

ver.2021 노베이스 모던 웹개발

- 6. GraphQL 과 PostgreSQL 연동

작성자: 경상국립대학교 컴퓨터과학과 조교 이자룡

목차

p.2	-	0. Orientation
p.3	-	1. 샘플 DB 만들기
p.6	-	2. PostGraphile 설치
p.9	-	3. 쿼리(query)
p.10	-	포트 끄고 키는 법
p.13	-	QUERY CASE 1: 한 명의 유저 데이터를 가져올 경우
p.16	-	QUERY CASE 2: 모든 유저 데이터를 가져올 경우
p.19	-	4. 뮤테이션(mutation)
p.19	-	CREATE
p.20	-	UPDATE
p.20	-	DELETE
p.21	-	실습
p.27	-	5. 마치며

0. Orientation

* 이 수업을 시작하기 전에 반드시, JavaScript, TypeScript, React Hook, SQL 기초를 알고 시작하길 바란다. 《ver.2021 노베이스 모던 웹개발》 첫번째 강좌인 『React 시작 전에 알아야 할 JavaScript, TypeScript』, 두번째 강좌인 『React Hook』, 다섯번째 강좌인 『GraphQL을 위한 PostgreSQL』이 준비되어 있다.

이 수업에선 뭘 배울것인가? 실컷 DB 만들었으면 사용해야한다. React에서 PostgreSQL에 접근해 CRUD 하는 방법은 여러가지가 있는데, 우리 GraphQL을 사용해볼 것이다. 단, 이 수업에선 GraphQL의 핵심인 query와 mutation만 배워본다. 깊은 지식이 필요한 사람은 《ver.2021 노베이스 모던 웹개발》 코스를 완주한 후, 다른 사람의 자료를 참고해서 공부해보길 바란다.

주의사항, GraphQL은 업계 표준 기술이 아니다. 이 강의에선 서버에 접근하는 표준 기술인 REST API에 대한 설명을 하지 않는다. 우리 쉬운 웹앱 만들기를 추구하기 때문이다. REST API 방식으로 서버에 접근하고 싶다면 Python 풀스택 프레임워크인 Django를 배워보길 추천한다.

GraphQL은 2015년에 탄생했다. 라이브러리가 아니라, ‘어딘가’에서 데이터를 가져오는 ‘방식’이다. 여기서 말하는 ‘어딘가’란 다음과 같은 것들이다.

1. API
2. SQL Database
3. NoSQL Database

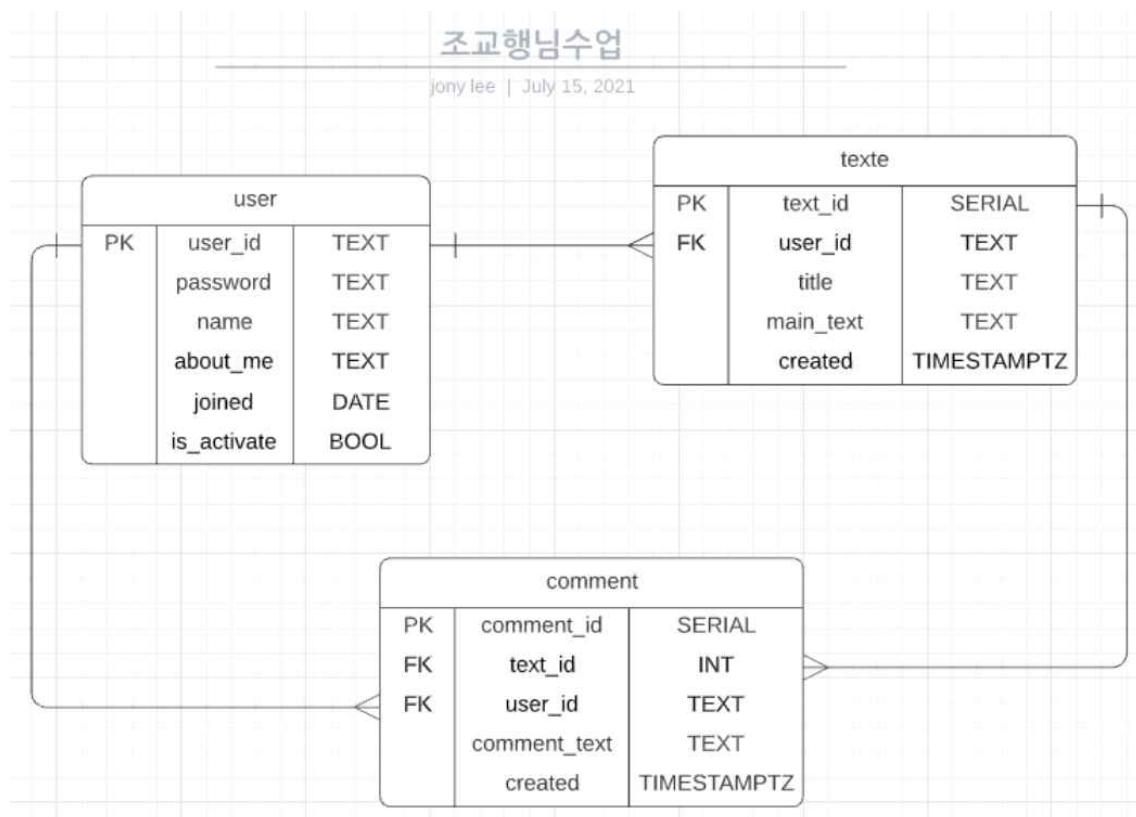
즉, GraphQL은 우리가 배워본 PostgreSQL에만 접근할 수 있는 기술이 아니다. 우리가 상상할 수 있는 거의 모든 데이터에 접근 가능하다. 그러나 라이브러리가 아닌 하나의 ‘방식’이기 때문에, 개발하기 위해선 GraphQL을 편하게 쓸 수 있도록 만들어진 라이브러리가 필요하다.

