Interface Test: MySQL(데이터베이스 Sequelize 라이브러리 연동 검증)

1. 데이터베이스 스키마 생성

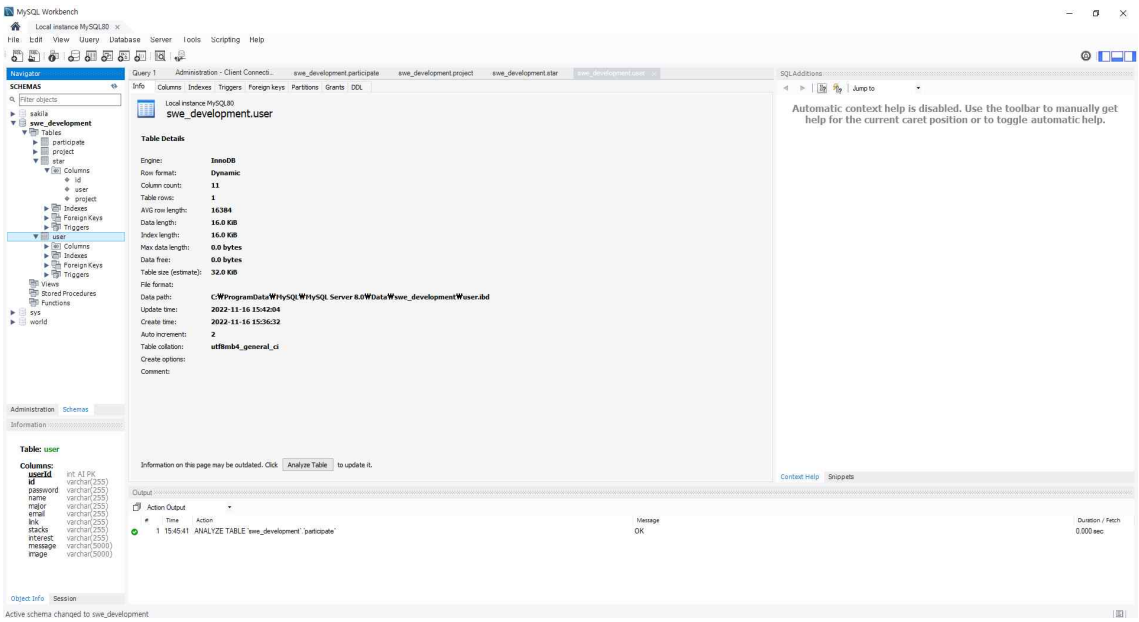
1.1) 사용할 데이터베이스가 처음부터 빈 경우일 때

본 시스템 백엔드 프로그램이 작동하려면 MySQL 프로그램을 설치해 로컬 환경에서 데이터베이스 서버와 연동이 되어야 하는데, 이때 로컬 데이터베이스 서버에는 backend/config 경로의 config.json 파일에 명시한 데이터베이스가 생성되어 있어야 한다. 다음 테스트는 데이터베이스 관련 기본 요건이 충족되었을 때, 백엔드 프로그램 실행 후 콘솔창의 모습이다. 다음과 같이 정상적으로 데이터베이스와 연결이 되는 것을 볼 수 있다.

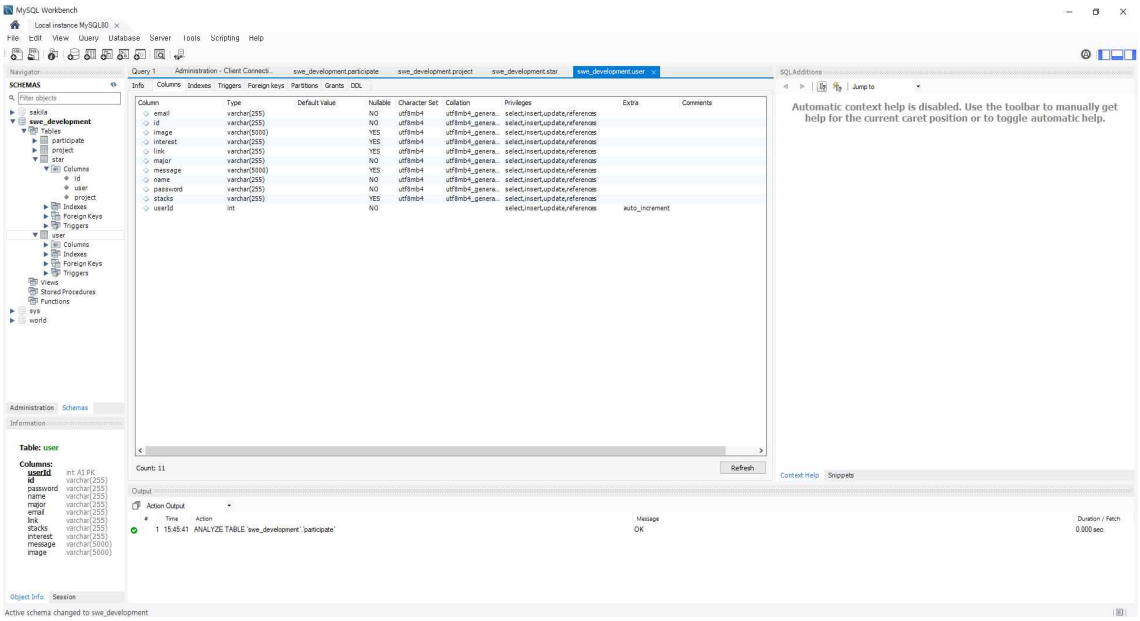
```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
backend server started at port 8001
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'user' AND TABLE_SCHEMA = 'swe_development'
Executing (default): SHOW INDEX FROM `user` FROM `swe_development`
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'project' AND TABLE_SCHEMA = 'swe_development'
Executing (default): SHOW INDEX FROM `project` FROM `swe_development`
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'star' AND TABLE_SCHEMA = 'swe_development'
Executing (default): SHOW INDEX FROM `star` FROM `swe_development`
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'participate' AND TABLE_SCHEMA = 'swe_development'
Executing (default): SHOW INDEX FROM `participate` FROM `swe_development`
database connect
```

