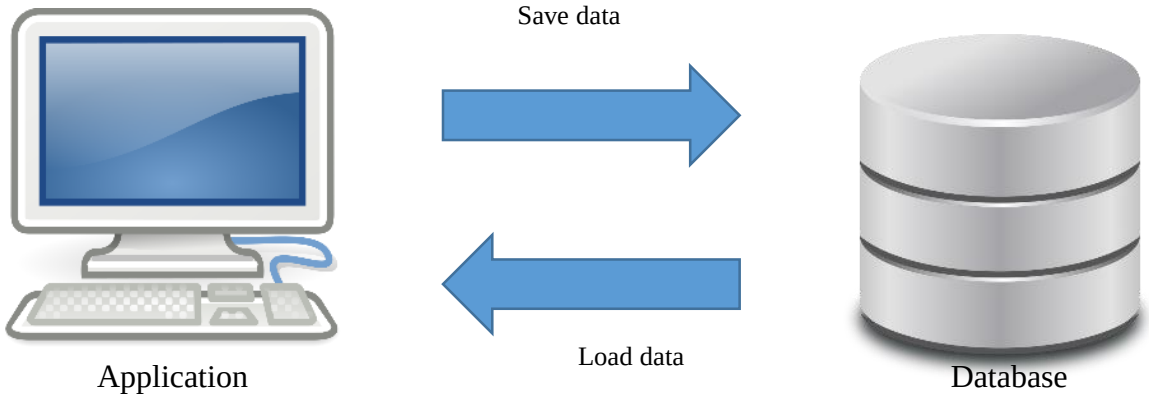
(Ch5-X) Python DBMS Binding

Table of Contents

DBMS and Application

- Database Application:
  - 일반적으로 주메모리(main memory) 상에서 실행
  - 프로그램이 종료 때 메모리 상에 저장되어 있던 관련 데이터들이 사라짐
  - 따라서 프로그램 수행 도중 생성된 데이터를 영구적으로 보존하려면..
    - **파일 시스템(file system)**: 파일의 형태로 데이터를 저장
    - **데이터베이스(database)**: 데이터베이스에 데이터를 저장

- 규모가 큰 어플리케이션의 경우, 데이터의 양이 많으며 구조도 복잡할 것이므로 데이터베이스를 사용하여 체계적으로 데이터를 관리하는 것이 합당!



DBMS와 Application의 연동

- 대부분의 경우, Application과 DBMS의 연동 기능을 구현할 수 있도록 **커넥터(connector 혹은 middleware 혹은 driver)**가 제공됨
  - API 혹은 Library의 형태로 제공
  - APP의 요청을 DBMS가 받아 실행한 결과를 APP이 받아볼 수 있도록 연결
- 예시: ODBC(Open Database Connectivity)

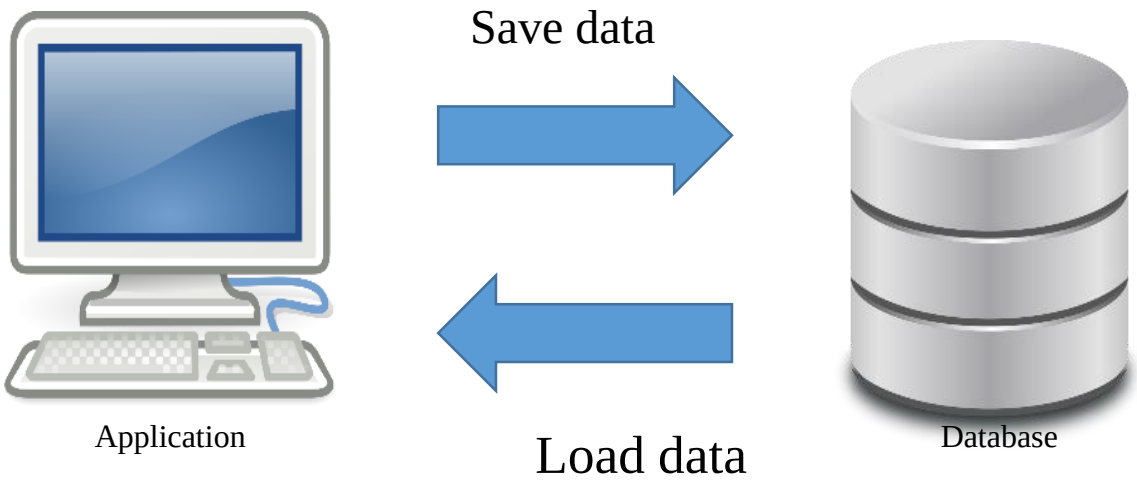
  JDBC(Java Database Connectivity)

  Python – MySQL connector: MySQLdb, PyMySQL,….

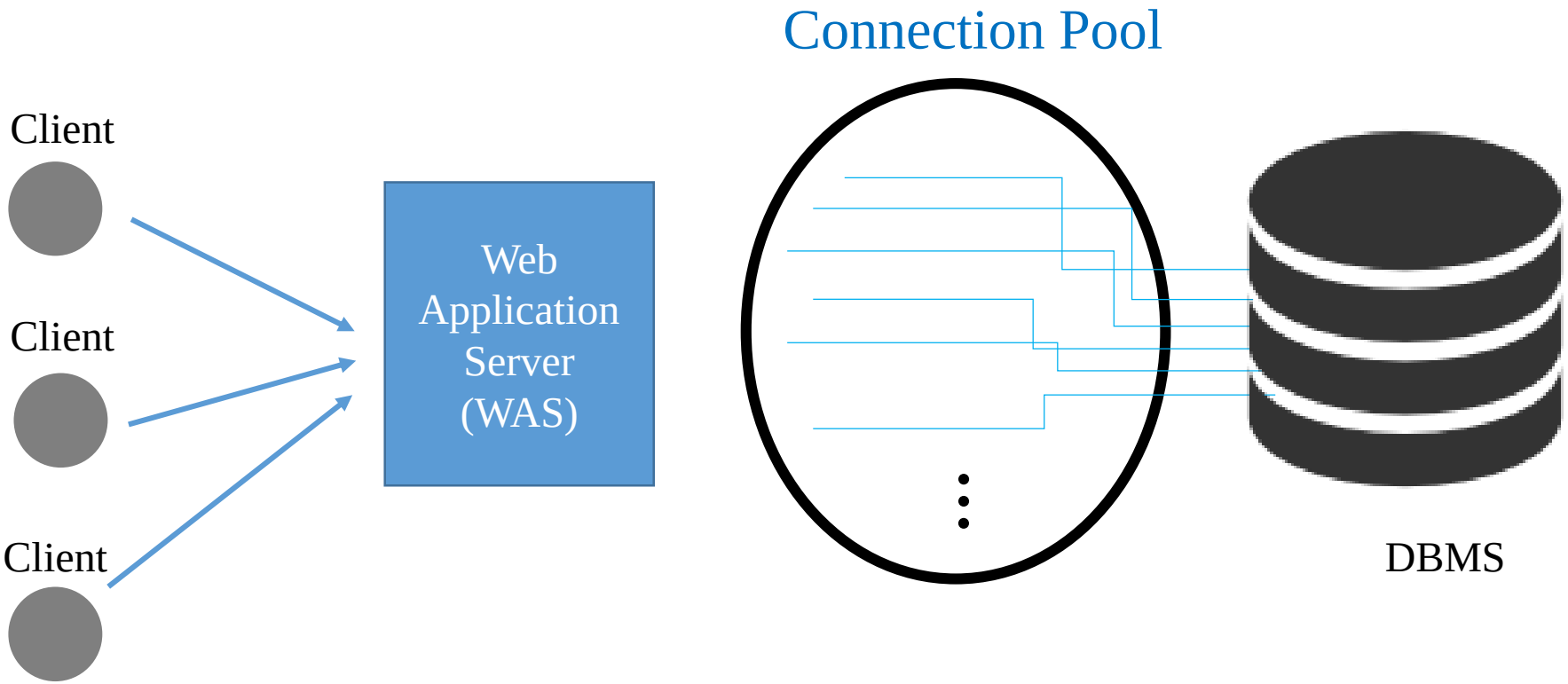
- DBMS – Application 연동법
  1. 활용하고자 하는 DBMS를 설치
  2. APP 개발에 사용하는 PL와 DB를 연결하는 connector 준비
  3. Connector를 활용하여 DBMS와 연동가능한 APP 을 PL로 coding