GraphQL 세계에서의 표준은 Apollo GraphQL이라는 회사다. 대표적인 라이브러리는 두가지가 있는데, 하나는 Apollo Client, 다른 하나는 Apollo Server이다. 그러나 Apollo Server는 데이터베이스 접근을 공식적으로 지원하지 않는다. 무슨 말이냐, GraphQL은 모든 데이터에 접근할 수 있지만, SQL DB를 위한 특별한 기술을 따로 제공하진 않는다는 것이다. 대신, 공식문서에선 이 문제를 해결하기 위한 다른 라이브러리(Knex를 사용하는 SQLDataSource)를 추천한다. 이 문제 때문에, 우리 Apollo Server는 사용하지 않고, React에서 사용할 수 있는 Apollo Client만 써보도록 하겠다.

서버 측 GraphQL 을 SQL DB 와 연결하기 위해 PostGraphile 이라는 라이브러리를 사용해볼 것이다. 실제 사용해본 경험으로선, 이것을 실행시키기 위한 명령어만 어려울 뿐이지 사용 자체는 매우 쉽다. 기본적인 CRUD 함수를 자동으로 만들어주고, 고급 사용자는 직접 서버 함수를 만들수도 있다. 게다가 적당히 인기 있는 라이브러리인데, 개발자들의 전세계적인 코드 저장소인 GitHub 에선 10.5k 의 스타를 받았다. 단점도 존재한다. 표준기술이 아니기에 내년에 사라져도 이상할 것이 없는 라이브러리이며, PostgreSQL 을 쓰지 않고 다른 SQL DB를 사용한다면 아무짝에 쓸모없는 라이브러리라는 것이다. 어쨌든, 우리 쉽게 웹앱 만드는 게 목적인 사람들이기에, 쉬운 기술이라면 한번 사용해보자.

다시 강조하지만, GraphQL은 표준 기술이 아니다. REST API 를 배워보고 싶은 사람은 Django 를 배워보길 추천한다. GraphQL을 심도깊게 배워보고 싶은 사람은, PostGraphile 이 아닌 Apollo Backend 를 배워보길 바란다.

1. 샘플 DB 만들기



먼저, 다음 ERD 에 해당하는 샘플 데이터베이스를 만들 계획이다. 가장 좋은 방법은, 여러분 스스로 지금까지 배웠던 지식을 활용해 직접 만들어보는 것이다. DB 수업 마지막에도 그걸 강조했었다. 직접 해보는 게 중요하다.

그러나, 귀찮은 사람들을 위해서 코드를 제공한다.

```

```
CREATE TABLE "user" (
 user_id TEXT NOT NULL PRIMARY KEY,
 password TEXT NOT NULL,
 name TEXT NOT NULL,
 about_me TEXT,
 joined DATE NOT NULL DEFAULT CURRENT_DATE,
 is_activate BOOL NOT NULL DEFAULT TRUE
);
```

```
CREATE TABLE texte (
 text_id SERIAL NOT NULL PRIMARY KEY,
 user_id TEXT NOT NULL,
 title TEXT NOT NULL,
 main_text TEXT NOT NULL,
 created TIMESTAMPTZ DEFAULT now()
);
```

```
CREATE TABLE comment (
 comment_id SERIAL NOT NULL PRIMARY KEY,
 text_id INT NOT NULL,
 user_id TEXT NOT NULL,
 comment_text TEXT NOT NULL,
 created TIMESTAMPTZ DEFAULT now()
);
```

```
INSERT INTO "user" (
 user_id, password, name, about_me, joined
) VALUES (
 'assistant0603',
 '123123',
 '조교행님',
 '안녕하세요 조교행님입니다.',
 DATE '2013-04-13'
);
```

```
INSERT INTO "user" (
 user_id, password, name, about_me, joined
) VALUES (
 'sujini',
 '111111',
 '수지니',
 '리액트 배우고싶은 수지니예요',
 DATE '2013-04-14'
);
```

```
INSERT INTO "user" (
 user_id, password, name, about_me, joined
) VALUES (
 'joayo',
 'joayo123',
 '조아요맨',
 '조아요~~',
 DATE '2013-08-21'
);
```

```
INSERT INTO "user" (
 user_id, password, name, joined
) VALUES (
 'corini',
 '113366',
 '코리니',
 DATE '2015-12-10'
);
```

```

INSERT INTO texte (
 user_id, title, main_text
) VALUES (
 'assistant0603',
 '조교행님의 첫 글 ㅎㅎ',
 '안녕 애들아 난 조교 행님이야. 우리 같이 리액트 잘 배워보자!'
);

INSERT INTO texte (
 user_id, title, main_text
) VALUES (
 'corini',
 '리액트... 막막하네요',
 '어떻게 공부해야되는지 모르겠어요. 혹시 잘 작성되어있는 커리큘럼같은게 있나요?'
);