[그림 1] 데이터베이스 스키마 정상적으로 연동된 모습

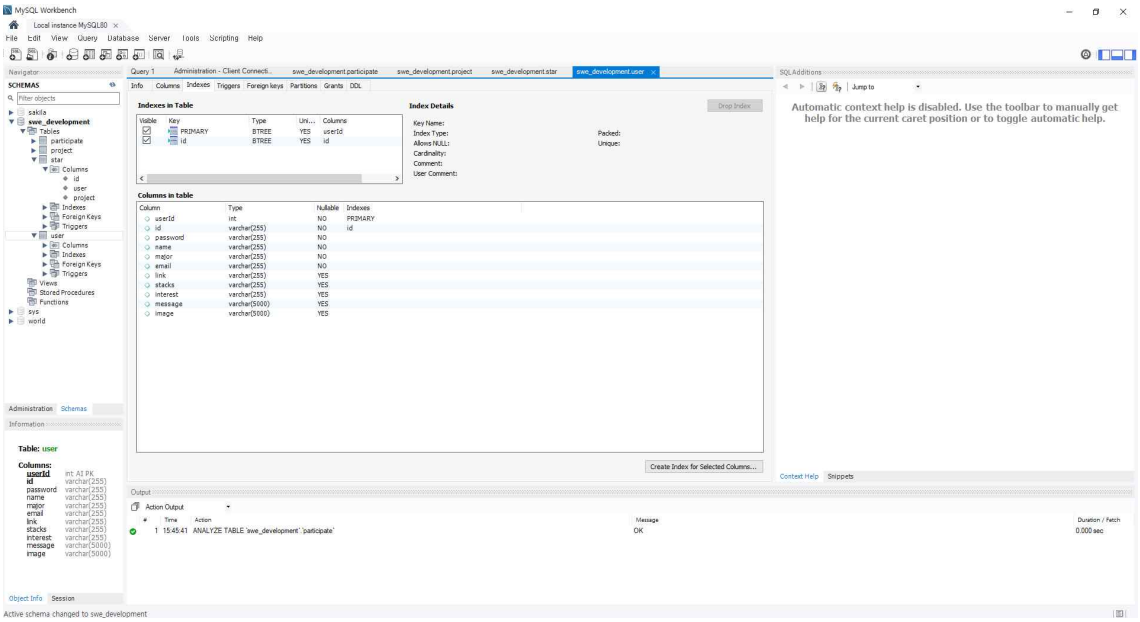
아래 사진들은 실제로 데이터베이스가 연동된 것을 MySQL Workbench 프로그램을 통해서 확인한 사진이다.



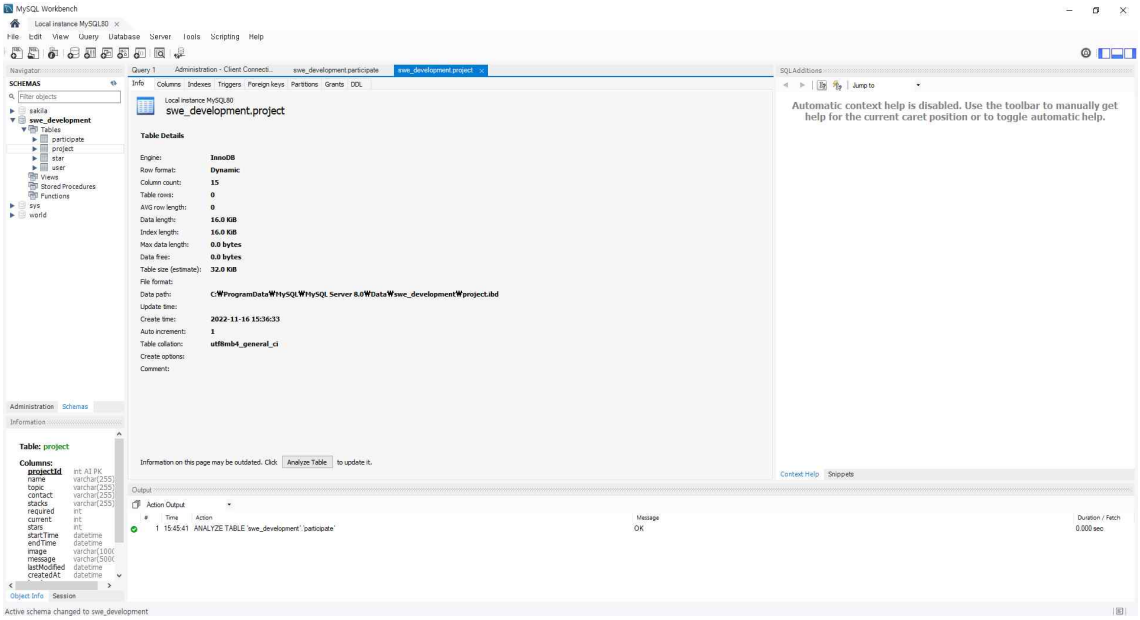
[그림 2] swe_devlopment 데이터베이스의 user 테이블 정보



