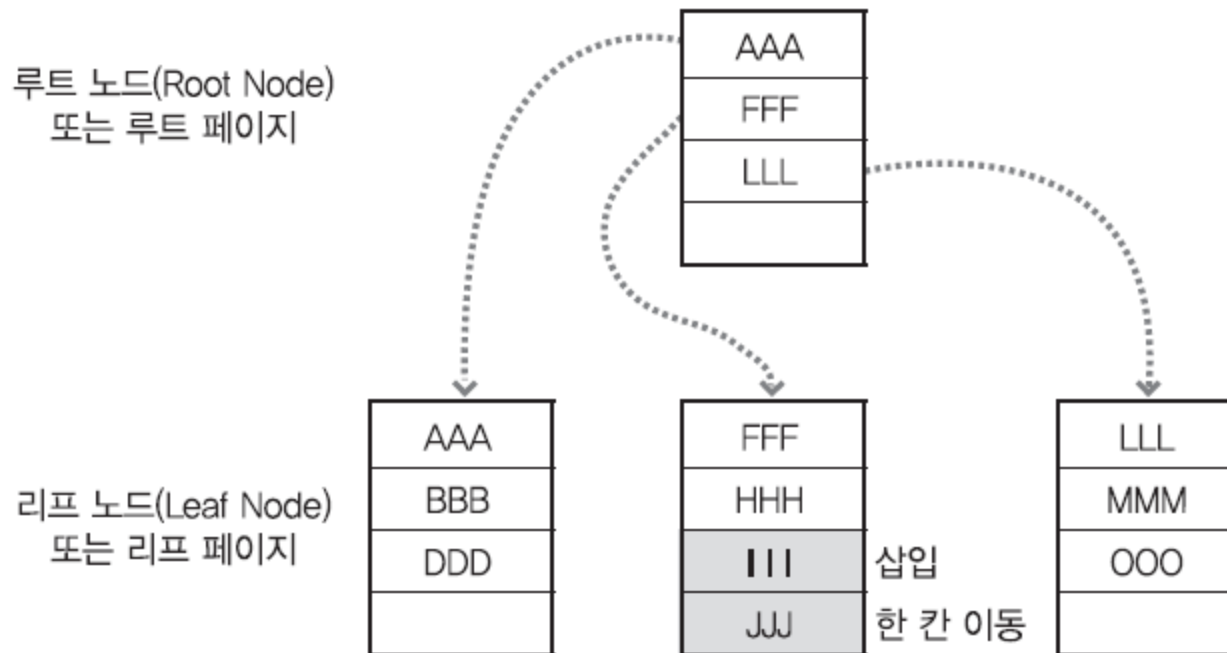
- ‘자료 구조’ 에 나오는 범용적으로 사용되는 데이터 구조
- 인덱스 표현할 때 편리하게 사용
- 균형이 잡힌 트리



# 인덱스의 내부 작동

## ❖ 페이지 분할

- **SELECT 문의 효율성 향상**
- **INSERT 문이 일어날 경우 속도 저하되는 단점**
  - 주어진 공간 이상으로 데이터 들어가면 페이지 분할 일어남