Model for Connecting Application to DBMS

- Open a connection
- Create a "statement" object (string)
- Execute queries using the Statement object to send queries and fetch results
- Exception mechanism to handle errors

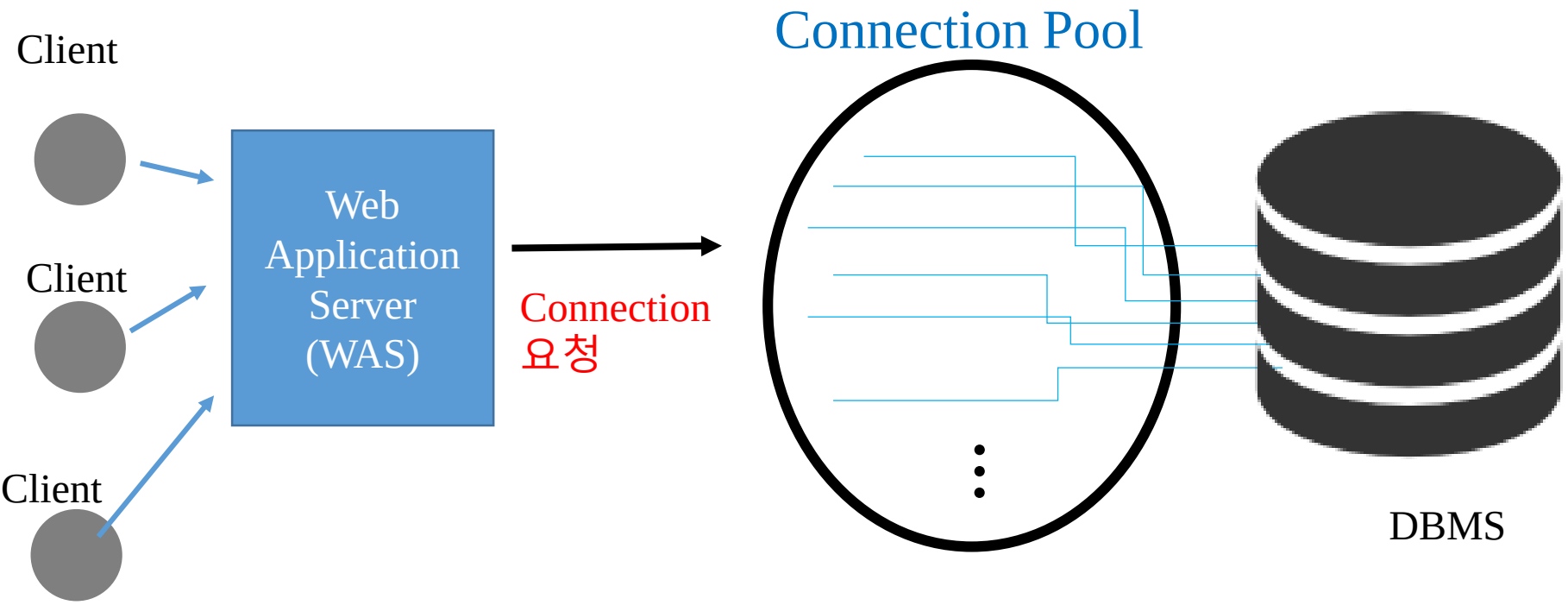


DB Connection Pool: General Web Env. [1/4]