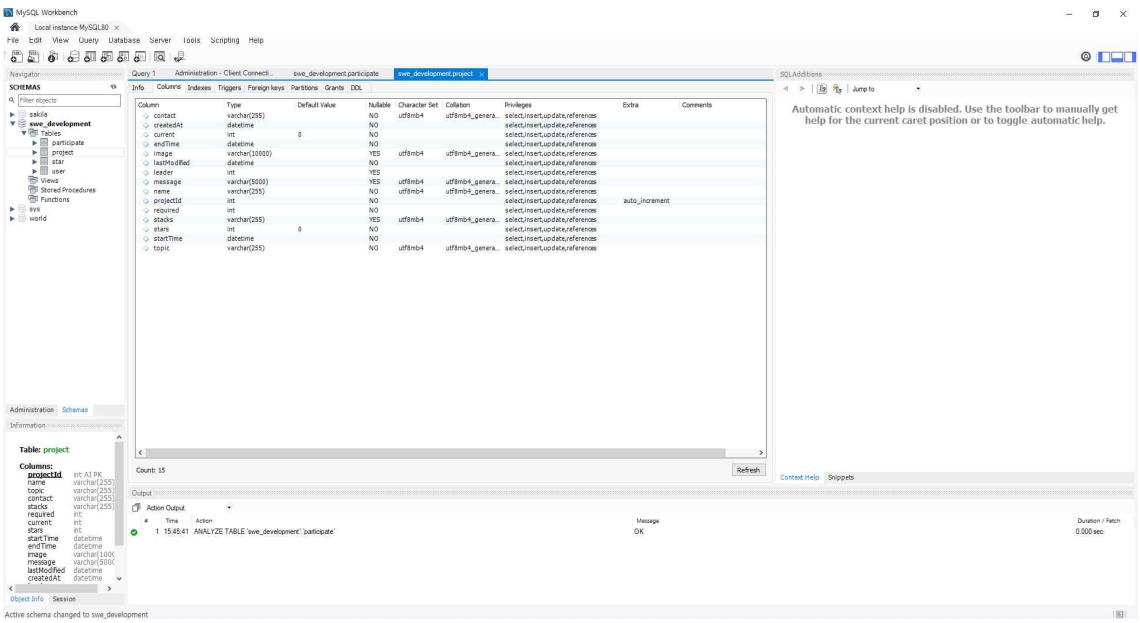
[그림 3] swe_devlopment 데이터베이스의 user 테이블 Column



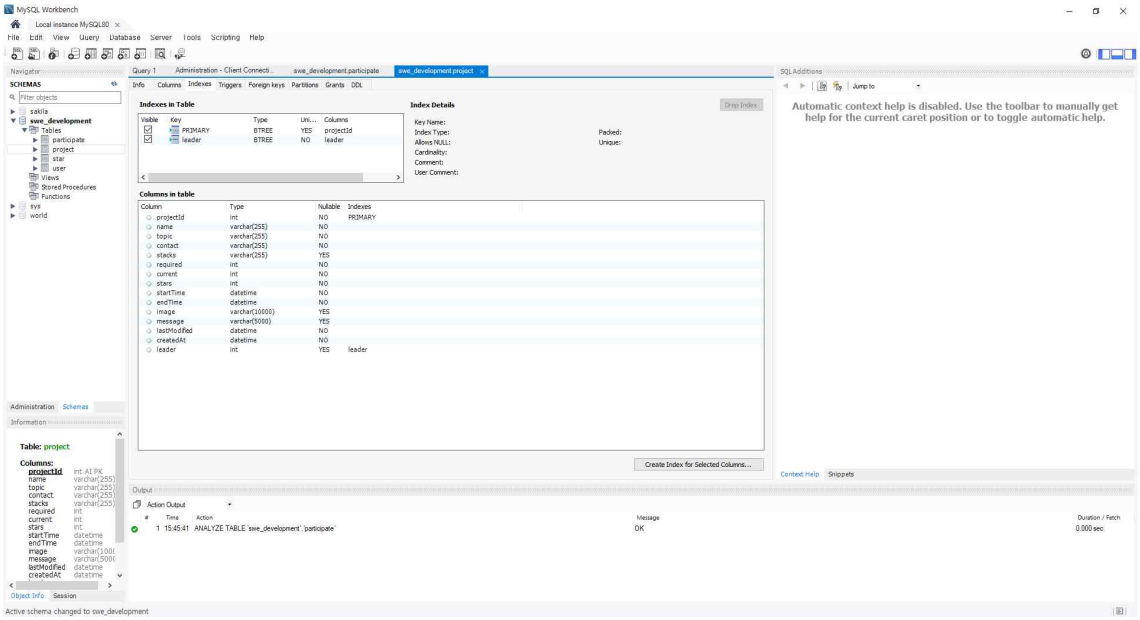
[그림 4] swe_devlopment 데이터베이스의 user 테이블 Index



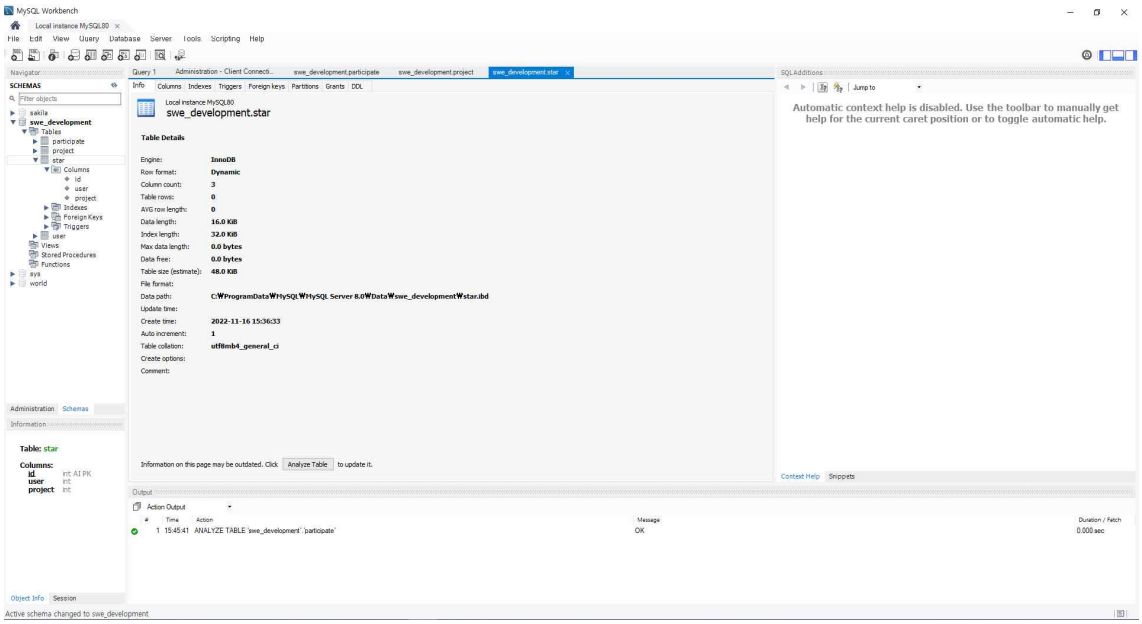
[그림 5] swe_development 데이터베이스 project 테이블 정보



