Nodejs MySQL DB connection pool

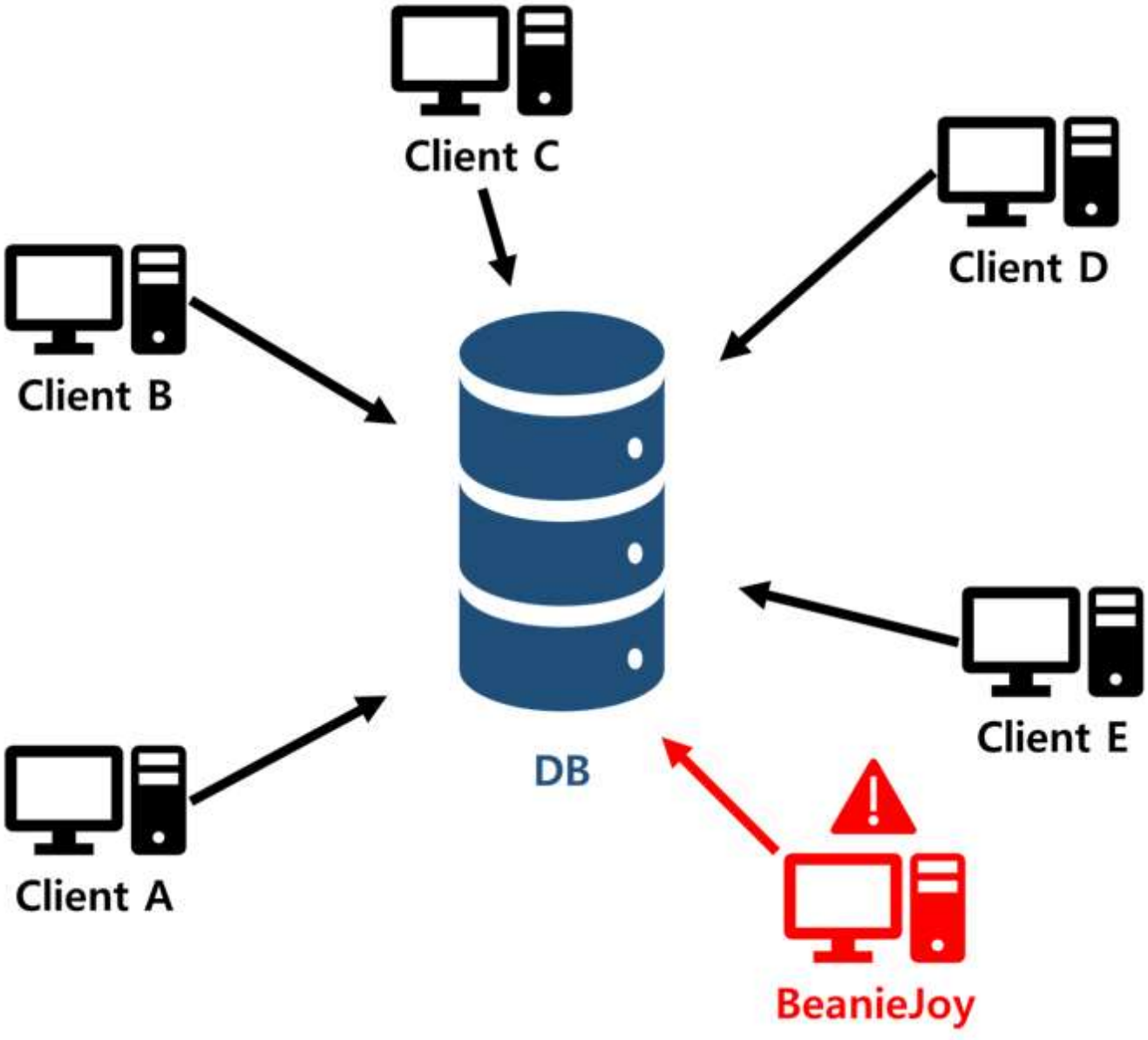
DB Connection Pool의 등장 배경

기존 Connection 방식