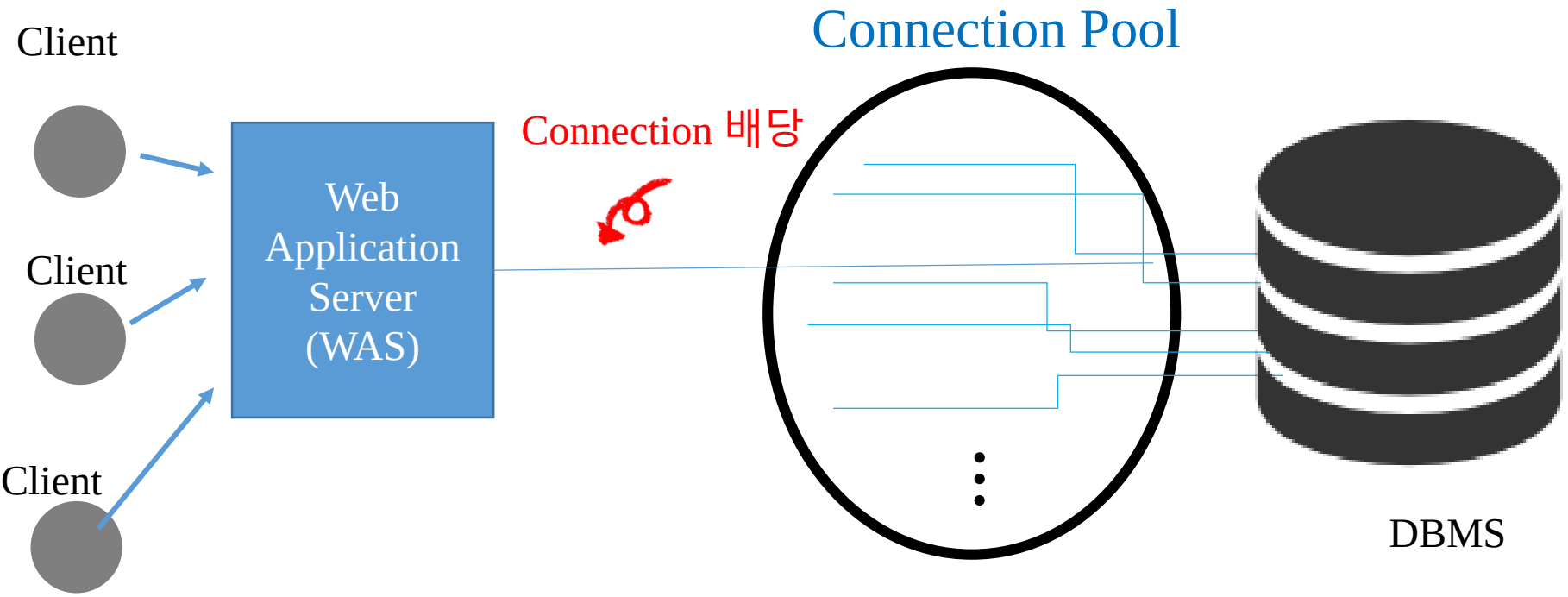


- 효율적 공유자원 관리 위해서 필요 (Evade Server Slow Down or Server Crash)
- Connection들은 **Connection pool**안에 WAS에 의해서 관리되고 Client & Server는 serve 받기만 함

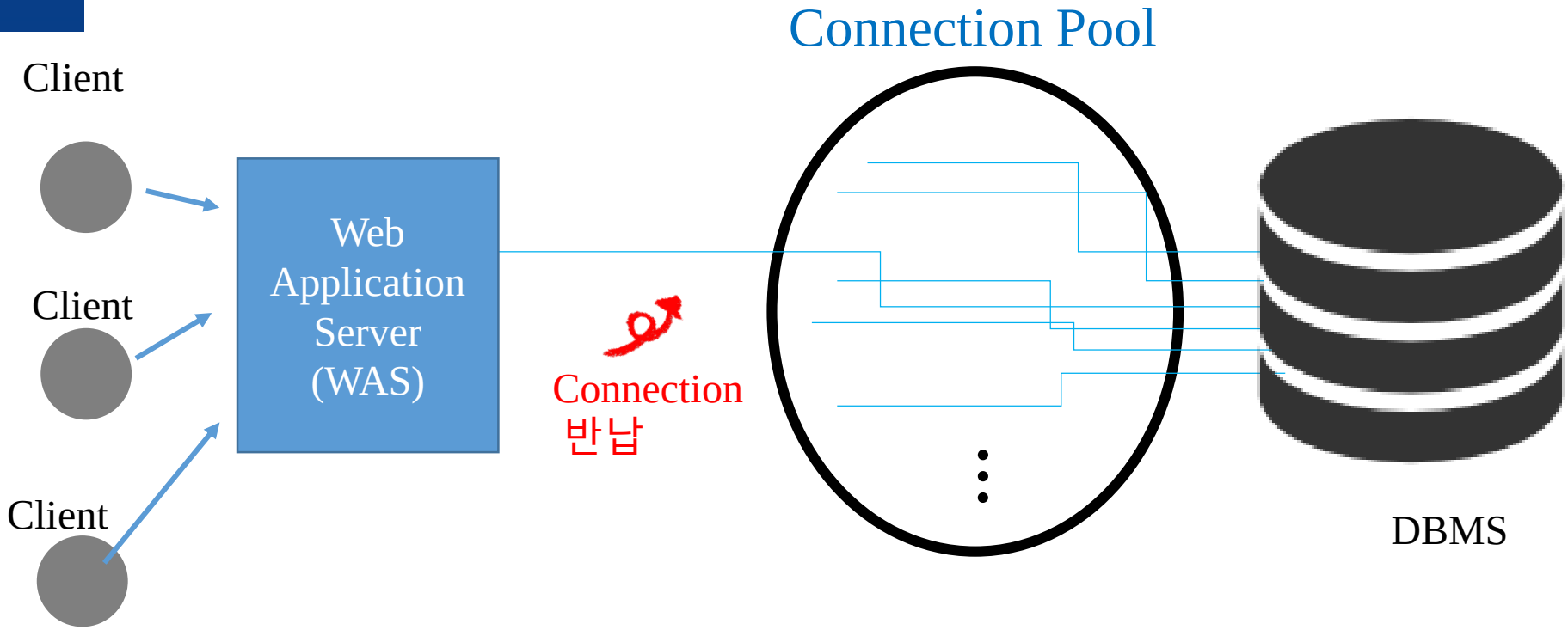
DB Connection Pool: General Web Env. [2/4]



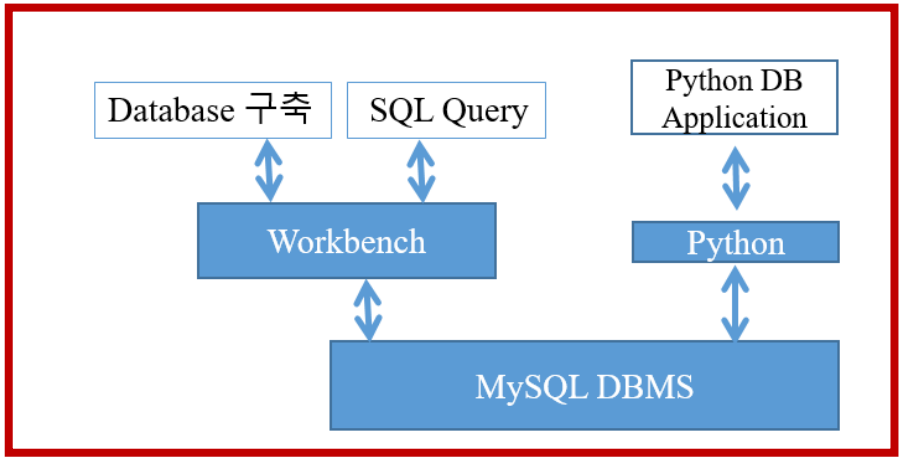
DB Connection Pool: General Web Env. [3/4]



DB Connection Pool : General Web Env. [4/4]