# 인덱스의 내부 작동

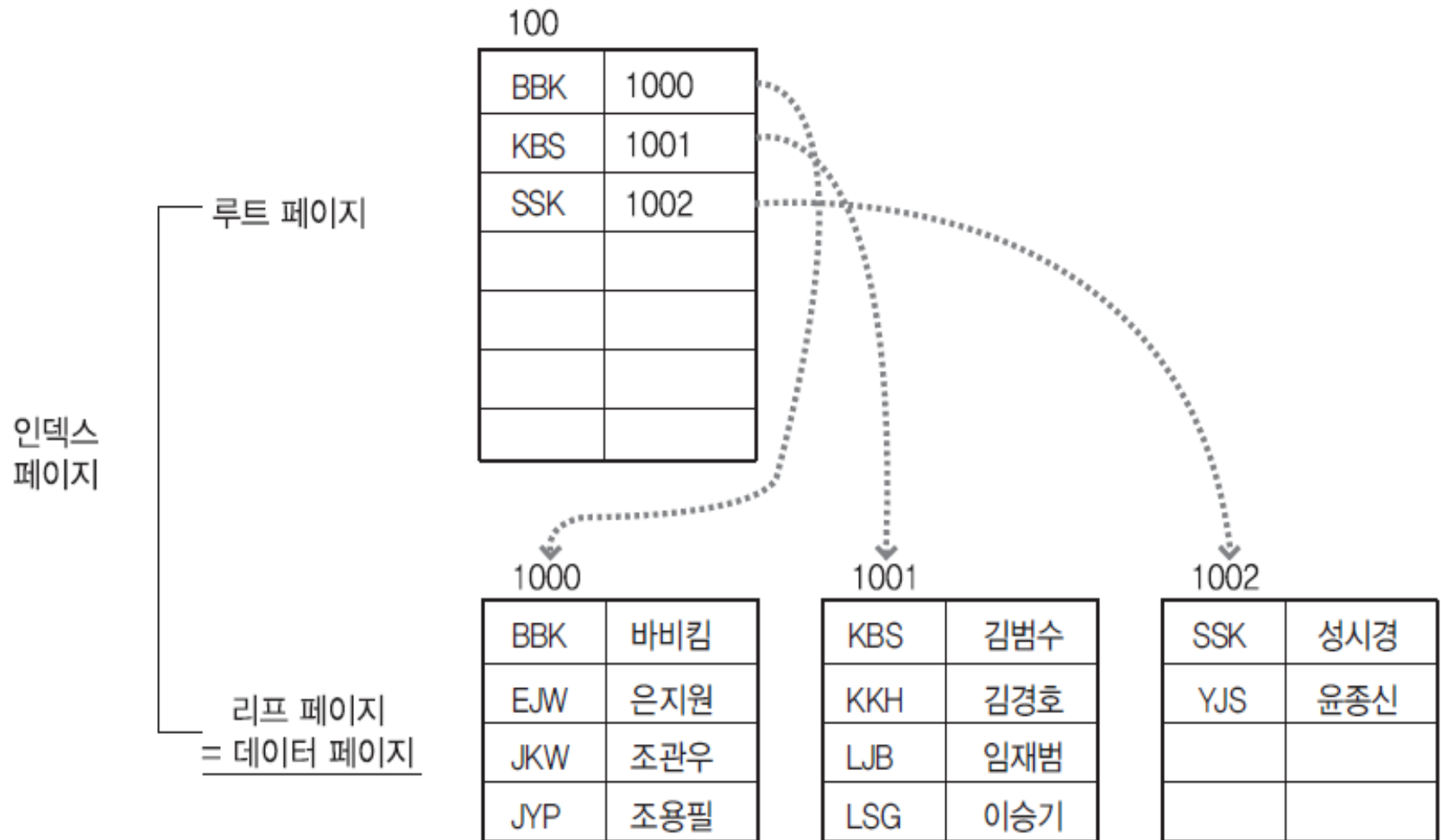
- ❖ 클러스터형 인덱스와 보조 인덱스의 구조
  - 인덱스 없는 테이블의 예시

데이터 페이지  
(Heap 영역)

1000	1001	1002
LSG 이승기	SSK 성시경	JKW 조관우
KBS 김범수	LJB 임재범	BBK 바비킴
KKH 김경호	YJS 윤종신	
JYP 조용필	EW 은지원	