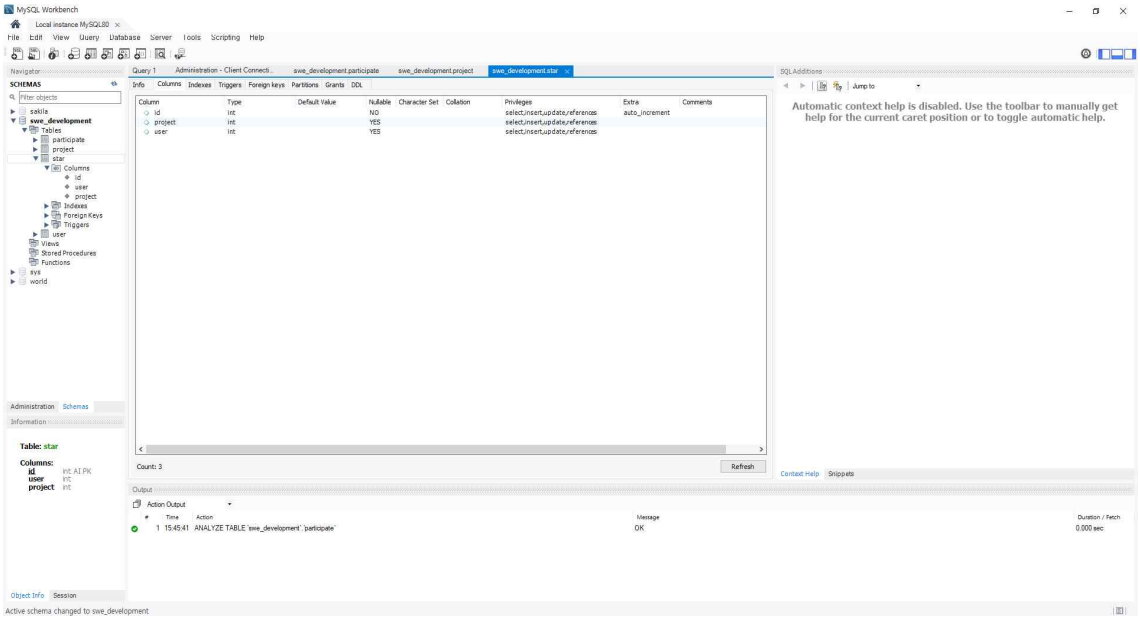
[그림 6] swe_development 데이터베이스 project 테이블 Column



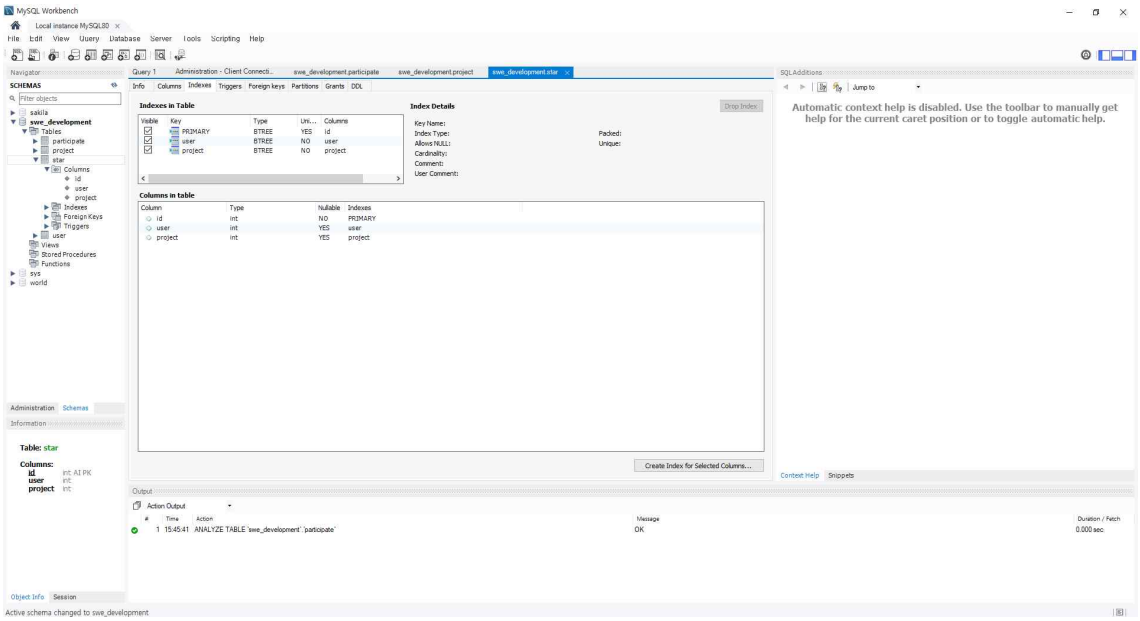
[그림 7] swe_development 데이터베이스 project 테이블 Index