INSERT INTO comment (
 text_id, user_id, comment_text
) VALUES (
 '1',
 'sujini',
 '잘 부탁드립니다 ㅎㅎㅎ'
);

INSERT INTO comment (
 text_id, user_id, comment_text
) VALUES (
 '1',
 'joayo123',
 '조아요~~'
);

INSERT INTO comment (
 text_id, user_id, comment_text
) VALUES (
 '2',
 'corini',
 '아시는 분은 댓글 부탁드립니다~~'
);

INSERT INTO comment (
 text_id, user_id, comment_text
) VALUES (
 '2',
 'corini',
 'ㅠㅠㅠㅠㅠㅠ'
);

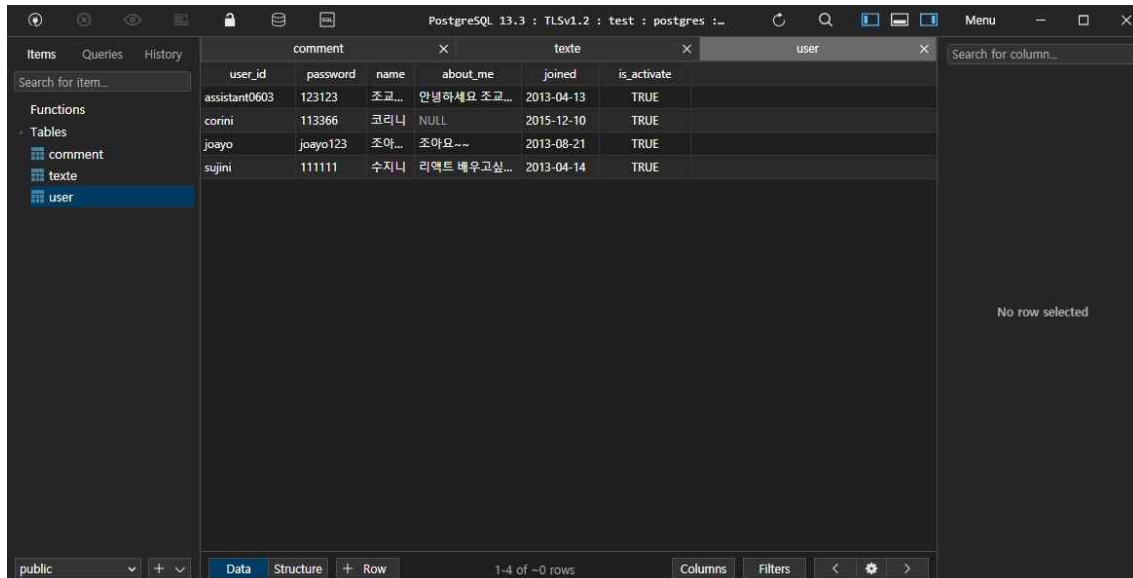
INSERT INTO comment (
 text_id, user_id, comment_text
) VALUES (
 '2',
 'sujini',
 '조교행님 강의는 어떤가요?'
);

INSERT INTO comment (
 text_id, user_id, comment_text
) VALUES (
 '2',
 'assistant0603',
 '제가 바로 조교행님입니다.'
);

...

```

다시 말하지만, 웬만하면 복사 붙여넣기 하지 말고 지금까지 배운것만 활용해 DB를 만들어보길 추천한다.

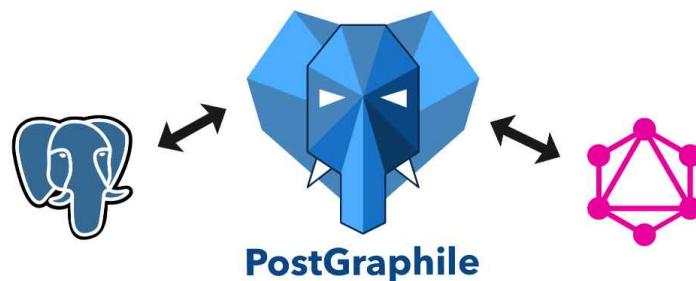


| user_id       | password | name  | about_me    | joined     | is_activate |
|---------------|----------|-------|-------------|------------|-------------|
| assistant0603 | 123123   | 조교... | 안녕하세요 조교... | 2013-04-13 | TRUE        |
| corini        | 113366   | 코리니   | NULL        | 2015-12-10 | TRUE        |
| joayo         | joayo123 | 조아... | 조아요~~~      | 2013-08-21 | TRUE        |
| sujini        | 111111   | 수지니   | 리액트 배우고싶... | 2013-04-14 | TRUE        |

TablePlus 접속 시 데이터가 잘 나오는것을 확인할 수 있다. 이번 수업에선 user 테이블만 사용해볼 것이다.

## 2. PostGraphile 설치

구글 검색 postgraphile



Instantly spin-up a GraphQL API server by pointing PostGraphile at your existing PostgreSQL database

이 수업은 웬만하면 공식문서대로 진행하려 했는데, PostGraphile 의 공식문서는

여러분의 서버 지식이 빈약하면 하나도 이해할 수 없을 정도로 매우 어렵다!  
그래서 안타깝지만, 다음 코드로 실행해보도록 하자.

```
$ npx postgraphile -c 'postgres://디비유저이름:디비유저비번@localhost/디비이름' --watch --enhance-graphiql --dynamic-json -o
```

유저이름은 postgres 이고, 비밀번호는 111111, db 이름은 sample\_for\_graphql 이라고 가정하면 다음과 같다.

