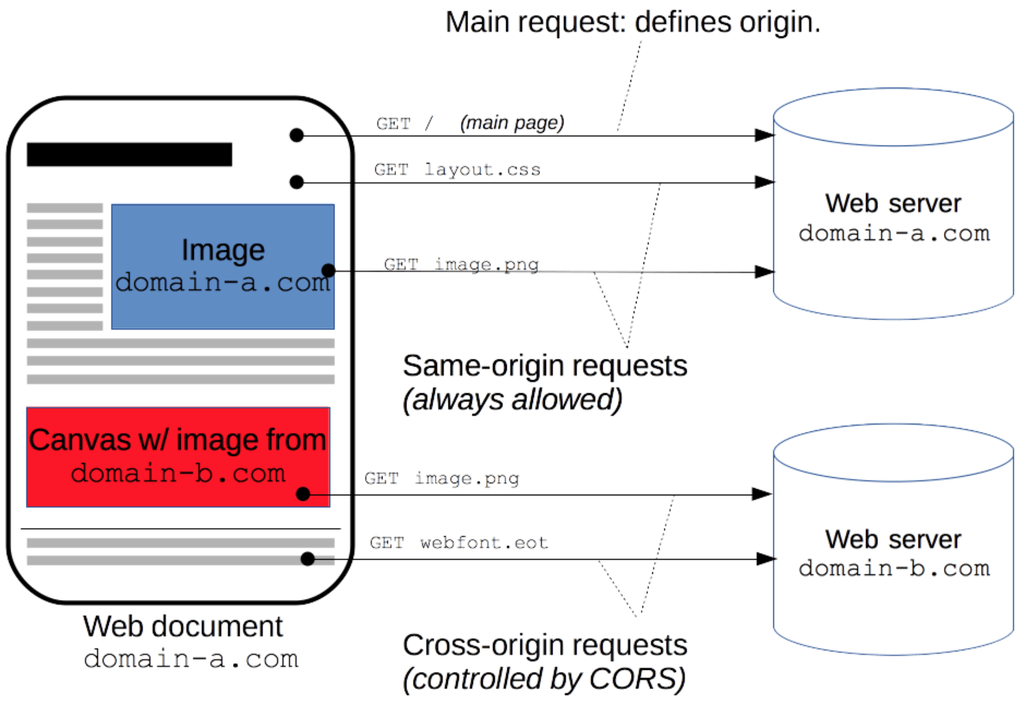
Week 8

Topic 3: Data Verification

- 前端
  - 提供即時反饋、減少等待時間
  - 減少錯誤的資料輸入，降低伺服器負擔
  - 提供可視化的錯誤訊息，幫助使用者快速修正
- 後端
  - 提供安全性保障，防止惡意攻擊
  - 確保數據完整及一致性
  - 前後端分離時，依然有數據驗證邏輯

Topic 4: Fetch and CORS

- Cross-Origin Resource Sharing，跨來源資源共享
- 基於安全性考量，程式碼所發出的跨來源 HTTP 請求會受到限制
- 制提供了網頁伺服器跨網域的存取控制，增加跨網域資料傳輸的安全性



https://developer.mozilla.org/zh-TW/docs/Web/HTTP/CORS

Topic 4: Fetch and CORS

- Fetch: https://www.google.com/
- Can not get a response. It has been blocked by CORS policy.

```
<script>
    fetch("https://www.google.com/").then((response)=>{
        console.log(response)
    })
</script>
```

```
Access to fetch at '  index.html:1
https://www.google.com/' from
origin 'http://127.0.0.1:5500' has
been blocked by CORS policy: No
'Access-Control-Allow-Origin'
header is present on the requested
resource. If an opaque response
serves your needs, set the
request's mode to 'no-cors' to
fetch the resource with CORS
disabled.

▶GET                index.html:10
https://www.google.com/
net::ERR_FAILED 200 (OK)

▶Uncaught      index.html:10
(in promise)
TypeError: Failed to fetch
    at index.html:10:9
```

Topic 4: Fetch and CORS

- Fetch: https://padax.github.io/taipei-day-trip-resources/taipei-attractions-assignment.json

```
<script>
      fetch(
        "https://padax.github.io/taipei-day-trip-resources/taipei-attractions-assignment.json"
      ).then((response) => {
        console.log(response);
      });
```

```
                                    index.html:13
Response {type: 'cors', url: 'htt
ps://padax.github.io/taipei-day-t
▶ rip-resources/taipei-attractions-
assignment.json', redirected: fal
se, status: 200, ok: true, …}
```