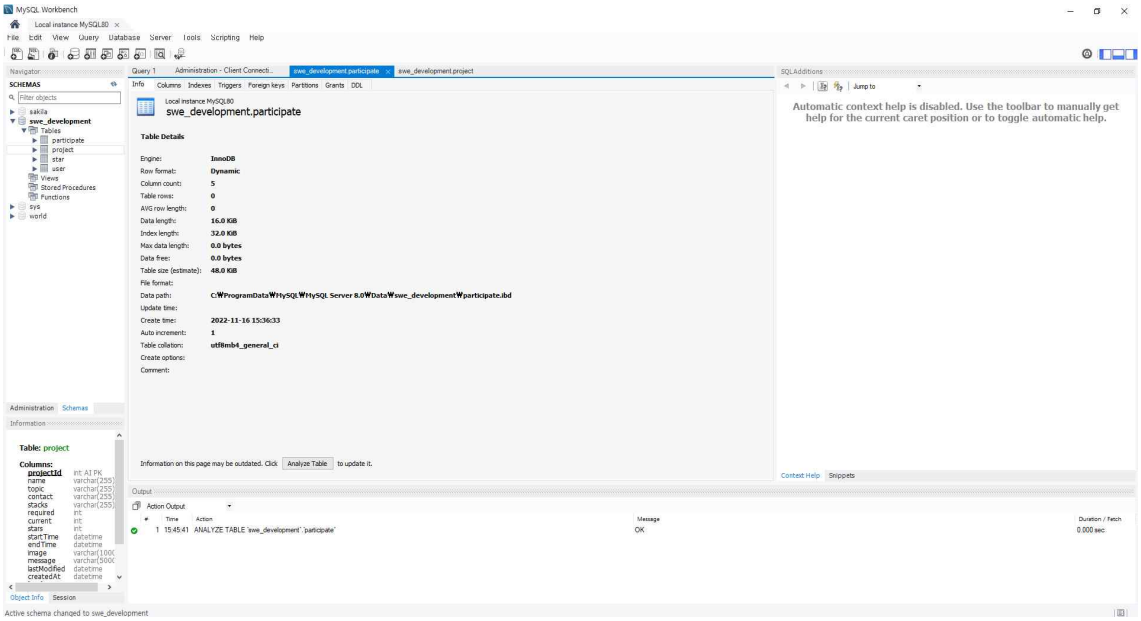
[그림 8] swe_development 데이터베이스 star 테이블 정보



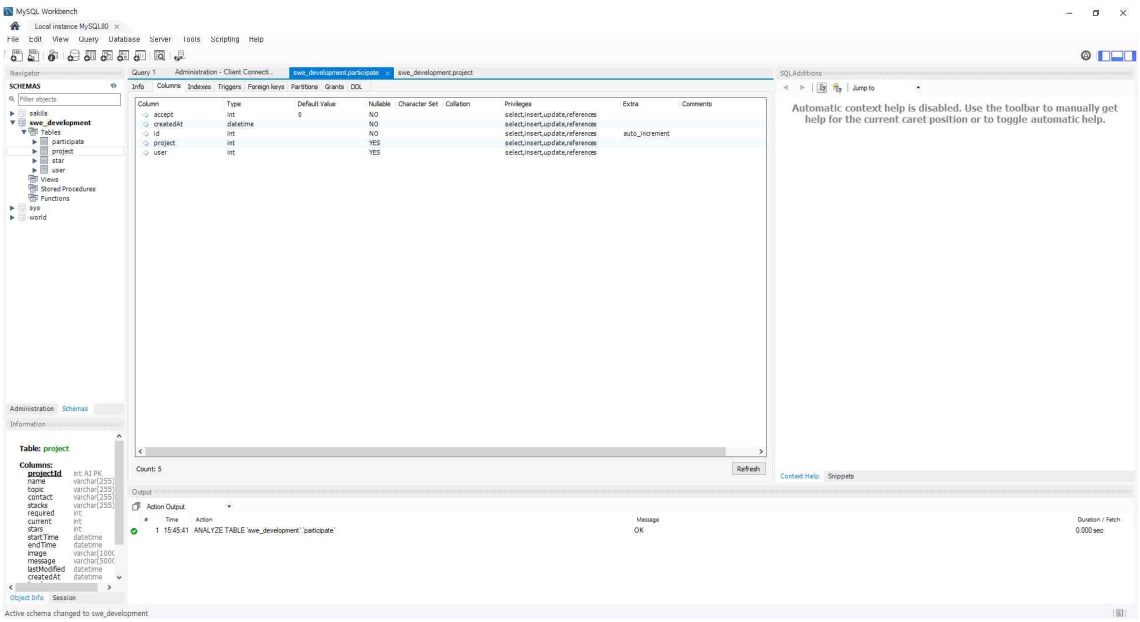
[그림 9] swe_development 데이터베이스 star 테이블 Column



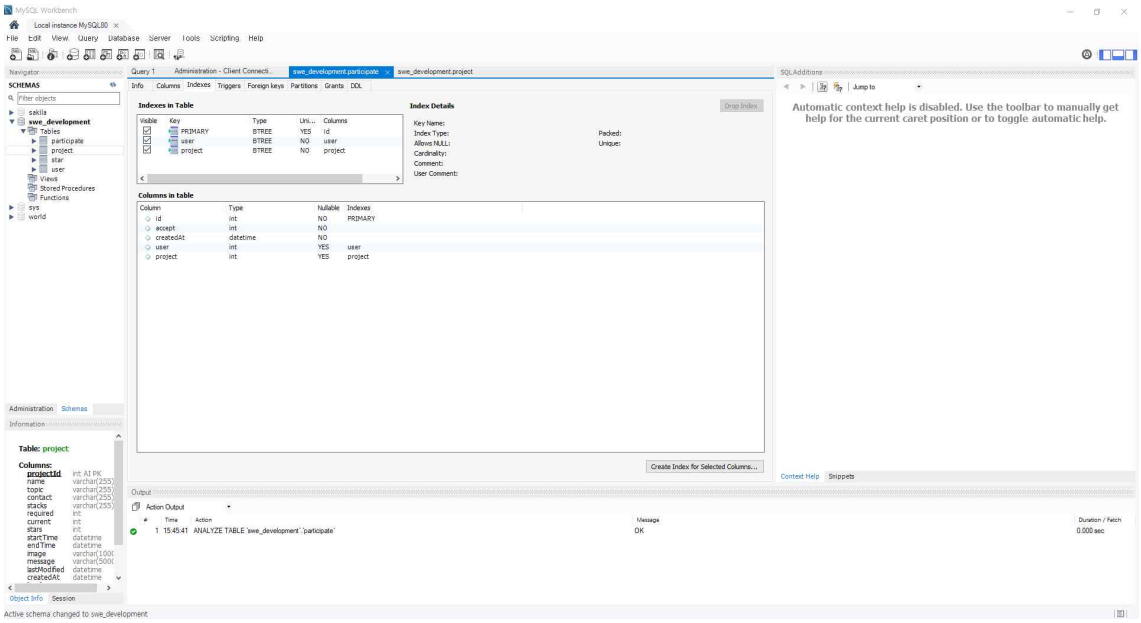
[그림 10] swe_development 데이터베이스 star 테이블 Index



[그림 11] swe_development 데이터베이스 participate 테이블 정보



[그림 12] swe_development 데이터베이스 participate 테이블 Column



[그림 13] swe_development 데이터베이스 particiapte 테이블 Index

1.2) 사용할 데이터베이스가 생성되지 않은 상태로 실행할 때

본 시스템 백엔드 프로그램 실행 전 MySQL 데이터베이스에 어플리케이션 코드 상대 경로로 backend/config/config.json에 명시된 데이터베이스 이름으로 데이터베이스가 생성되어 있어야 한다. 해당 테스트는 실행 기본 조건을 만족하지 않은 상황의 테스트를 위해서 진행했다. 실행결과는 다음과 같이 어플리케이션이 시작되지 않고 오류를 띄우는 경우를 볼 수 있다.