Our NoteBook Environment



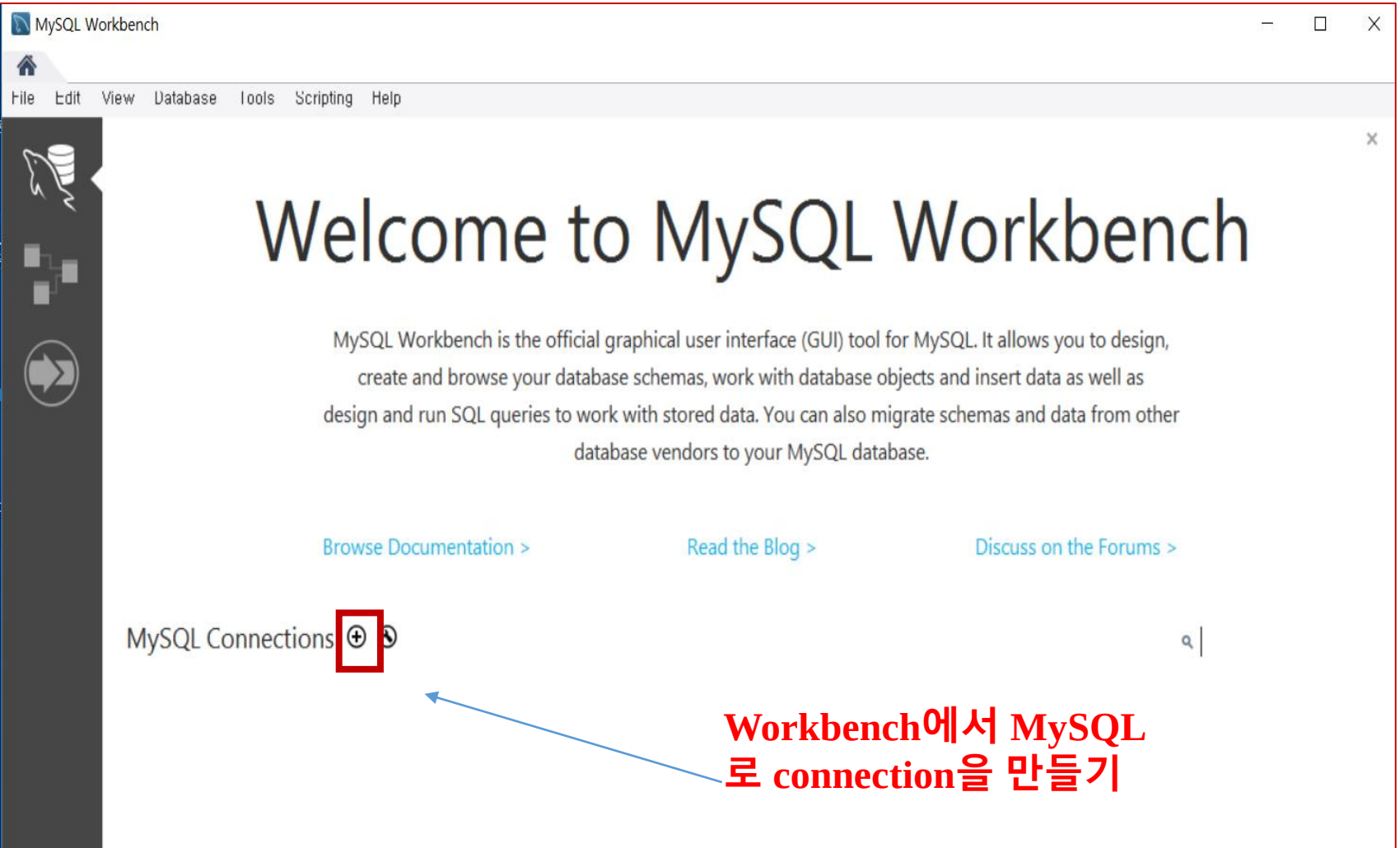
Python – MySQL Connector 종류들

- Connector/python module
  - https://dev.mysql.com/downloads/connector/python/
  - MySQL 진영에서 개발
  - **Windows에서 Python 3.5 이상 미지원!**
  - >>> import mysql.connector
- PyMySQL module
  - https://github.com/PyMySQL/PyMySQL/
  - Installation via pip: pip install PyMySQL
  - Documentation: http://pymysql.readthedocs.io
  - >>> import pymysql.cursors
- **MySQLdb module**
  - https://pypi.python.org/pypi/MySQL-python/1.2.5#downloads
  - pip install MySQL-python (Python3은 이것을 support 안함)
  - pip3 install mysqlclient (Python3에서 따르는 방식)
  - >>> import MySQLdb

Table of Contents

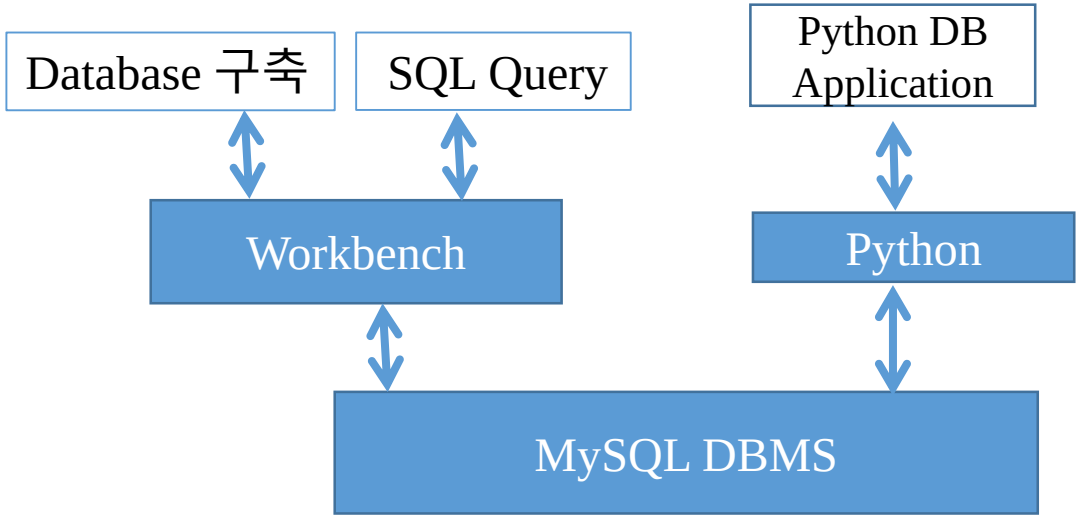
MySQL & Workbench 노트북에 설치완료!