```
$ npx postgraphile -c 'postgres://postgres:111111@localhost/sample_for_graphql' --watch --enhance-graphiql --dynamic-json -o
```

무슨 뜻인지 알아보면,

|     |                                                                            |
|-----|----------------------------------------------------------------------------|
| npx | npm 에서 제공하는, 설치를 하지 않고 1회용으로 패키지 사용 시 쓰는 명령이다. 여기선 postgraphile 패키지를 사용한다. |
| -c  | 서버 작동시킴                                                                    |

'postgres://postgres:111111@localhost/sample\_for\_graphql'

|       |      |      |           |
|-------|------|------|-----------|
| 사용자이름 | 비밀번호 | ip주소 | 접속할 DB 이름 |
|-------|------|------|-----------|

내 컴퓨터에 있는 postgres 에 접속할 것이기 때문에, localhost라고 썼다. 이 명령으로 인해, postgres 기본 포트인 5432 에 접근할 것이다.

|         |                                          |
|---------|------------------------------------------|
| --watch | postgraphile 이 특정 작업을 할때마다 터미널에 로그를 출력하라 |
|---------|------------------------------------------|

\* 로그는 작업기록을 말한다.

|                    |                                          |
|--------------------|------------------------------------------|
| --enhance-graphiql | graphiql 이라는 웹앱을 사용할것인데, 이것의 추가기능을 키는 명령 |
| --dynamic-json     | dynamic json 기능 사용                       |
| -o 또는 --cors       | CORS 허용을 위함                              |

\* CORS는 웹 보안과 관련있는데, 이 옵션을 켜주지 않으면 React 에서 PostGraphile 에 접근하지 못한다. CORS 는, 다른 주소의 컴퓨터에 함부로 접근할 수 없도록 하는 안전장치라고만 이해해두도록 하자. 아마 앞으로의 여러분 개발

인생에 상당한 짜증을 일으킬 것인데, 이 에러를 발견하면 어떻게 해결해야되는지 상황에 맞게 검색하가며 찾아보도록 하자.

\* - (하나) 와 -- (두개) 의 차이는, 옵션의 축약어와 옵션이다. 버전 출력을 예를 들면,

-v  
--version

```
+ jonyDev npx postgraphile -c 'postgres://postgres:11111@localhost/postgres' --watch --enhance-graphiql --dynamic-json -o
PostGraphile v4.12.3 server listening on port 5000 🚀