Topic 4: Fetch and CORS

Backend

```python
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

# origins = ["http://127.0.0.1:5500"]

# 加入 CORSMiddleware
# app.add_middleware(
#     CORSMiddleware,
#     allow_origins=origins,
#     allow_credentials=True,
#     allow_methods=["*"],
#     allow_headers=["*"],
# )

@app.get("/")
async def get_data():
    return {"data": "hello"}

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="localhost", port=8000)
```

Frontend

```html
<script>
        fetch("http://localhost:8000/").then((response) => {
          console.log(response);
        });
    </script>
```

```
Access to fetch at 'http://localhost:8000/' from origin 'http://127.0.0.1:5500' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.                index.html:1

▶ GET http://localhost:8000/ net::ERR_FAILED 200 (OK)                index.html:27

▶ Uncaught (in promise) TypeError: Failed to fetch
    at index.html:27:7                index.html:27
```

Topic 4: Fetch and CORS

Backend

```python
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

origins = ["http://127.0.0.1:5500"]

# 加入 CORSMiddleware
app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

@app.get("/")
async def get_data():
    return {"data": "hello"}

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="localhost", port=8000)
```

Frontend

```html
<script>
      fetch("http://localhost:8000/")
        .then((response) => {
          console.log(response);
          return response.json();
        })
        .then((data) => {
          console.log(data);
        });
    </script>
```

```
                                    index.html:29
  Response {type: 'cors', url: 'http://
▶ localhost:8000/', redirected: false,
  status: 200, ok: true, …}
─────────────────────────────────────────
▶ {data: 'hello'}                   index.html:33
```

Topic 5: Primary Key and Index

- Primary key
  - 唯一性和完整性
  - 快速查找與記錄
  - 與 foreign key 搭配使用，建立 table 之間的關係，維護資料參照完整性與數據一致性

Topic 5: Primary Key and Index

- Index
  - 提高查詢性能
  - 加速排序與分組操作
  - 加速連結操作

Topic 5: Primary Key and Index

```python
import mysql.connector
import random, string

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="password",
    database="testDB"
)

mycursor = mydb.cursor(dictionary=True)

name = "test"
username = "test"
password = "test"

# 新增帳號
sql = "INSERT INTO member (name, username,password) VALUES (%s,%s,%s)"
val = (name, username, password)
mycursor.execute(sql,val)
mydb.commit()
print(mycursor.rowcount)

for i in range (1000000):
    name = ''.join(random.choice(string.ascii_letters) for x in range(2))
    username = ''.join(random.choice(string.ascii_letters) for x in range(2))
    password = ''.join(random.choice(string.ascii_letters) for x in range(2))
    sql = "INSERT INTO member (name, username,password) VALUES (%s,%s,%s)"
    val = (name, username, password)
    mycursor.execute(sql,val)
    mydb.commit()
```

- name, username, password: 2 digits [A-Z] [a-z]
- Data: 1,000,000

```
mysql> SELECT * FROM member;
+---------+------+----------+----------+
| id      | name | username | password |
+---------+------+----------+----------+
|       1 | test | test     | test     |
|       2 | CQ   | VM       | nZ       |
|       3 | xm   | LN       | Tj       |
|       4 | Tx   | ZH       | tw       |
|       5 | mt   | WB       | Yf       |
|       6 | QC   | bi       | AY       |
|       7 | OO   | Qp       | jJ       |
|       8 | NO   | Th       | Jn       |
|       9 | Wy   | kM       | mT       |
|      10 | uA   | ov       | vC       |
|      11 | KQ   | Jn       | Ct       |
|      12 | FO   | Jo       | ek       |
|      13 | yP   | jY       | BW       |
|      14 | qS   | va       | Ag       |
|      15 | JZ   | he       | Ah       |
|      16 | Fa   | ou       | dE       |
```