바탕화면에 돌고래 Icon
(Workbench Icon)이 생긴것을 click

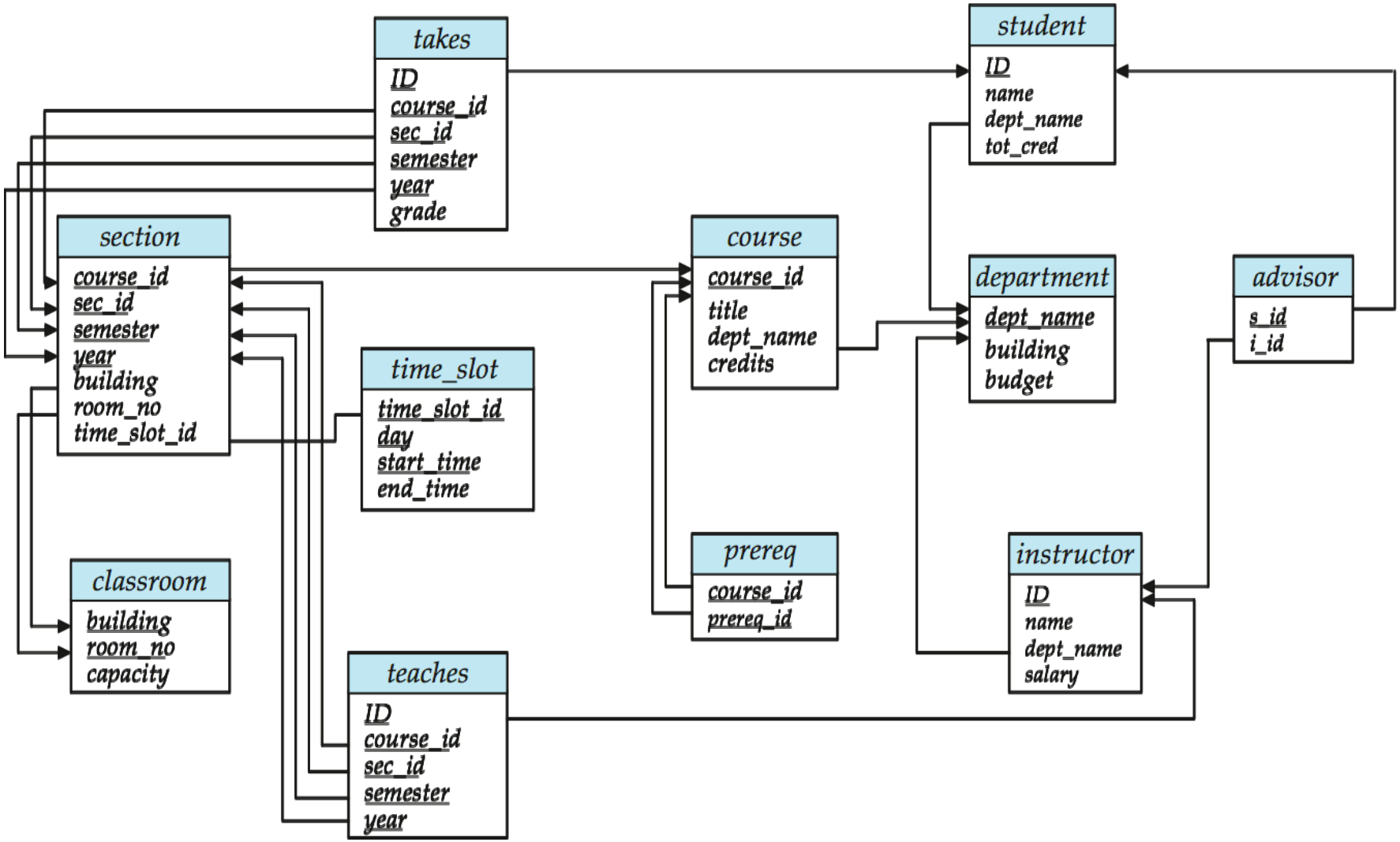


DataBase 구축과 DB Application Program

- Database 구축은 User-Friendly한 Workbench에서
- SQL Query도 User-Friendly한 Workbench에서
- DB Application Program은 Python에서



참고자료



Underline Attribute는 Primary Key, Arrow는 Foreign Key

Arrow가 끝나는곳에 있는 attribute들은 전부 underlined! (➔ key)

참고자료

DDL.sql

```sql
create table classroom
    (building       varchar(15),
     room_number        varchar(7),
     capacity       numeric(4,0),
     primary key (building, room_number)
    );

create table department
    (dept_name      varchar(20),
     building       varchar(15),
     budget             numeric(12,2) check (budget > 0),
     primary key (dept_name)
    );

create table course
    (course_id      varchar(8),
     title          varchar(50),
     dept_name      varchar(20),
     credits        numeric(2,0) check (credits > 0),
     primary key (course_id),
     foreign key (dept_name) references department(dept_name)
        on delete set null
    );

create table instructor
    (ID         varchar(5),
     name           varchar(20) not null,
     dept_name      varchar(20),
     salary         numeric(8,2) check (salary > 29000),
     primary key (ID),
     foreign key (dept_name) references department(dept_name)
        on delete set null
```

참고자료

smallRelatioanInsertFile.sql

```sql
insert into classroom values ('Packard', '101', '500');
insert into classroom values ('Painter', '514', '10');
insert into classroom values ('Taylor', '3128', '70');
insert into classroom values ('Watson', '100', '30');
insert into classroom values ('Watson', '120', '50');
insert into department values ('Biology', 'Watson', '90000');
insert into department values ('Comp. Sci.', 'Taylor', '100000');
insert into department values ('Elec. Eng.', 'Taylor', '85000');
insert into department values ('Finance', 'Painter', '120000');
insert into department values ('History', 'Painter', '50000');
insert into department values ('Music', 'Packard', '80000');
insert into department values ('Physics', 'Watson', '70000');
insert into course values ('BIO-101', 'Intro. to Biology', 'Biology', '4');
insert into course values ('BIO-301', 'Genetics', 'Biology', '4');
insert into course values ('BIO-399', 'Computational Biology', 'Biology', '3');
insert into course values ('CS-101', 'Intro. to Computer Science', 'Comp. Sci.', '4');
insert into course values ('CS-190', 'Game Design', 'Comp. Sci.', '4');
insert into course values ('CS-315', 'Robotics', 'Comp. Sci.', '3');
insert into course values ('CS-319', 'Image Processing', 'Comp. Sci.', '3');
insert into course values ('CS-347', 'Database System Concepts', 'Comp. Sci.', '3');
insert into course values ('EE-181', 'Intro. to Digital Systems', 'Elec. Eng.', '3');
insert into course values ('FIN-201', 'Investment Banking', 'Finance', '3');
insert into course values ('HIS-351', 'World History', 'History', '3');
insert into course values ('MU-199', 'Music Video Production', 'Music', '3');
insert into course values ('PHY-101', 'Physical Principles', 'Physics', '4');
insert into instructor values ('10101', 'Srinivasan', 'Comp. Sci.', '65000');
insert into instructor values ('12121', 'Wu', 'Finance', '90000');
insert into instructor values ('15151', 'Mozart', 'Music', '40000');
insert into instructor values ('22222', 'Einstein', 'Physics', '95000');
insert into instructor values ('32343', 'El Said', 'History', '60000');
insert into instructor values ('33456', 'Gold', 'Physics', '87000');
insert into instructor values ('45565', 'Katz', 'Comp. Sci.', '75000');
insert into instructor values ('58583', 'Califieri', 'History', '62000');
insert into instructor values ('76543', 'Singh', 'Finance', '80000');
insert into instructor values ('76766', 'Crick', 'Biology', '72000');
insert into instructor values ('83821', 'Brandt', 'Comp. Sci.', '92000');
```