# 인덱스의 내부 작동

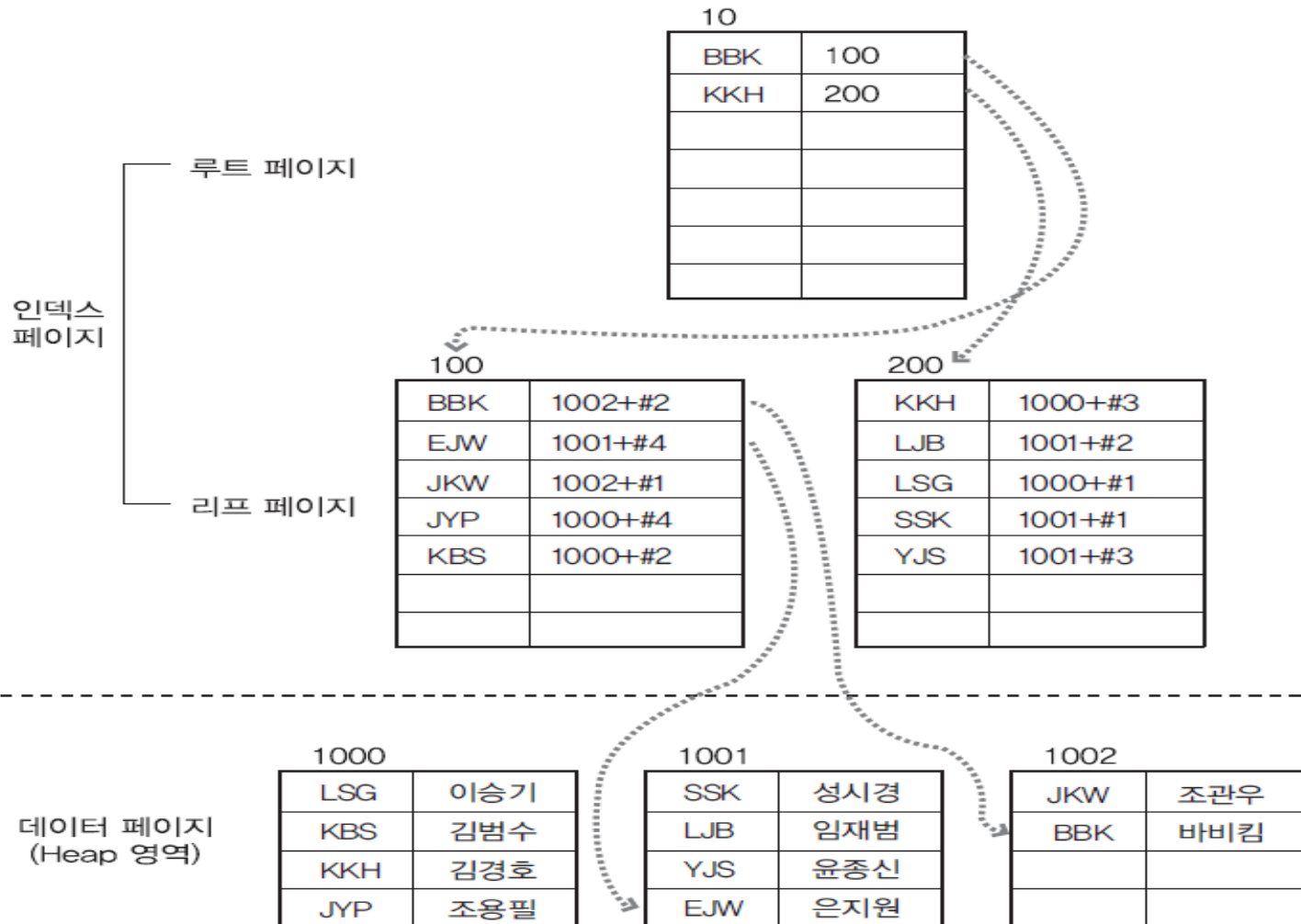
- ❖ 클러스터형 인덱스와 보조 인덱스의 구조
  - 클러스터형 인덱스 구성한 테이블 구조



# 인덱스의 내부 작동

## ❖ 클러스터형 인덱스와 보조 인덱스의 구조

### ■ 보조 인덱스 구성한 테이블 구조



# 인덱스의 내부 작동

## ❖ 클러스터형 인덱스와 보조 인덱스의 구조

### ■ 클러스터형 인덱스의 특징

- 클러스터형 인덱스의 생성 시에는 데이터 페이지 전체 다시 정렬
  - 이미 대용량의 데이터가 입력된 상태라면 업무시간에 클러스터형 인덱스 생성하는 것은 심각한 시스템 부하
- 인덱스 자체의 리프 페이지가 곧 데이터 : 인덱스 자체에 데이터가 포함
- 클러스터형 인덱스는 보조 인덱스보다 검색 속도는 더 빠름
  - 데이터의 입력/수정/삭제는 더 느림
- 클러스터형 인덱스는 성능이 좋지만 테이블에 한 개만 생성 가능
  - 어느 열에 클러스터형 인덱스 생성하는지에 따라 시스템의 성능이 달라짐