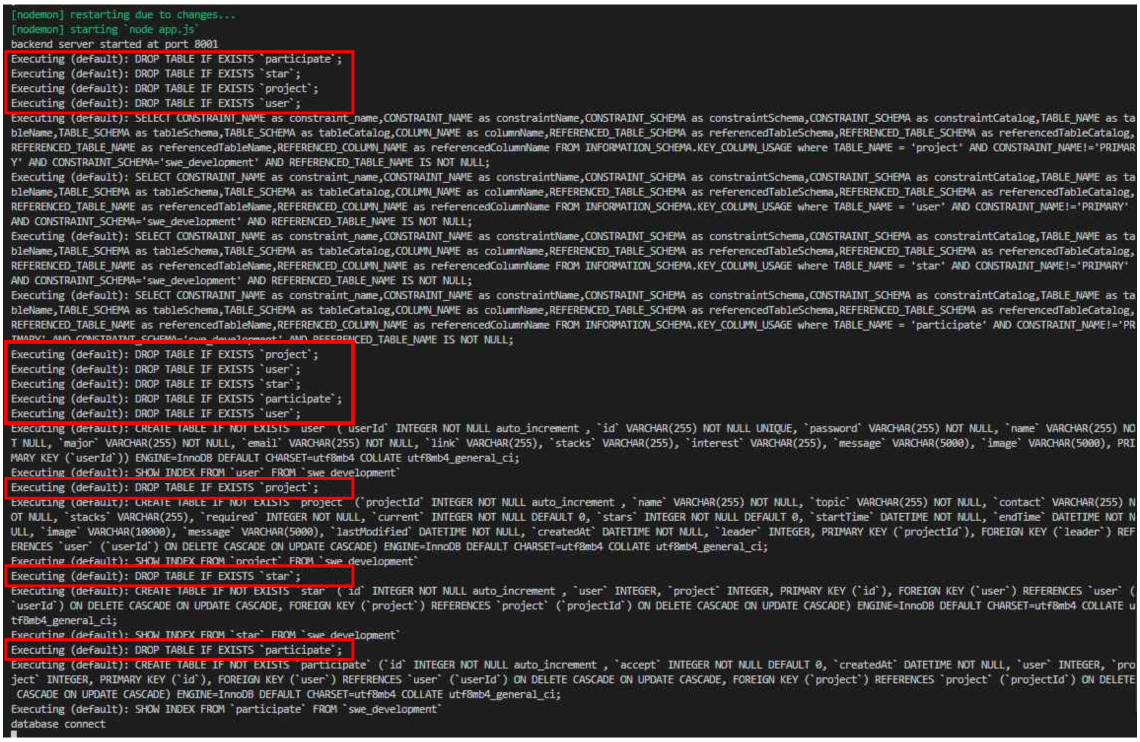
```
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
backend server started at port 8001
ConnectionError [SequelizeConnectionError]: Unknown database 'swe_development'
    at ConnectionManager.connect (C:\Users\안현준\Desktop\swe\2022fall_42class_team12\backend\node_modules\sequelize\lib\dialects\mysql\connection-manager.js:102:17)
    at processTicksAndRejections (node:internal/process/task_queues:96:5)
    at async ConnectionManager._connect (C:\Users\안현준\Desktop\swe\2022fall_42class_team12\backend\node_modules\sequelize\lib\dialects\abstract\connection-manager.js:220:24)
    at async C:\Users\안현준\Desktop\swe\2022fall_42class_team12\backend\node_modules\sequelize\lib\dialects\abstract\connection-manager.js:174:32
    at async ConnectionManager.getConnection (C:\Users\안현준\Desktop\swe\2022fall_42class_team12\backend\node_modules\sequelize\lib\dialects\abstract\connection-manager.js:197:7)
    at async C:\Users\안현준\Desktop\swe\2022fall_42class_team12\backend\node_modules\sequelize\lib\sequelize.js:304:26
    at async MySQLQueryInterface.tableExists (C:\Users\안현준\Desktop\swe\2022fall_42class_team12\backend\node_modules\sequelize\lib\dialects\abstract\query-interface.js:102:17)
    at async Function.sync (C:\Users\안현준\Desktop\swe\2022fall_42class_team12\backend\node_modules\sequelize\lib\model.js:939:21)
    at async Sequelize.sync (C:\Users\안현준\Desktop\swe\2022fall_42class_team12\backend\node_modules\sequelize\lib\sequelize.js:376:9) {
  parent: Error: Unknown database 'swe_development'
      at Packet.asError (C:\Users\안현준\Desktop\swe\2022fall_42class_team12\backend\node_modules\mysql2\lib\packets\packet.js:728:17)
      at ClientHandshake.execute (C:\Users\안현준\Desktop\swe\2022fall_42class_team12\backend\node_modules\mysql2\lib\commands\command.js:29:26)
      at Connection.handlePacket (C:\Users\안현준\Desktop\swe\2022fall_42class_team12\backend\node_modules\mysql2\lib\connection.js:456:32)
      at PacketParser.onPacket (C:\Users\안현준\Desktop\swe\2022fall_42class_team12\backend\node_modules\mysql2\lib\connection.js:85:12)
      at PacketParser.executeStart (C:\Users\안현준\Desktop\swe\2022fall_42class_team12\backend\node_modules\mysql2\lib\packet_parser.js:75:16)
      at Socket.<anonymous> (C:\Users\안현준\Desktop\swe\2022fall_42class_team12\backend\node_modules\mysql2\lib\connection.js:92:25)
      at Socket.emit (node:events:527:28)
      at Socket.emit (node:domain:475:12)
      at addChunk (node:internal/streams/readable:315:12)
      at readableAddChunk (node:internal/streams/readable:289:9) {
    code: 'ER_BAD_DB_ERROR',
    errno: 1049,
    sqlState: '42000',
    sqlMessage: "Unknown database 'swe_development'",
    sql: undefined
  },
  original: Error: Unknown database 'swe_development'
```