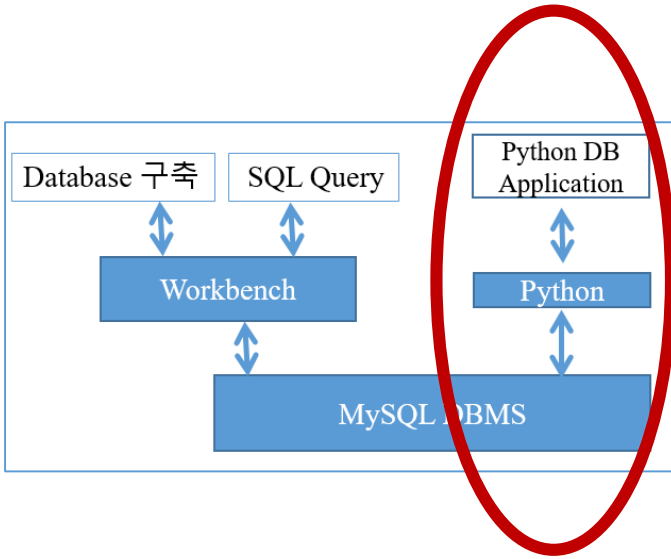
Now, We are Here!

NoteBook

Python + MySQLdb module

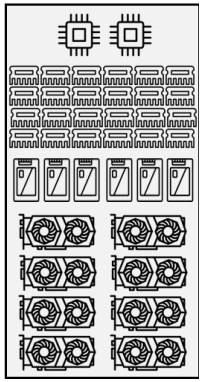
Workbench IDE

MySQL DBMS



NoteBook

DS Server



Python + MySQLdb module

Workbench IDE

MySQL DBMS

Now we move to Python + MySQL DBMS

- So far …

- MySQL DBMS Installer로 설치완료
  - Server, Workbench, Connector/Python,….
- MySQL Workbench에서 python_testdb schema 생성
- MySQL Workbench에서 python_testdb schema 내부에 table들을 생성
- MySQL Workbench에서 table들 내부로 record 입력 완성

- Now…

- We want to make a Python Database Application Program which can communicate with MySQL
- First, we need to install MySQLdb module

(Method A) MySQLdb Module - mysqlclient

- MySQLdb module을 사용하려면 MySQL-python 를 install해야 하는데 (Python3에서는 지원을 안함)
- mysqlclient is a wrapper of MySQL-python, supports Python3
  - PyPI에 있는 mysqlclient module
  - Install mysqlclient by pip in order to import MySQLdb
- MySQL DBMS Version 5.7 까지만 지원 (as of 2018, Aug)

C++ Visual Studio 가 PC에 설치되어 있으때만 작동

```
C:₩Users₩dalaetm>pip3 install mysqlclient
Collecting mysqlclient
```

```
C:₩WINDOWS₩system32>pip install mysqlclient
Collecting mysqlclient
  Using cached https://files.pythonhosted.org/packages/b4/76/ced0523e11
1/mysqlclient-1.4.1-cp37-cp37m-win_amd64.whl
Installing collected packages: mysqlclient
Successfully installed mysqlclient-1.4.1

C:₩WINDOWS₩system32>_
```

Now we can do the following

```
>>> import MySQLdb
```

(Method B) Anaconda에서 mysqlclient 설치

Click Windows Button → Anaconda Prompt → 우클릭 및 관리자 권한으로 실행 → conda install –c bioconda mysqlclient

관리자: Anaconda Prompt

```
(C:₩Users₩user₩Anaconda3) C:₩WINDOWS₩system32>conda install -c bioconda mysqlclient
Fetching package metadata .................
Solving package specifications: .

Package plan for installation in environment C:₩Users₩user₩Anaconda3:

The following NEW packages will be INSTALLED:

    mysql-connector-c: 6.1.11-hf3e53a5_0
    mysqlclient:       1.3.13-py36hfa6e2cd_0

The following packages will be UPDATED:

    conda:             4.3.30-py36h7e176b0_0 --> 4.5.11-py36_0
    conda-env:         2.6.0-0               --> 2.6.0-1
    libpng:            1.6.30-vc14_1         --> 1.6.34-h79bbb47_0
    pycosat:           0.6.2-py36_0          --> 0.6.3-py36hfa6e2cd_0

Proceed ([y]/n)? y

conda-env-2.6. 100% |###############################| Time: 0:00:00 176.30 kB/s
mysql-connecto 100% |###############################| Time: 0:00:01   4.66 MB/s
libpng-1.6.34- 100% |###############################| Time: 0:00:00  24.42 MB/s
mysqlclient-1. 100% |###############################| Time: 0:00:00  24.37 MB/s
pycosat-0.6.3- 100% |###############################| Time: 0:00:00  15.10 MB/s
conda-4.5.11-p 100% |###############################| Time: 0:00:00  22.84 MB/s

(C:₩Users₩user₩Anaconda3) C:₩WINDOWS₩system32>
```

(Method C) Python에서 .whl mysqlclient 다운로드 [1/2]

- .whl file : Python용 precompiled binary file
- https://www.lfd.uci.edu/~gohlke/pythonlibs/#mysqlclient
  - Python 버전과 OS bit 수에 따라 whl file 선택하여 download

Mysqlclient, a fork of the MySQL-python interface for the MySQL database.

mysqlclient-1.3.13-cp27-cp27m-win32.whl
mysqlclient-1.3.13-cp27-cp27m-win_amd64.whl
mysqlclient-1.3.13-cp34-cp34m-win32.whl
mysqlclient-1.3.13-cp34-cp34m-win_amd64.whl
mysqlclient-1.3.13-cp35-cp35m-win32.whl
mysqlclient-1.3.13-cp35-cp35m-win_amd64.whl
mysqlclient-1.3.13-cp36-cp36m-win32.whl
mysqlclient-1.3.13-cp36-cp36m-win_amd64.whl
mysqlclient-1.3.13-cp37-cp37m-win32.whl
mysqlclient-1.3.13-cp37-cp37m-win_amd64.whl