Topic 5: Primary Key and Index

- SELECT * FROM member WHERE username=‘Vk’ and password=‘Fu’ => 0.36 secs

```
mysql> SELECT * FROM member WHERE username='Vk' and password='Fu';
+--------+------+----------+----------+----------+---------------------+
| id     | name | username | password | follower | time                |
+--------+------+----------+----------+----------+---------------------+
| 121804 | RZ   | VK       | fu       |        0 | 2024-05-24 00:57:29 |
| 499185 | hO   | vK       | fu       |        0 | 2024-05-24 00:58:43 |
| 999988 | Nm   | Vk       | Fu       |        0 | 2024-05-24 01:00:20 |
+--------+------+----------+----------+----------+---------------------+
3 rows in set (0.36 sec)
```

- SELECT * FROM member WHERE username=‘Vk’ and password=‘Fu’ and name=‘RZ’ => 0.36 secs

```
mysql> SELECT * FROM member WHERE username='Vk' and password='Fu' and name='RZ';
+--------+------+----------+----------+----------+---------------------+
| id     | name | username | password | follower | time                |
+--------+------+----------+----------+----------+---------------------+
| 121804 | RZ   | VK       | fu       |        0 | 2024-05-24 00:57:29 |
+--------+------+----------+----------+----------+---------------------+
1 row in set (0.36 sec)
```

```
mysql> EXPLAIN SELECT * FROM member WHERE username='Vk' and password='Fu';
+----+-------------+--------+------------+------+---------------+------+---------+------+--------+----------+-------------+
| id | select_type | table  | partitions | type | possible_keys | key  | key_len | ref  | rows   | filtered | Extra       |
+----+-------------+--------+------------+------+---------------+------+---------+------+--------+----------+-------------+
|  1 | SIMPLE      | member | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 997497 |     1.00 | Using where |
+----+-------------+--------+------------+------+---------------+------+---------+------+--------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> EXPLAIN SELECT * FROM member WHERE username='Vk' and password='Fu' and name='RZ';
+----+-------------+--------+------------+------+---------------+------+---------+------+--------+----------+-------------+
| id | select_type | table  | partitions | type | possible_keys | key  | key_len | ref  | rows   | filtered | Extra       |
+----+-------------+--------+------------+------+---------------+------+---------+------+--------+----------+-------------+
|  1 | SIMPLE      | member | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 997497 |     0.10 | Using where |
+----+-------------+--------+------------+------+---------------+------+---------+------+--------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> CREATE INDEX idx_username_password ON member (username,password);
Query OK, 0 rows affected (1.53 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Topic 5: Primary Key and Index

SELECT * FROM member WHERE username='Vk' and pass...	3 row(s) returned	0.355 sec / 0.0000060 sec
USE testDB	0 row(s) affected	0.00067 sec
SELECT * FROM member WHERE username='Vk' and pass...	3 row(s) returned	0.0016 sec / 0.0000079 sec

- SELECT * FROM member WHERE username=‘Vk’ and password=‘Fu’ => 0.001 secs

```
mysql> SELECT * FROM member WHERE username='Vk' and password='Fu';
+--------+------+----------+----------+----------+---------------------+
| id     | name | username | password | follower | time                |
+--------+------+----------+----------+----------+---------------------+
| 121804 | RZ   | VK       | fu       |        0 | 2024-05-24 00:57:29 |
| 499185 | hO   | vK       | fu       |        0 | 2024-05-24 00:58:43 |
| 999988 | Nm   | Vk       | Fu       |        0 | 2024-05-24 01:00:20 |
+--------+------+----------+----------+----------+---------------------+
3 rows in set (0.01 sec)
```

- SELECT * FROM member WHERE username=‘Vk’ and password=‘Fu’ and name=‘RZ’ => 0.001 secs

```
mysql> SELECT * FROM member WHERE username='Vk' and password='Fu' and name='RZ';
+--------+------+----------+----------+----------+---------------------+
| id     | name | username | password | follower | time                |
+--------+------+----------+----------+----------+---------------------+
| 121804 | RZ   | VK       | fu       |        0 | 2024-05-24 00:57:29 |
+--------+------+----------+----------+----------+---------------------+
1 row in set (0.00 sec)
```

```
mysql> EXPLAIN SELECT * FROM member WHERE username='Vk' and password='Fu';
+----+-------------+--------+------------+------+----------------------+----------------------+---------+-------------+------+----------+-------+
| id | select_type | table  | partitions | type | possible_keys        | key                  | key_len | ref         | rows | filtered | Extra |
+----+-------------+--------+------------+------+----------------------+----------------------+---------+-------------+------+----------+-------+
|  1 | SIMPLE      | member | NULL       | ref  | idx_username_password | idx_username_password | 2044    | const,const |    3 |   100.00 | NULL  |
+----+-------------+--------+------------+------+----------------------+----------------------+---------+-------------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