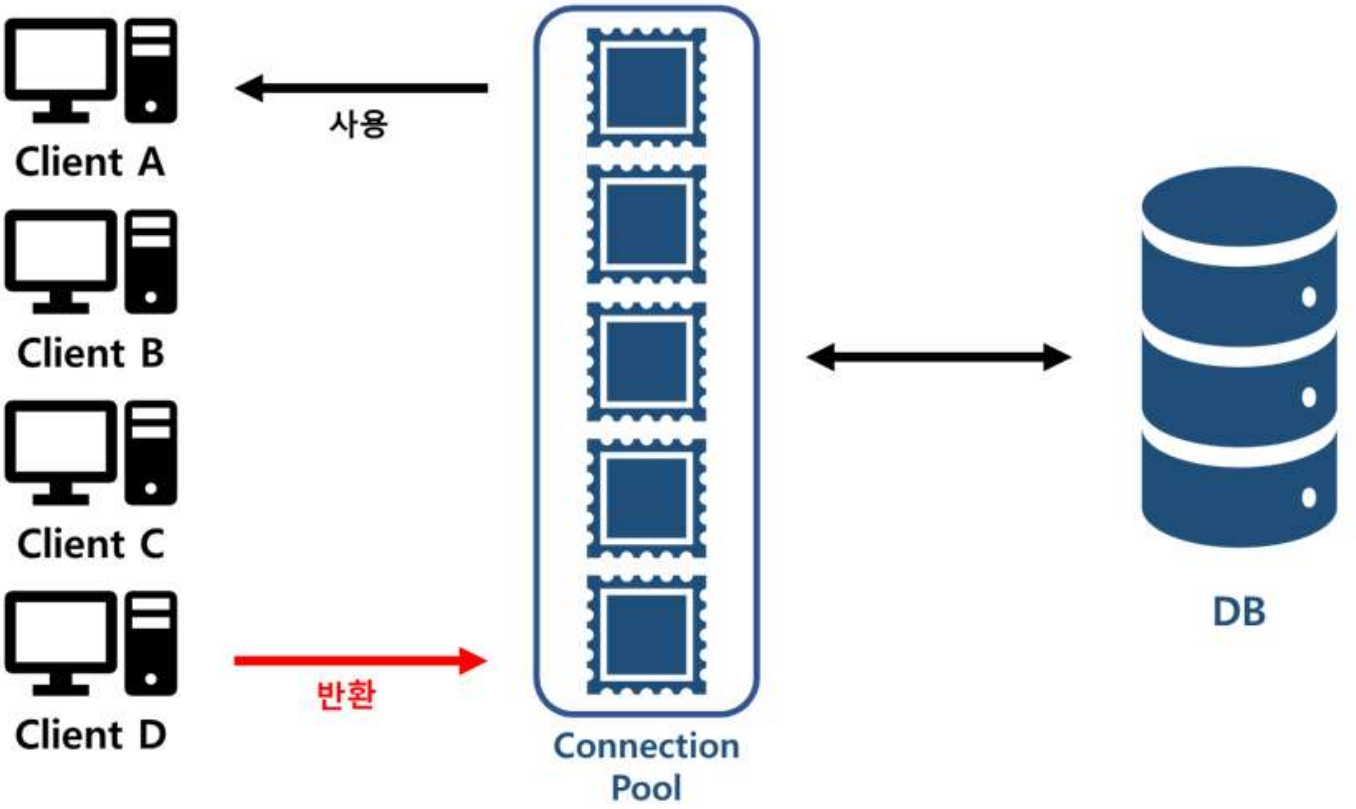
기존에 DB와 Connection하는 방식은 DB에 쿼리문을 보낼 때마다 Connection을 하고 결과를 받아왔음
하지만 이런 방식은 아래에 문제점이 있다.