## ❖ 클러스터형 인덱스와 보조 인덱스의 구조

### ■ 보조 인덱스의 특징

- 보조 인덱스 생성
  - 데이터 페이지는 그냥 둔 상태에서 별도의 페이지에 인덱스 구성
- 인덱스 자체의 리프 페이지는 데이터가 위치하는 주소 값 (RID)
- 클러스터형보다 검색 속도는 더 느림 : 데이터의 입력/수정/삭제는 덜 느림
- 보조 인덱스는 여러 개 생성할 수 있음
  - 남용할 경우에는 오히려 시스템 성능을 떨어뜨리는 결과

# 인덱스의 내부 작동

## ❖ 클러스터형 인덱스와 보조 인덱스의 구조

### ■ 클러스터형 인덱스와 보조 인덱스가 혼합되어 있을 경우

- 보조 인덱스와 혼합되어 사용되는 경우 되도록이면 클러스터형 인덱스로 설정할 열은 적은 자릿수의 열을 선택하는 것이 바람직함
- 인덱스를 검색하기 위한 일차 조건
  - WHERE절에 해당 인덱스를 생성한 열의 이름이 나와야 함
  - WHERE절에 해당 인덱스를 생성한 열 이름이 나와도 인덱스를 사용하지 않는 경우도 많으므로 조건을 잘 따져 볼 것

# 인덱스 생성/변경/삭제

## ❖ 인덱스 생성

형식:

```
CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name  
    [index_type]  
    ON tbl_name (index_col_name,...)  
    [index_option]  
    [algorithm_option ; lock_option] ...
```

index\_col\_name:

```
col_name [(length)] [ASC ; DESC]
```

index\_type:

```
USING {BTREE ; HASH}
```

index\_option:

```
KEY_BLOCK_SIZE [=] value  
; index_type  
; WITH PARSER parser_name  
; COMMENT 'string'
```

algorithm\_option:

```
ALGORITHM [=] {DEFAULT|INPLACE|COPY}
```

lock\_option:

```
LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
```

인덱스 제거 : 간단히 인덱스 삭제하는 형식  
DROP INDEX 인덱스이름 ON 테이블이름;

# 인덱스를 생성해야 하는 경우

## ❖인덱스에 대한 결론

- **인덱스는 열 단위에 생성**
  - 두 개 이상의 열을 조합해서 인덱스 생성 가능
- **WHERE절에서 사용되는 열에 인덱스를 만들어야 함**
  - 테이블 조회 시 WHERE절의 조건에 해당 열이 나오는 경우에만 인덱스 주로 사용
- **WHERE절에 사용되더라도 자주 사용해야 가치가 있음**
  - SELECT문이 자주 사용 되어야 효과적
  - INSERT 문이 자주 사용되거나 자주 사용되는 열에 생성된 인덱스가 클러스터형이면 효율 감소

## ❖인덱스에 대한 결론

- **데이터의 중복도가 높은 열은 인덱스 만들어도 효과 없음**
  - 인덱스의 관리에 대한 비용 높은 경우 쓸모 없음
- **외래 키 지정한 열의 경우**
  - 자동으로 외래 키 인덱스 생성
- **JOIN에 자주 사용되는 열의 경우**
  - 인덱스를 생성해 주는 것이 좋음



# 인덱스를 생성해야 하는 경우

## ❖ 인덱스에 대한 결론

- **INSERT/UPDATE/DELETE가 얼마나 자주 일어나는지 고려해야 함**
  - 인덱스는 단지 읽기에서만 성능 향상
  - 데이터의 변경에서는 오히려 부담
- **클러스터형 인덱스는 테이블당 하나만 생성 가능**
  - 클러스터형 인덱스를 생성할 열은 범위(BETWEEN, >, < 등의 조건)로 사용하거나 집계 함수를 사용하는 경우 아주 적절하게 사용
  - 테이블에 아예 없는 것이 좋은 경우도 있음
- **사용하지 않는 인덱스는 제거**
  - 공간 확보할 뿐 아니라 데이터의 입력 시에 발생하는 부하도 많이 줄일 수 있음

# 스토어드 프로시저

# 스토어드 프로시저

## ❖ 스토어드 프로시저의 개요

### ■ 스토어드 프로시저(Stored Procedure)

- 저장 프로시저라고도 불림
- MySQL에서 제공되는 프로그래밍 기능
- 쿼리문의 집합으로 어떠한 동작을 일괄 처리하기 위한 용도로 사용
- 쿼리 모듈화
  - 필요할 때마다 호출만 하면 훨씬 편리하게 MySQL 운영
  - **CALL 프로시저\_이름( )** 한 줄로 해결되는 편리함