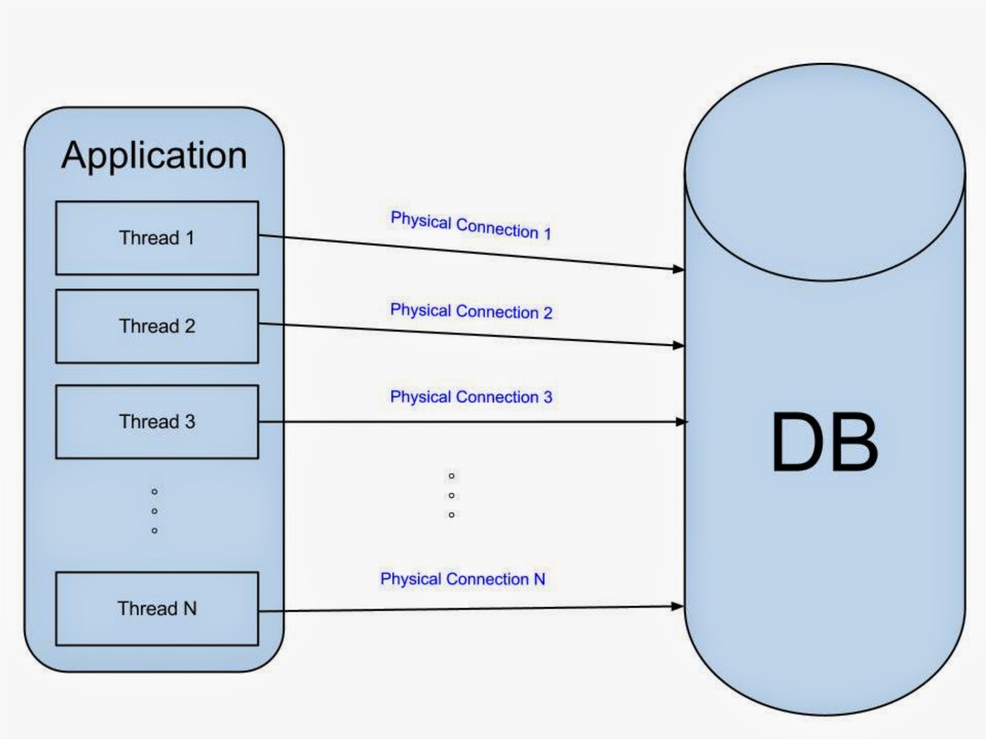
```
mysql> EXPLAIN SELECT * FROM member WHERE username='Vk' and password='Fu' and name='RZ';
+----+-------------+--------+------------+------+----------------------+----------------------+---------+-------------+------+----------+-------------+
| id | select_type | table  | partitions | type | possible_keys        | key                  | key_len | ref         | rows | filtered | Extra       |
+----+-------------+--------+------------+------+----------------------+----------------------+---------+-------------+------+----------+-------------+
|  1 | SIMPLE      | member | NULL       | ref  | idx_username_password | idx_username_password | 2044    | const,const |    3 |    10.00 | Using where |
+----+-------------+--------+------------+------+----------------------+----------------------+---------+-------------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

Topic 5: Primary Key and Index

- 前綴匹配 (LIKE 'abc%'): 索引可以被利用並加速查詢。
- 後綴匹配 (LIKE '%xyz'): 索引無法被有效利用，導致全表掃描。
- 中間匹配 (LIKE '%abc%'): 索引無法被有效利用，導致全表掃描。
- 全文索引: 適用於更複雜的文本搜索，比 LIKE 操作有顯著的性能提升。

Topic 6: Connection Pool

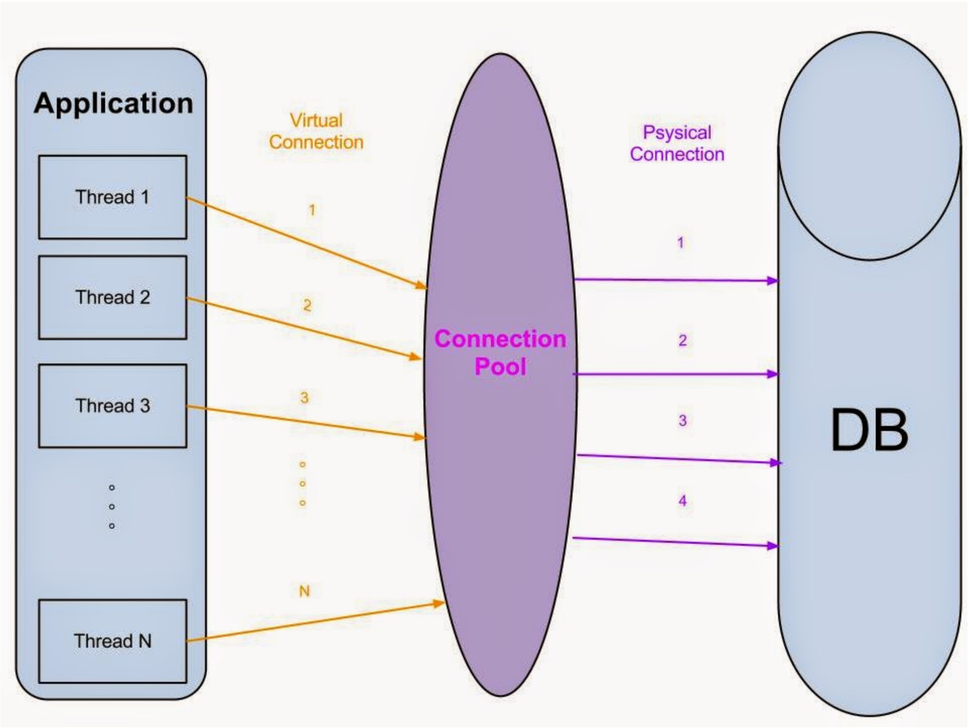
- Connection Pool(連線池) 是一種資料庫連線管理的機制，它介於應用程式與資料庫之間
- 集中管理資料庫的連線，能有效提升應用程式存取資料庫的效能及減少連線的錯誤
- 資料庫連線的建立成本是昂貴的，故當有許多Thread都需要建立connection時，其資源的耗費是龐大的
- 一個成本昂貴的connection在程式使用完後，馬上就被Close掉，在使用上並沒有達到效益極大化。



https://peggg327.blogspot.com/2014/11/connection-pool.html

Topic 6: Connection Pool

- Pool會keep住與DB的連線。程式需要使用時跟pool要即可。
- 可設定與DB最大的連線數，避免超過DB所能負擔的連線數。



https://peggg327.blogspot.com/2014/11/connection-pool.html

Topic 6: Connection Pool