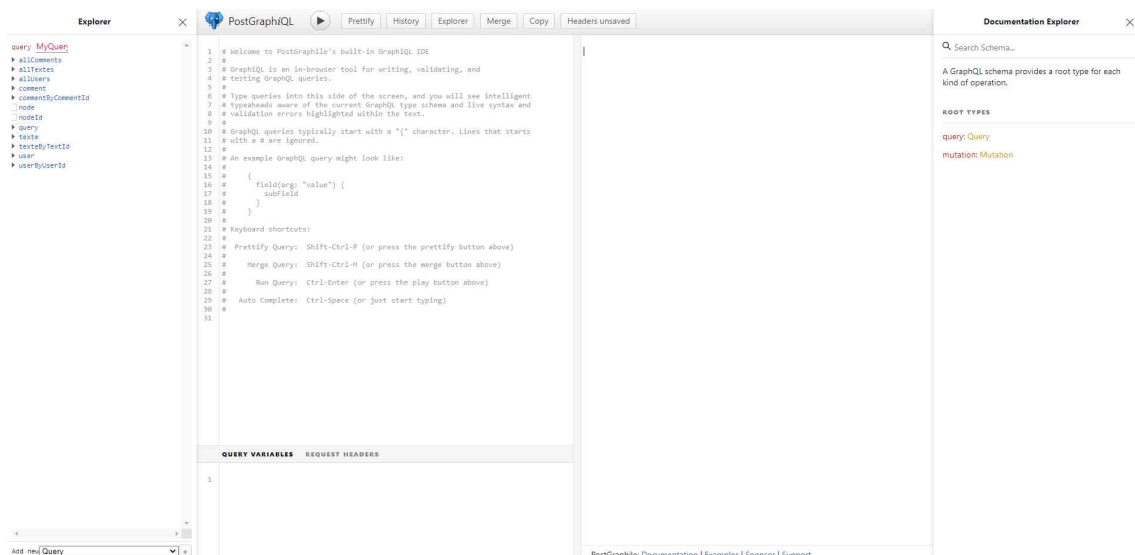
• GraphQL API: http://localhost:5000/graphql
• GraphiQL GUI/IDE: http://localhost:5000/graphiql
• Postgres connection: postgres://postgres:[SECRET]@localhost/postgres (watching)
• Postgres schema(s): public
• Documentation: https://graphile.org/postgraphile/introduction/
• Node.js version: v14.17.2 on linux x64
• Join Storyscript in supporting PostGraphile development: https://graphile.org/sponsor/
```

위 화면과 같이 뜨면서, 보여지는 코드 아래에 아무 에러도 뜨지 않으면 성공이다.

첫번째 링크는 React 에서 PostGraphile 에 접속할 주소이다.

두번째 링크는 GraphQL을 연습해볼 수 있는 툴인 GraphiQL 이다.

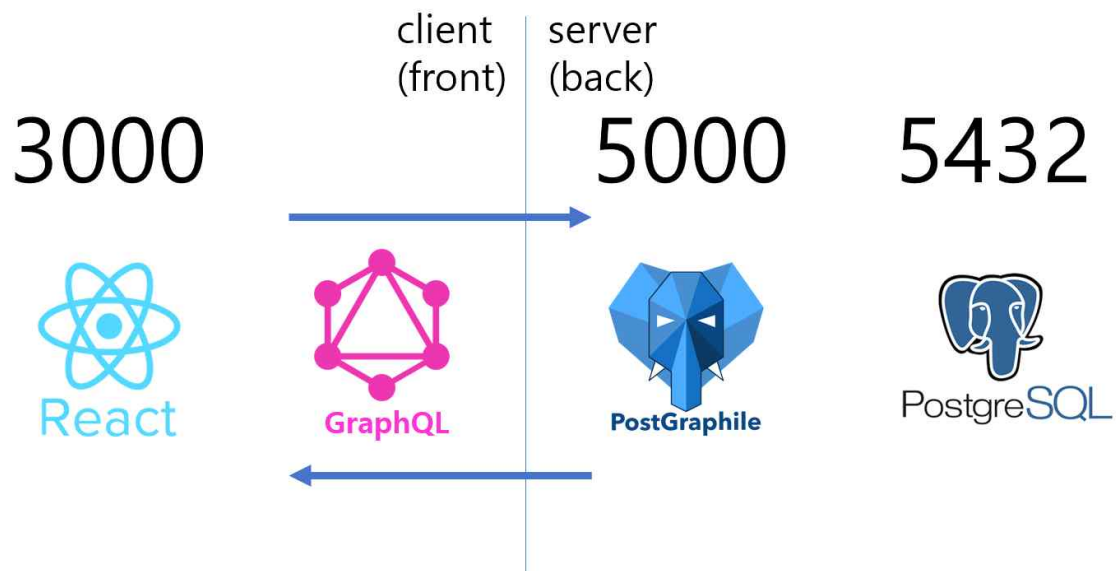
두번째 링크인 graphiql 에 접속해보자. ctrl + 클릭



왼쪽에 allComments, allTextes 같은, 자동으로 생성된 함수들이 잘 보이면 성공이다.

무슨 일이 일어났는지 정리해보면,





우리의 데이터는 5432 포트에 위치한 PostgreSQL 에 저장되어있다.  
 GraphQL 을 쓸 수 있는 라이브러리인 PostGraphile 을 통해 PostgreSQL 에  
 접속하고, PostGraphile 은 5000 포트에서 실행된다.  
 React 는 3000 포트에서 실행될 것이다. Apollo Client 라이브러리를  
 사용할것인데, React에서 사용하는 GraphQL 이다. 이것을 통해 5000번 포트에  
 접속해서 GraphQL 을 사용해 데이터베이스 CRUD 가 가능하다.

무슨 말인지 당장은 모를 것이다. 이걸 제대로 느끼려면 실전개발수업까지 가봐야  
 안다. 우선, GraphQL의 쿼리와 뮤테이션을 이해하는것을 목표로 하자.

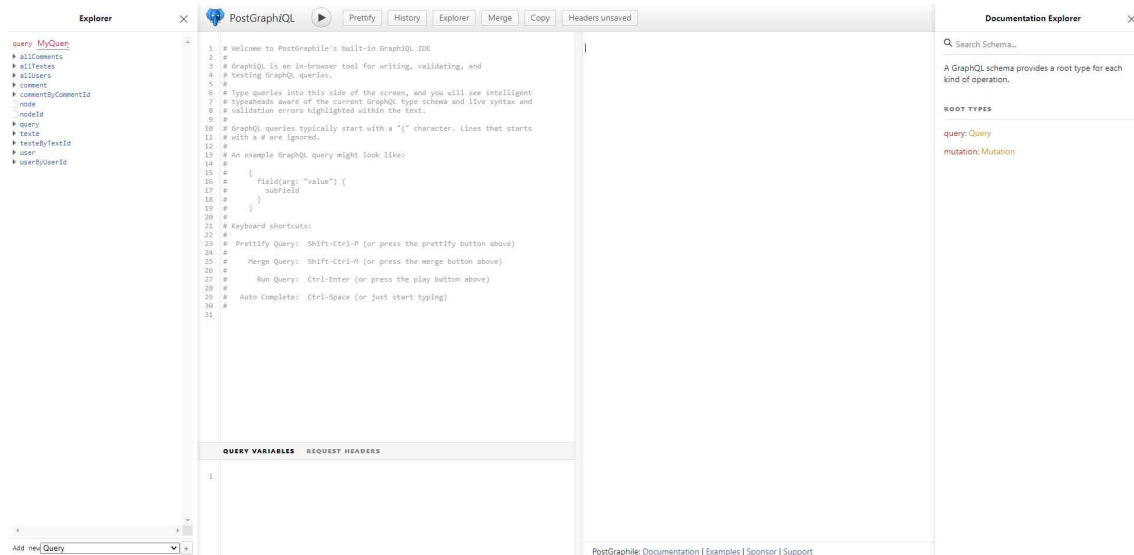
### 3. 쿼리(query)

GraphQL 을 사용하는 CRUD 는 크게 두 가지로 나뉜다.  
 쿼리(query)를 작성해 데이터를 JSON 으로 가져온다 (R)  
 뮤테이션(Mutation)으로 데이터를 생성(C), 수정(U), 삭제(D)한다.

클라이언트, 우리 기준으로 React 에서 Apollo Client 를 통해 쿼리 또는 뮤테이션  
 요청을 PostGraphile 로 보내면, 요청에 해당하는 JSON 을 받을 것이다.

먼저 쿼리부터 알아보자.

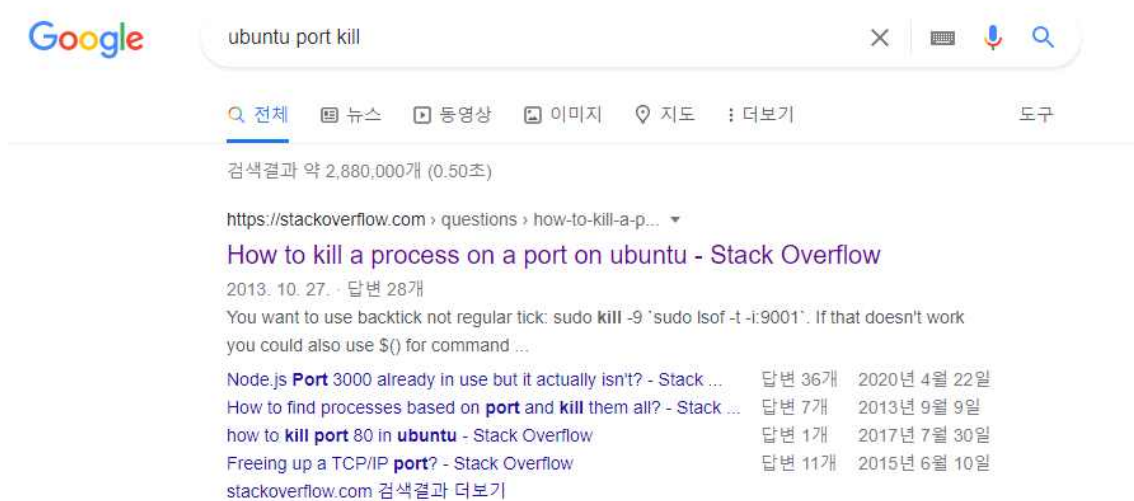
## PORT 끄고 키는 법



localhost:5000/graphiql 로 접속한 화면이다. 아까전에 긴 명령어를 입력하면 접속할 수 있다.

진행하기 전에 포트 끄는법에 대해 잠깐 설명하고 넘어가자. 간혹 포트가 이미 잡혀있어서 연결할 수 없는 경우엔, 해당 포트를 강제로 꺼줘야한다. ubuntu 에서 포트를 끌 땐 어떻게 하면 되는가? 나도 모든 것을 아는 게 아니기때문에 구글검색을 해보자.

구글 검색 ubuntu port kill



신뢰의 이름 Stack Overflow 에 들어가보면,

stackoverflow About Products For Teams Search... Log in Sign up

Home PUBLIC Questions Tags Users COLLECTIVES Explore Collectives FIND A JOB Jobs Companies TEAMS Stack Overflow for Teams - Collaborate and share knowledge with a private group. Create a free Team What is Teams?

## How to kill a process on a port on ubuntu

Asked 9 years, 5 months ago Active 1 month ago Viewed 1.1m times

647 I am trying to kill a process in the command line for a specific port in ubuntu.

If I run this command I get the port:

```
sudo lsof -t -i:9001
```

so...now I want to run:

```
sudo kill 'sudo lsof -t -i:9001'
```

I get this error message:

```
ERROR: garbage process ID "lsof -t -i:9001".
Usage:
kill pid ... Send SIGTERM to every process listed.
kill signal pid ... Send a signal to every process listed.
kill -s signal pid ... Send a signal to every process listed.
kill -l List all signal names.
kill -L List all signal names in a nice table.
kill -l signal Convert between signal numbers and names.
```

I tried `sudo kill 'lsof -t -i:9001'` as well

ubuntu port kill

Share Improve this question Follow

edited Oct 29 '19 at 17:49 marc\_s 680k • 159 • 1261 • 1392

asked Feb 19 '12 at 2:44 Tampa 63.4k • 106 • 251 • 390

11 FYI, Ubuntu questions have better chance of being answered on the Ubuntu part of the StackExchange (StackOverflow) network: askubuntu.com - JScoobyCed Feb 19 '12 at 3:06

The Overflow Blog

- Privacy is an afterthought in the software lifecycle. That needs to change.
- Why you should build on Kubernetes from day one

Featured on Meta

- Deprecating our mobile views
- Planned maintenance scheduled for Saturday, July 24, 2021 at 12:00pm UTC...

Linked

- 7 Deploying Meteor app from OS X to Linux causes bcrypt issues
- 0 xmllrpc - dynamically choosing port number
- 1 Getting error while trying to run a maven based simple springboot project
- 0 Docker-Composing error During Programming connectivity
- 1 Nginx address port 80 already in use
- 0 The command "gradle jettyRunWar" doesn't work, on Ubuntu
- 0 Erro 98 (Address already in use) Python Opencv

Related

- 2738 How can you find out which process is listening on a TCP or UDP port on...

해당 질문에 대한 추천이 무려 647개이다.

1261 You want to use backtick not regular tick:

```
sudo kill -9 `sudo lsof -t -i:9001`
```

If that doesn't work you could also use `$(C)` for command interpolation:

```
sudo kill -9 $(sudo lsof -t -i:9001)
```

Share Improve this answer Follow

edited Sep 3 '19 at 2:58 Tanya Branagan 381 • 1 • 8 • 17

answered Feb 19 '12 at 2:47 zellio 25.7k • 1 • 38 • 57

48 You forgot to add the signal option. Should read: `sudo kill -9 $(sudo lsof -t -i:9001) -` Francesco Gramano May 18 '15 at 4:18

9 @FrancescoGramano: you don't want to use -9 unless you have to. - zellio Jun 1 '15 at 16:55

2 when `sudo kill` can't kill the process, @putra.koreng 's method works well - Fangxin Dec 15 '17 at 1:07

3 if above is not working then try this: [stackoverflow.com/a/50515868/9339242](https://stackoverflow.com/a/50515868/9339242) - Arayan Singh May 24 '18 at 18:29

1 Is this OS independent or linux specific? If Linux specific then, even this command works `os.system("fuser -k 8080/tcp");` - Ridhuvashan Jul 25 '18 at 12:04

Show 8 more comments

채택된 답변의 추천은 1261개로 매우 높다. 첫번째 코드가 실행되지 않았을 때 두번째 코드를 실행시키라고 되어있으니, 두번째를 실행해보자.

```
~/.jonyDev
jonyDev npx postgraphile -c 'postgres://postgres:11111@localhost/postgres' --watch --enhance-graphiql --dynamic-json -o
PostGraphile v4.12.3 server listening on port 5000

• GraphQL API: http://localhost:5000/graphql
• GraphiQL GUI/IDE: http://localhost:5000/graphiql
• Postgres connection: postgres://postgres:[SECRET]@localhost/postgres (watching)
• Postgres schema(s): public
• Documentation: https://graphile.org/postgraphile/introduction/
• Node.js version: v14.17.2 on linux x64
• Join Surge.io in supporting PostGraphile development: https://graphile.org/sponsor/

^Z
[11] + 3984 suspended npx postgraphile -c 'postgres://postgres:11111@localhost/postgres' --watch
jonyDev npx postgraphile -c 'postgres://postgres:11111@localhost/postgres' --watch --enhance-graphiql --dynamic-json -o
events.js:352
 throw er; // Unhandled 'error' event
 ^

Error: listen EADDRINUSE: address already in use 127.0.0.1:5000
 at Server.setupListenHandle [as _listen2] (net.js:1320:16)
 at listenInCluster (net.js:1368:12)
 at GetAddrInfoReqWrap.dolisten [as callback] (net.js:1505:7)
 at GetAddrInfoReqWrap.onlookup [as oncomplete] (dns.js:71:8)
Emitted 'error' event on Server instance at:
 at emitErrorNT (net.js:1347:8)
 at processTicksAndRejections (internal/process/task_queues.js:82:21) {
 code: 'EADDRINUSE',
 errno: -98,
 syscall: 'listen',
 address: '127.0.0.1',
 port: 5000
}
```

현재 실행되고 있는 작업에서 나가고싶을 때, 우리의 경우는 PostGraphile 에서 나가고 싶을 땐 ctrl+z 를 사용한다. 이걸 프로그램마다 다르다. ctrl+c 를 사용하기도 하고, .exit 를 사용하기도 한다. ctrl+z 를 입력해서 나가서 다시 접속을 시도할 경우, 이미 해당 포트, 5000 이 실행되고 있기 때문에 실행할 수 없다는 에러가 뜬다.

Stack Overflow 에서 배운 명령을 이용하자면,

```
$ sudo kill -9 $(sudo lsof -t -i:5000)
```

맨 끝에 종료하고자하는 네자리 포트 숫자를 입력해주자. PostGraphile은 5000번이다.

```
~/.jonyDev
jonyDev sudo kill -9 $(sudo lsof -t -i:5000)
jonyDev
```

정상적으로 종료된 경우엔 다음과 같이 아무것도 뜨지 않는다. 만약, 이미 꺼져있는 포트라면?

```

→ jonyDev sudo kill -9 $(sudo lsof -t -i:5000)
→ jonyDev sudo kill -9 $(sudo lsof -t -i:5000)

Usage:
kill [options] <pid> [...]

Options:
<pid> [...] send signal to every <pid> listed
--<signal>, -s, --signal <signal>
 specify the <signal> to be sent
-l, --list=[<signal>] list all signal names, or convert one to a name
-L, --table list all signal names in a nice table

-h, --help display this help and exit
-V, --version output version information and exit

For more details see kill(1).
→ jonyDev |

```

다음과 같이, 사용법 설명을 출력하는것을 확인할 수 있다.

이제 다시 긴 명령어를 입력해 PostGraphile 을 작동시켜보면, GraphiQL 툴이 브라우저에 나타날 것이다.

QUERY CASE1: 한 명의 유저 데이터를 가져올 경우

| USER          |          |      |                   |            |            |
|---------------|----------|------|-------------------|------------|------------|
| user_id       | password | name | about_me          | joined     | isactivate |
| assistant0603 | 123123   | 조교행님 | 안녕하세요 조교행님입니다.    | 2013-04-13 | TRUE       |
| corini        | 113366   | 코리니  |                   | 2015-12-10 | TRUE       |
| joayo         | joayo123 | 조야요맨 | 조야요~~             | 2013-08-21 | TRUE       |
| sujini        | 111111   | 수지니  | 리엑트 배우고싶은 수지니예요~~ | 2013-04-14 | FALSE      |

### React에서 작성할 Query