# 스토어드 프로시저

## ❖ 스토어드 프로시저의 개요

### ■ 기본 형식

형식 :

```
DELIMITER $$
```

```
CREATE PROCEDURE 스토어드 프로시저이름( IN 또는 OUT 파라미터 )
```

```
BEGIN
```

이 부분에 SQL 프로그래밍 코딩..

```
END $$
```

```
DELIMITER ;
```

```
CALL 스토어드 프로시저이름();
```

# 스토어드 프로시저

```
DROP PROCEDURE IF EXISTS division_insert;
delimiter //
create procedure division_insert (
    vjno int, vname varchar(20),
    vphone varchar(20), vposition varchar(20))
begin
    insert into division
        values(vjno,vname,vphone,vposition);
end;
//
delimiter ;
실행
sql> call division_insert(50,'대 박','010-1111-1111','서울');
```

# 스토어드 프로시저

```
drop procedure if exists emp_insert;
```

```
delimiter //
```

```
create procedure emp_insert
```

```
    (vempno int, vename varchar(20), vjob varchar(20),  
     vsal float(7,2), vdeptno int(2))
```

```
begin
```

```
    insert into emp (empno,ename,job,hireddate,sal,deptno)  
    values (vempno,vename,vjob,now(),vsal,vdeptno);
```

```
end //
```

```
delimiter ;
```

```
call emp_insert(1234,'철수','코파기',2000,10);
```

# 스토어드 프로시저

## ❖ 스토어드 프로시저의 개요

### ■ 스토어드 프로시저의 수정과 삭제

- 수정 : ALTER PROCEDURE
- 삭제 : DROP PROCEDURE

### ■ 매개 변수의 사용

- 입력 매개 변수를 지정하는 형식
  - **IN 입력\_매개 변수\_이름 데이터\_형식**
- 입력 매개 변수가 있는 스토어드 프로시저 실행 방법
  - **CALL 프로시저\_이름(전달 값);**
- 출력 매개 변수 지정 방법
  - **OUT 출력\_매개 변수\_이름 데이터\_형식**
  - 출력 매개 변수에 값 대입하기 위해 주로 **SELECT... INTO**문 사용



# 스토어드 프로시저

## ❖ 스토어드 프로시저의 개요

### ■ 프로그래밍 기능

- 더 강력하고 유연한 기능 포함하는 스토어드 프로시저 생성

### ■ 스토어드 프로시저 내의 오류 처리

- 스토어드 프로시저 내부에서 오류가 발생했을 경우

– **DECLARE 액션 HANDLER FOR 오류조건 처리할\_문장 구문**

# 스토어드 프로시저

## ❖ 스토어드 프로시저의 특징

### ■ MySQL의 성능 향상

- 긴 쿼리가 아니라 짧은 프로시저 내용만 클라이언트에서 서버로 전송
  - 네트워크 부하 줄임 → MySQL의 성능 향상

### ■ 유지관리가 간편

- 응용 프로그램에서는 프로시저만 호출
  - 데이터베이스 단에서 관련된 스토어드 프로시저의 내용 일관되게 수정/유지보수

### ■ 모듈식 프로그래밍 가능

- 언제든지 실행 가능 + 편리한 관리
- 모듈식 프로그래밍 언어와 동일한 장점

### ■ 보안 강화에 편리

- 스토어드 프로시저에만 접근 권한을 주어 DB가 안전해짐

# 스토어드 함수

## ❖ 스토어드 함수 (Stored Function)

- 사용자가 직접 만들어서 사용하는 함수
- 스토어드 프로시저와 상당히 유사
  - 형태와 사용 용도에 있어 차이 있음
- 스토어드 함수의 개요

```
DELIMITER $$  
CREATE FUNCTION 스토어드 함수이름( 파라미터 )  
    RETURNS 반환형식  
BEGIN  
  
    이 부분에 프로그래밍 코딩..  
    RETURN 반환값;  
  
END $$  
DELIMITER ;  
SELECT 스토어드_함수이름();
```