```python
from mysql.connector import pooling

# 資料庫資訊
mydb = {
    "host":"localhost",
    "user":"root",
    "password":"password",
    "database":"testDB"
}

# 建立連接池
cnxpool = pooling.MySQLConnectionPool(pool_name="mypool", pool_size=5, **mydb)

# 從連接池取得連線
cnx = cnxpool.get_connection()

# 進行操作
mycursor = cnx.cursor()

# 查詢帳號
command = "SELECT * FROM member WHERE username=%s and password=%s"
username="Vk"
password="Fu"
val = (username, password)
mycursor.execute(command,val)
myresult = mycursor.fetchall()
print(myresult)

# 關閉游標與連接
mycursor.close()
cnx.close()
```

Topic 7: Cross-Site Scripting (XSS)

- 跨站腳本攻擊（Cross-Site Scripting，縮寫為XSS）是一種常見的網絡安全漏洞，發生在攻擊者能夠在其他用戶的瀏覽器中注入惡意腳本時。這些腳本通常是JavaScript，但也可能是其他類型的代碼。XSS漏洞主要分為三種類型：

  1. 反射型（Reflected XSS）：惡意腳本隨著URL參數或者其他即時輸入被注入，當受害者點擊惡意鏈接或提交表單時，腳本會在受害者的瀏覽器中執行。

  2. 儲存型（Stored XSS）：惡意腳本被永久存儲在目標伺服器上，例如在數據庫、留言板、社交媒體貼文等。當其他用戶訪問包含惡意腳本的頁面時，腳本自動執行。

  3. DOM型（DOM-based XSS）：這種類型的XSS攻擊是通過操作瀏覽器的DOM（文檔對象模型）來實現的，這些操作不會經過伺服器，而是在客戶端動態地改變頁面結構。