[그림 14] swe_development 데이터베이스 미생성 상태로 백엔드 어플리케이션 실행 결과

1.3) 이전에 사용한 데이터베이스가 생성되어 있는 경우

코드에 backend/app.js 경로의 파일을 보면 데이터베이스 연동하는 부분에서 force 옵션이 true로 된 상태로 테스트를 진행한다. 실제 배포시에는 이 부분을 false로 바꾸어 진행해야

한다. 이유는 force 옵션 값이 true인 상태로 데이터베이스 연동을 시작하게 되면 기존 데이터베이스의 내용이 전부 삭제되는 것을 볼 수가 있다. 아래 사진은 백엔드 프로그램 실행 시 force 옵션을 true로 놓고 실행한 모습이다. 결과 이미지에 빨간색으로 표시한 부분이 테이블이 존재할 경우 테이블을 삭제하는 부분인 것을 알 수 있다. 이와 같은 경우는 테스트 단계에서 데이터베이스 스키마 변화가 정확히 반영되는지를 확인하기 위해서 진행했다.

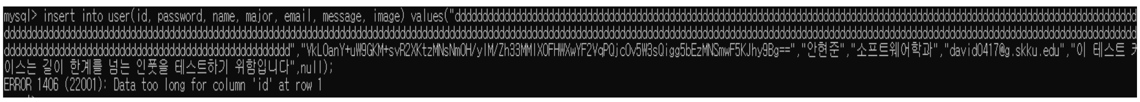


[그림 15] 기존에 데이터베이스가 사용하던 데이터베이스가 있는 상태에서 실행 결과

2. MySQL 데이터베이스 정보 삽입 테스트

2.1) 삽입된 정보가 제한 크기 초과하는 경우

해당 테스트는 본 시스템의 잠재적인 오류를 찾아내기 위해서 실시했다. 본 시스템 어플리케이션 코드는 입력값으로 받는 값에 제한을 두는 형태로 삽입된 정보가 제한 크기를 넘는 경우를 예방했다. 본 테스트 케이스는 시스템의 외부 프로그램과 연결간에 발생할 수 있는 오류를 검증하기 위해 실시된 테스트이다. 본 어플리케이션으로는 이에 해당하는 경우를 발생시킬 수 없어 실제로 직접 MySQL 프로그램에 오류를 발생할 입력값을 주어서 실시했다. 테스트 결과는 아래 그림과 같다.



[그림 16] 입력된 삽입 정보가 스키마의 제한 크기를 초과하는 경우 오류 발생

2.2) 정상적인 경우

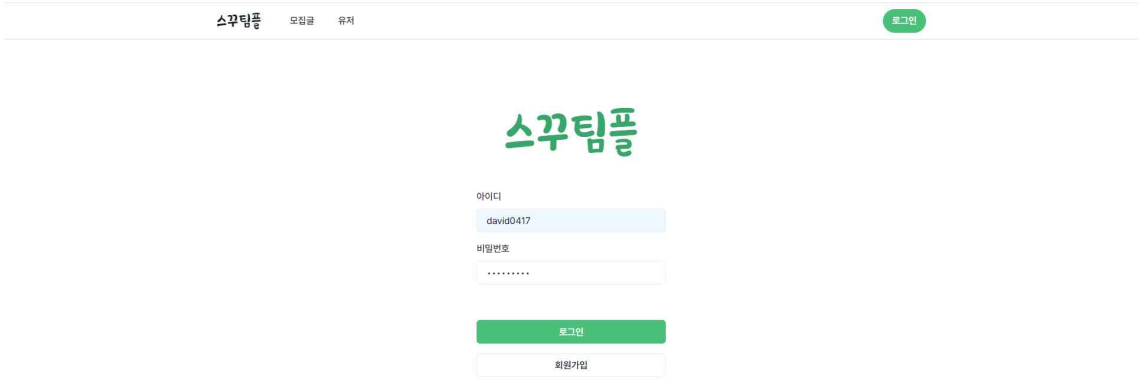
본 테스트 케이스는 위의 테스트 케이스와는 다르게 정상적인 범위에서 입력값을 넣었을 때 성공적으로 삽입이 이루어지는 경우를 보이기 위한 테스트 케이스이다.

[그림 17] 입력된 삽입 정보가 스키마의 제한 크기를 만족하는 정상 경우

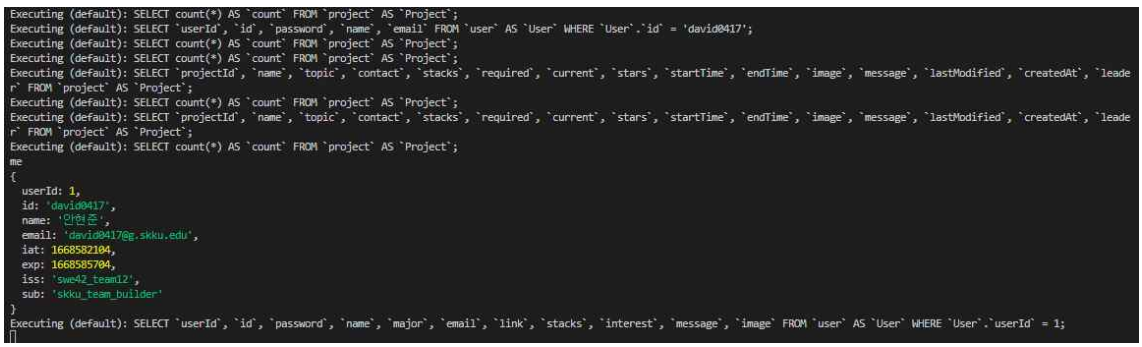
3. MySQL 데이터베이스 정보 조회 테스트

3.1) 조건을 충족하는 정보가 있을 경우

본 테스트는 데이터베이스와 구현 어플리케이션 사이 연동을 검증할 목적으로 MySQL과 Sequelize 라이브러리 연동이 정상적으로 이루어지는지 검증하기 위한 테스트이다. 테스트는 본 어플리케이션의 로그인 기능을 통해서 로그인 정보가 있는 경우와 없는 경우 어떻게 진행되는지와 실제로 직접 MySQL을 통해서 정보 조회를 하는 방식으로 진행했다.