# 스토어드 함수

## ❖ 스토어드 함수 와 스토어드 프로시저의 차이점

### ■ 스토어드 함수

- 파라미터에 IN, OUT 등을 사용할 수 없음
  - 모두 입력 파라미터로 사용
- RETURNS문으로 반환할 값의 데이터 형식 지정
  - 본문 안에서는 RETURN문으로 하나의 값 반환
- SELECT 문장 안에서 호출
- 안에서 집합 결과 반환하는 SELECT 사용 불가
  - **SELECT... INTO... 는 집합 결과 반환하는 것이 아니므로 예외적으로 스토어드 함수에서 사용 가능**
- 어떤 계산 통해서 하나의 값 반환하는데 주로 사용

# 스토어드 함수

1은 function 사용, 0은 사용안함

```
SET GLOBAL log_bin_trust_function_creators = 1;
```

```
use test;
```

```
drop function if exists getAgefun;
```

```
delimiter //
```

```
create function getAgeFun(bYear int)
```

```
returns int
```

```
begin
```

```
    declare age int;
```

```
    set age = year(current_date()) - bYear;
```

```
    return age;
```

```
end //
```

```
delimiter ;
```

```
select getAgeFun(1979);
```

# 스토어드 함수

## ❖ 스토어드 함수 실습

- 저장해 놓았던 sqlDB.sql 이용해 sqlDB 데이터베이스 초기화
- 출생년도 입력하면 나이 출력되는 함수 작성
  - 테이블을 조회할 때 주로 사용되는 함수
- 현재 저장된 스토어드 함수의 이름 및 내용 확인
  - **SHOW CREATE FUNCTION getAgeFunc;**
- 스토어드 함수 삭제
  - **DROP FUNCTION getAgeFunc;**

# 스토어드 함수

```
-- 사번, 이름, 입사일, 입사기간  
-- function during_day  
delimiter //
```

```
create function during_day (v_hiredate date) returns int  
begin
```

```
    declare dur int;
```

```
    set dur = to_days(now()) - to_days(v_hiredate);
```

```
    return dur;
```

```
end //
```

```
delimiter ;
```

```
select empno, ename, hiredate,  
       during_day(hiredate) '근무기간' from emp;
```



# trigger

```
drop table sawon;
create table sawon (empno int(4) not null primary key, ename
varchar(20),tel varchar(30));
drop table sal_t;
create table sal_t (
    sal_id int not null primary key auto_increment,
    empno int(4) references sawon(empno), sal int(10) default 0);
delimiter //
create trigger insert_sal
    after insert on sawon for each row
begin
    insert into sal_t (empno) values (new.empno);
end //
delimiter ;
insert into sawon values (1111,'철수','010-1111-1111');
insert into sawon values (1112,'길동','010-1234-1234');
select * from sawon;select * from sal_t;
```

# trigger

```
create table product (pno varchar(10) not null primary key, pname  
varchar(20), stock int default 0);  
create table warehouse (  
    num int not null primary key auto_increment,  
    pno varchar(10) references product(pno), qty int);  
delimiter //  
create trigger wh_insert  
after insert on warehouse for each row  
begin  update product set stock = stock + new.qty  
    where pno=new.pno;  
end //  
delimiter ;  
insert into product values('a1','사과',0);  
insert into product values('b1','바나나',0);  
insert into product values('c1','귤',0);  
insert into warehouse(pno,qty) values ('a1',11);  
insert into warehouse(pno,qty) values ('a1',3);  
insert into warehouse(pno,qty) values ('b1',11);  
insert into warehouse(pno,qty) values ('c1',3);  
elect * from product;  
select * from warehouse;
```

# trigger

**delimiter //**

**create trigger wh\_delete**

**after delete on warehouse for each row**

**begin**

**update product set**

**stock = stock - old.qty where pno=old.pno;**

**end //**

**delimiter ;**

**delete from warehouse where num=1;**

# trigger

```
delimiter //  
create trigger wh_update  
  after update on warehouse for each row  
Begin  
  update product  
    set stock = stock - old.qty + new.qty  
    where pno=old.pno;  
end //  
delimiter ;  
  
select * from warehouse;  
update warehouse set qty = 7 where num = 4;  
select * from product;
```