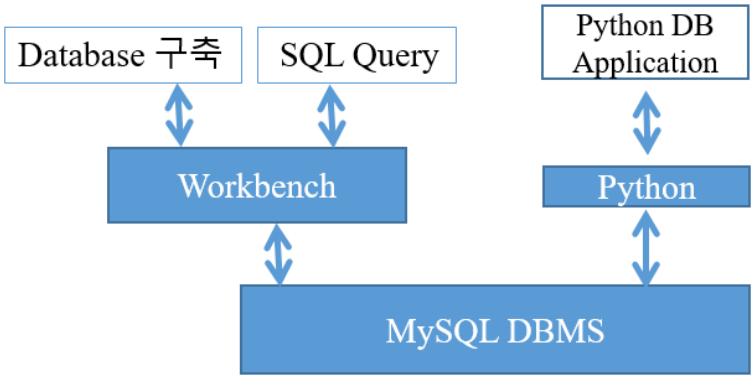
Python 3.6 or 3.7 이 대부분이고, OS는 64bit이 대부분

(Method C) Python에서 .whl mysqlclient 다운로드 [2/2]

- CMD창 실행 → download 받은 whl file 이 있는 곳으로 이동
- pip3 install [download받은 whl file]

```
(venv) C:\Users\lathe\Downloads>pip install mysqlclient-1.3.7-cp34-none-win32.whl
Processing c:\users\lathe\downloads\mysqlclient-1.3.7-cp34-none-win32.whl
Installing collected packages: mysqlclient
Successfully installed mysqlclient-1.3.7
You are using pip version 7.1.2, however version 8.1.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

(venv) C:\Users\lathe\Downloads>
```

- OK!
- We are really really ready to do the real stuff !
- Python Database Application Coding!



MySQLdb Module 구동 – Connection Object [1/2]

- Connection Constructor

connect (*parameters...*)

Constructor for creating a connection to the database.

Returns a Connection Object. It takes a number of parameters which are database dependent. [1]

- Parameters
  - host: name of host to connect to
  - user: user to authenticate as. Default: current effective user.
  - passwd: password to authenticate with. Default: no password.
  - db: database to use.
  - port: TCP port of MySQL server. Default: standard port (3306).

- example

```
import MySQLdb
db=MySQLdb.connect(passwd="moonpie",db="thangs")
```

MySQLdb Module 구동 – Connection Object [2/2]

- Methods in Connection Object (say, db)

db **.cursor ()**

Return a new Cursor Object using the connection.

db **.close()**

Close the connection now (rather than whenever .__del__() is called).

db **.commit ()**

Commit any pending transaction to the database.

db .rollback()

- this rolls back (cancels) the current transaction

MySQLdb Module – Cursor Object [1/3]

- Cursor object (c = connection_object.cursor( ))
  - represent a database cursor, which is used to manage the context of a fetch operation
- Attributes of cursor object

  c.description
  - This read-only attribute is a sequence of 7 items: name, type_code, display_size, internal_size, precision, scale, null_ok

  c.rowcount
  - This read-only attribute specifies the number of rows that the last .execute*( ) produced (for statements like SELECT) or affected (for DML like UPDATE or INSERT)

  c.arraysize
  - This read/write attribute specifies the number of rows to fetch at a time with .fetchmany( ). It defaults to 1 meaning to fetch a single row at a time

MySQLdb Module – Cursor Object [2/3]

- Cursor methods

c **.execute** (*operation* [, *parameters*])

Prepare and execute a database operation (query or command).

- Parameters may be provided as sequence or mapping and will be bound to variables in the operation.

c **.executemany** (*operation* , *seq_of_parameters*)

Prepare a database operation (query or command) and then execute it against all parameter sequences or mappings found in the sequence *seq_of_parameters* .

MySQLdb Module – Cursor Object [3/3]

- Methods of cursor object

c **.fetchmany** ([*size=cursor.arraysize*])

Fetch the next set of rows of a query result, returning a sequence of sequences (e.g. a list of tuples). An empty sequence is returned when no more rows are available.

- The number of rows to fetch per call is specified by the parameter
- If it is not given, the cursor's arraysize determines the number of rows to be fetched

c.**fetchone( )**： Fetch the next tuple of a query result

c **.fetchall** ()

Fetch all (remaining) rows of a query result, returning them as a sequence of sequences (e.g. a list of tuples). Note that the cursor's arraysize attribute can affect the performance of this operation.

c **.close** ()

Close the cursor now (rather than whenever __del__ is called).

Connection to MySQL using MySQLdb Module

To MySQL in DS Server

```
import MySQLdb as mysql

db = mysql.connect(host="ds1.snu.ac.kr", user="ds3_1", passwd="ds3_1", db="ds3_1")
```

DB 접속

To MySQL in NotebookPC

```
import MySQLdb
db = MySQLdb.connect("localhost","root","********","python_testdb")
```

Workbench에서 생성된 database

Connection to MySQL in NotebookPC using MySQLdb Module

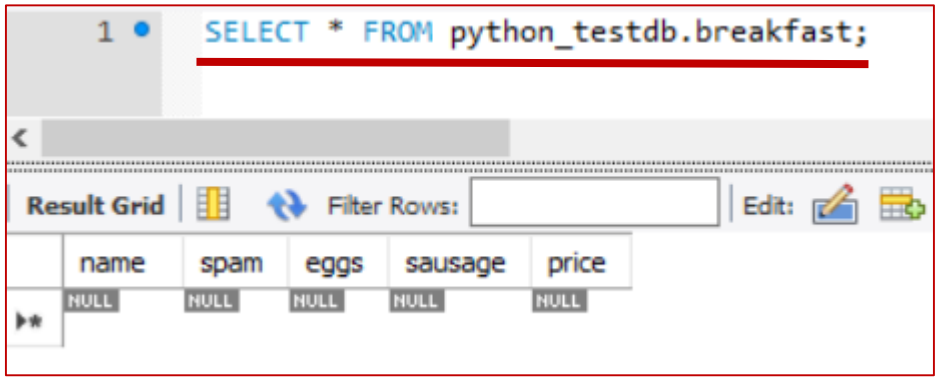
- Python에서 Connection Object생성, Cursor Object생성, Table 생성