[그림 18] 기존에 생성한 아이디 계정으로 로그인

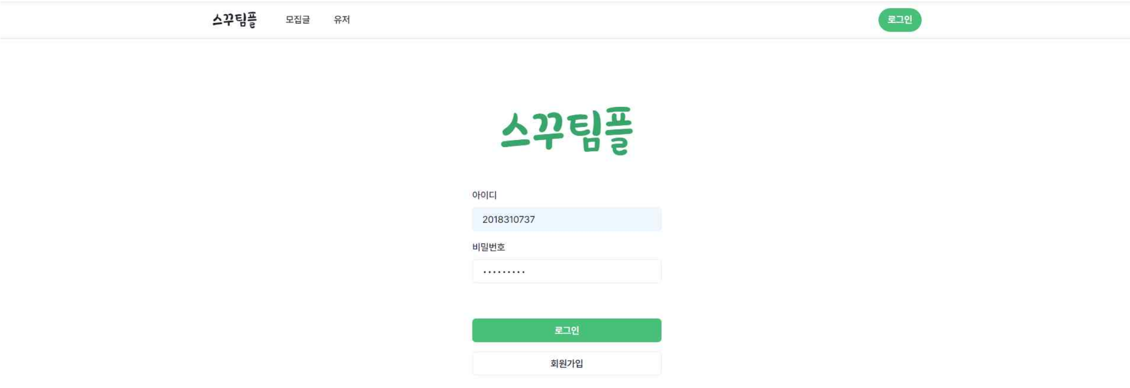


[그림 19] 기존에 생성한 아이디 계정으로 로그인 시도 후 백엔드 어플리케이션 콘솔창

3.2) 조건을 충족하는 정보가 없을 경우

다음 테스트는 위의 테스트와 반대로 없는 정보에 대해서 데이터베이스 내 레코드를 조회하는 기능의 결과를 보고자 실시한 테스트 케이스이다. 본 시스템의 프론트엔드 부분과 백엔드 부분을 같이 연결하여 프론트엔드에서 지시한 기능이 백엔드를 통해서 데이터베이스까지 연동

되는 데 문제가 없는지 보기 위해 로그인 기능을 이용했다. 실행결과 아무런 데이터도 조회되지 않음을 알 수 있다. 추가로 직접 MySQL에서 있는 정보랑 없는 정보를 조회하면 나오는 결과도 테스트했다.



[그림 20] 기존에 생성하지 않은 계정으로 로그인 시도

```
Executing (default): SELECT `userId`, `id`, `password`, `name`, `email` FROM `user` AS `User` WHERE `User`.`id` = '2018310737';
Executing (default): SELECT `userId`, `id`, `password`, `name`, `email` FROM `user` AS `User` WHERE `User`.`id` = '2018310737';
```

[그림 21] 기존에 생성하지 않은 계정으로 로그인 시도 시 백엔드 어플리케이션 콘솔창 결과

```
mysql> select * from user where id="david0417";
+--------+-----------+----------------------------------------------------------------------------+--------+--------------+-----------------------+---
| userId | id        | password                                                                   | name   | major        | email                 | lin
  | image                                                                                       |
+--------+-----------+----------------------------------------------------------------------------+--------+--------------+-----------------------+---
|      1 | david0417 | VkL0anY+uW9GKM+svR2XKtzMNsNm0H/yIM/Zh33MMIX0FHWXwYF2VqPQjc0v5W3sQigg5bEzMNSmwF5KJhy9Bg== | 안현준 | 소프트웨어학과 | david0417@g.skku.edu | NUL
야 | https://2022fall-42class-team12.s3.ap-northeast-2.amazonaws.com/profiles/1668581888769Sharoz.png |
+--------+-----------+----------------------------------------------------------------------------+--------+--------------+-----------------------+---
1 row in set (0.00 sec)

mysql> select * from user where id="2018310737";
Empty set (0.00 sec)

mysql>
```

[그림 22] 위 쿼리 결과는 있는 정보를 조회 결과이고, 아래 쿼리 결과는 없는 정보를 조회 결과

4. MySQL 데이터베이스 정보 수정

4.1) 수정할 정보가 제한 데이터 크기 초과하는 경우

본 테스트 케이스는 수정되는 정보가 스키마에서 설정한 데이터 레코드 값 범위를 벗어나가는 경우를 검증하기 위해서 테스트를 진행했다. 본 테스트 케이스들은 2장 MySQL 데이터베이스 정보 삽입 부분에서 다룬 테스트와 유사한 기능을 테스트하기 위한 목적으로 진행했다. 테스트 결과는 다음 이미지와 같다.

```
mysql> update user
    -> set id="dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd"
    -> where id="hyunjun0417";
ERROR 1406 (22001): Data too long for column 'id' at row 1
mysql>
```

[그림 23] 수정한 데이터의 크기가 스키마에서 제한한 크기보다 큰 경우 오류 발생

4.2) 수정할 정보가 정상적인 데이터 크기인 경우

본 테스트 케이스는 위의 4.1 테스트와는 다르게 정상적인 데이터 크기로 데이터가 수정될 경우에 대해서 테스트한 결과이다.

[그림 24] 수정한 데이터의 크기가 스키마에서 제한한 크기를 만족하는 정상적인 경우