문제점

여러 클라이언트로부터 동시 요청 수가 많아져 DB의 수용범위를 벗어날 때



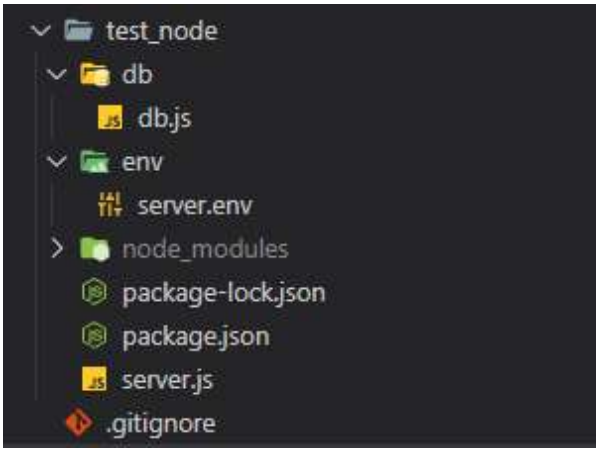
Connection Pool 방식



DB에 데이터를 가져오기 위해 쿼리문을 보낼 때마다 Connection을 생성하는 것이 비효율적이기 때문에 Connection Pool방식이 등장했다.
Connection Pool은 하나의 Pool을 생성한 뒤 지정한 수 만큼 Connctiond을 미리 할당을 한다.
그리고 Client에서 DB에 데이터를 가져오기위한 Connection을 Pool에서 가져와 사용한 뒤
다시 Pool에 Connection을 반환한다.
이렇게 Connection Pool 방식을 사용한다면 위에 문제상황처럼
갑자기 동시접속자 수가 많아져도 에러를 발생시키지 않고
Pool에 남아있는 Connection이 없다면 대기상태가 되고
다른 클라이언트에서 Connection이 반환되면 대기상태가 풀리게 된다

Nodejs에서 Mysql Connection Pool 사용

파일구성



DB Connection Pool 설정

db.js

```
const mysql = require('mysql');
const path = require('path')// 파일 경로 모듈
require('dotenv').config({ path: path.join(__dirname, './env/server.env') });//env 로드 모듈

const connection = { //database 접속 정보 입력
   host: process.env.DB_HOST,
   user: process.env.DB_USER,
   password: process.env.DB_PASS,
   port: process.env.DB_PORT,
   database: process.env.DB_DATABASE,
   connectionLimit : 30 // 커넥션수 30개로 설정
};

module.exports = mysql.createPool(connection);
```

DB 설정은 env파일 환경변수로 설정하고
Connection은 createPool을 module로 export해서 싱글톤 패턴으로 사용했다

Server Code

server.js

```
const express = require('express');
const path = require('path')// 파일 경로 모듈
require('dotenv').config({ path: path.join(__dirname, './env/server.env') });//env 로드 모듈
const mysql = require('./db/db'); // 싱글톤 패턴

const app = express();
const port = process.env.PORT || 5000;

pp.get("/connection_pool/test",(req, res)=>{
   const sql = "SELECT user_email, user_name FROM user";
   try {
       mysql.getConnection((err, connection)=>{ // Connection 연결
           console.log("connection_pool GET");
           if(err) throw err;

           connection.query(sql, (err, result, fields)=>{ // Query문 전송
               if(err) {
                   console.error("connection_pool GET Error / "+err);
                   res.status(500).send("message : Internal Server Error");
               }
               else {
                   if(result.length === 0){
                       res.status(400).send({
                           success : false,
                           message : "DB response Not Found"
                       });
                   }
                   else{
                       res.status(200).send({
                           success : true,
                           result
                        });
                   }
               }
           });
```

```
            connection.release(); // Connectino Pool 반환
        });
    } catch (err) {
        console.error("connection_pool GET Error / "+err);
        res.status(500).send("message : Internal Server Error");
    }
});

.listen(port, () => console.log(`Server Start Listening on port ${port}`));
```

싱글톤 패턴으로 만든 DB Connection Pool설정을 모듈로 가져와서
DB에 쿼리문을 보낼 때 getConnection으로 Pool에서 Connection을 가져온다음 query로 쿼리문을 보내고 getConnection 안에서 release()를 통해 connection을 반환하면 된다.

소스코드

https://github.com/gwon713/nodejsDB_connection_pool

참고사이트

https://beaniejoy.tistory.com/24

https://velog.io/@gwon713
권민제
성장하는 개발자!