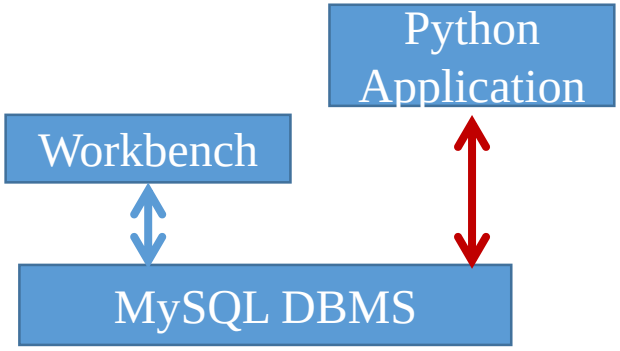
```
import MySQLdb

db = MySQLdb.connect("localhost","root","********","python_testdb")

c = db.cursor()
c.execute("CREATE TABLE breakfast(₩
              name VARCHAR(32) NOT NULL,₩
              spam INT,₩
              eggs INT,₩
              sausage INT,₩
              price FLOAT(8,2),₩
              PRIMARY KEY (name)₩
              );")
```

Workbench에서 생성된 database

Line을 계속 연결한다는 의미

MySQL workbench에 가서 결과 확인





Insert a Record using MySQLdb Module

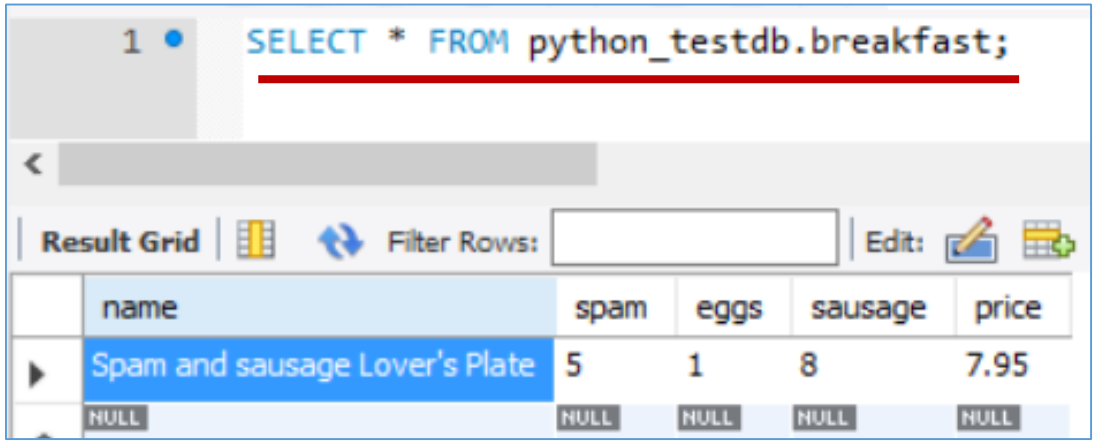
- Python에서 cursor.execute( ) 실행, coursor.rowcount 읽기

```
try:
    c.execute("INSERT INTO breakfast(name, spam, eggs, sausage, price)₩
               VALUES(₩"Spam and sausage Lover's Plate₩", 5,1,8,7.95)")
    db.commit()

except:
    db.rollback()

print("num of inserted row : %d"% c.rowcount)
```

DBMS가 c.execute( )를 성공하면 DBMS에게 db.commit( )을 요청
실패하면 db.rollback( ) 을 요청

```
num of inserted row : 1
```

MySQL workbench에 가서 확인



Insert Many Records using MySQLdb Module

- Python에서 cursor.executemany( ) 실행

```
try:
    c.executemany("INSERT INTO breakfast(name, spam, eggs, sausage, price)₩
                   VALUES(%s,%s,%s,%s,%s)",₩
                  [₩
                  ("Not So Much Spam Plate", 3,2,0,3.95),₩
                  ("Don't Want ANY SPAM! Plate",0,4,3,5.95)₩
                  ])
    db.commit()

except:
    db.rollback()

print("num of inserted row : %d"% c.rowcount)
```

```
num of inserted row :  3
```

MySQL workbench에 가서 확인

	name	spam	eggs	sausage	price
▶	Don't Want ANY SPAM! Plate	0	4	3	5.95
	Not So Much Spam Plate	3	2	0	3.95
	Spam and sausage Lover's Plate	5	1	8	7.95
*	NULL	NULL	NULL	NULL	NULL

SQL Query using MySQLdb Module

- Python에서 cursor.fetchall( ) 실행

```
min_price = 5

c.execute("SELECT name, spam, eggs, sausage₩
           FROM breakfast₩
           ₩HERE price > %s",(min_price,))

data = c.fetchall()

for row in data:
    print(row)
```

```
("Don't Want ANY SPAM! Plate", 0, 4, 3)
("Spam and sausage Lover's Plate", 5, 1, 8)
```

MySQL workbench에 가서 확인

	name	spam	eggs	sausage	price
▶	Don't Want ANY SPAM! Plate	0	4	3	5.95
	Not So Much Spam Plate	3	2	0	3.95
	Spam and sausage Lover's Plate	5	1	8	7.95
*	NULL	NULL	NULL	NULL	NULL

DBMS Version Check using MySQLdb Module

Python application 개발중에 DBMS Vresion Check이 필요할때도 있다

```
import MySQLdb
db = MySQLdb.connect("localhost","root","******","python_testdb" )
cursor = db.cursor()
cursor.execute("SELECT VERSION()")
data = cursor.fetchone()
print ("Database version : %s " % data)
db.close()
```

```
Database version : 5.7.11-log
```

DS Server의 MySQL에 MySQLdb 사용 예제

```
1   import MySQLdb as mysql
2
3   db = mysql.connect(host="ds1.snu.ac.kr", user="ds3_1", passwd="ds3_1", db="ds3_1")
4
5   cur = db.cursor()
6   cur.execute('create table test(col1 integer, col2 integer, col3 integer)')
7   cur.execute('insert into test values(1, 2, 3);')
8   cur.execute('insert into test values(4, 5, 6);')
9   cur.execute('insert into test values(7, 8, 9);')
10  cur.execute('select * from test;')
11
12  rows = cur.fetchall()
13  print(rows)
```

DB 접속

쿼리 실행

결과 반환

```
((1, 2, 3), (4, 5, 6), (7, 8, 9))
```

참고

_mysql C library 사용

```
import _mysql
```

```
db = _mysql.connect (host="ds1.snu.ac.kr", user="ds3_1", passwd="ds3_1", db="ds3_1")
```

DB 접속

```
db.query('create table test(col1 integer, col2 integer, col3 integer)')
db.query('insert into test values(1, 2, 3);')
db.query('insert into test values(4, 5, 6);')
db.query('insert into test values(7, 8, 9);')
db.query('select * from test;')
```

쿼리 요청

```
result = db.store_result()
rows = result.fetch_row(3, 1)
```

결과 반환

```
print(rows)
```

0: 튜플로 반환, 1: 딕셔너리로 반환

최대 3개의 row를 가져옴

```
({'col1': 1, 'col2': 2, 'col3': 3}, {'col1': 4, 'col2': 5, 'col3': 6}, {'col1': 7, 'col2': 8, 'col3': 9})
```