```

1 query {
2 getUserById {
3 userByUserId(userId: "assistant0603") {
4 aboutMe
5 name
6 }
7 }
8 }

```

내가 정한 쿼리 이름  
유저 아이디  
내가 필요한 컬럼이름

### React에서 받을 JSON

```

1 {
2 "data": {
3 "userByUserId": {
4 "aboutMe": "안녕하세요 조교행님입니다.",
5 "name": "조교행님"
6 }
7 }
8 }

```

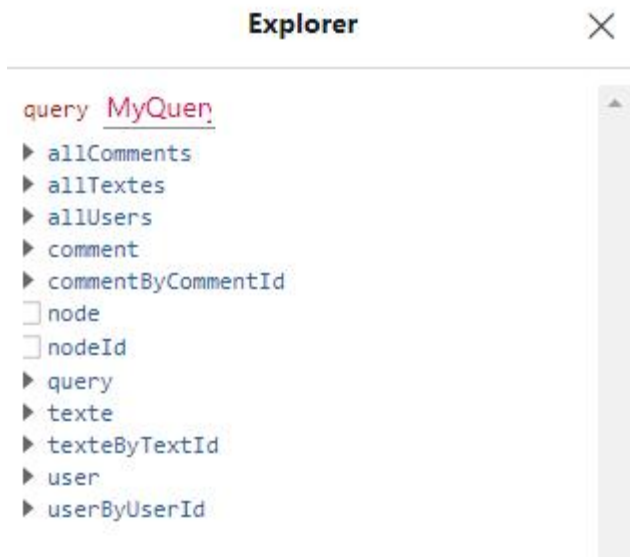
데이터를 보면, 내가 가져오고 싶은 건 id assistant0603 의 name 과 about\_me 이다. 다른 건 가져오고싶지 않다. 그럴 경우에, React 에선 다음 쿼리를 작성한다.

일단, 쿼리 이름을 getUserById 로, 내가 짓고 싶은 이름으로 정한다. 그 다음, userByUserId() 라는, PostGraphile 에서 자동으로 생성된 함수를 사용할 것이다. PostGraphile 의 장점 중 하나는, 기본적인 CRUD 를 위해 당장 필요한 함수를 자동으로 생성해준다는 것이다.

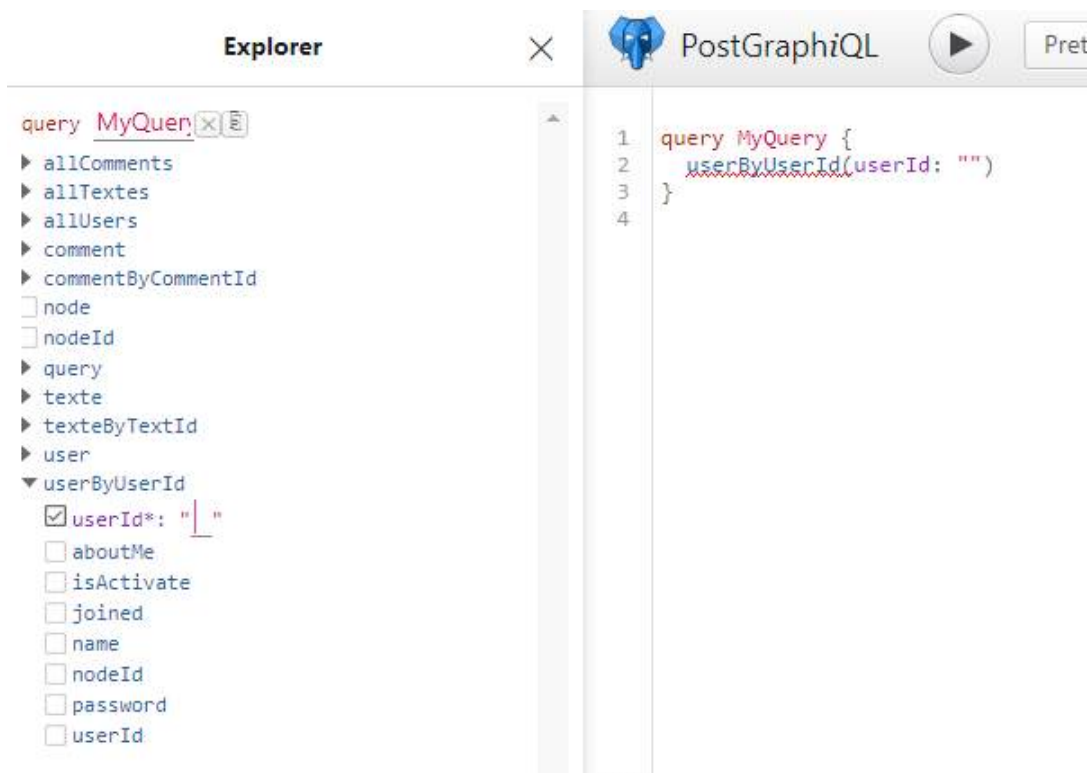
외우려고 하지 마라! GraphiQL 을 이용해서, 컨닝할것이다. 위 그림을 예쁘게 정리해두긴 했는데, 외울 필요가 하나도 없다.

함수의 파라미터로 유저 아이디인 assistant0603을 넣고, 내가 필요한 컬럼이름을 써넣으면, 오른쪽과 같은 JSON 으로 받을 것이다. 이것도 외우려고 하지 마라!

React 에서 `console.log()` 로 찍어보면 된다.

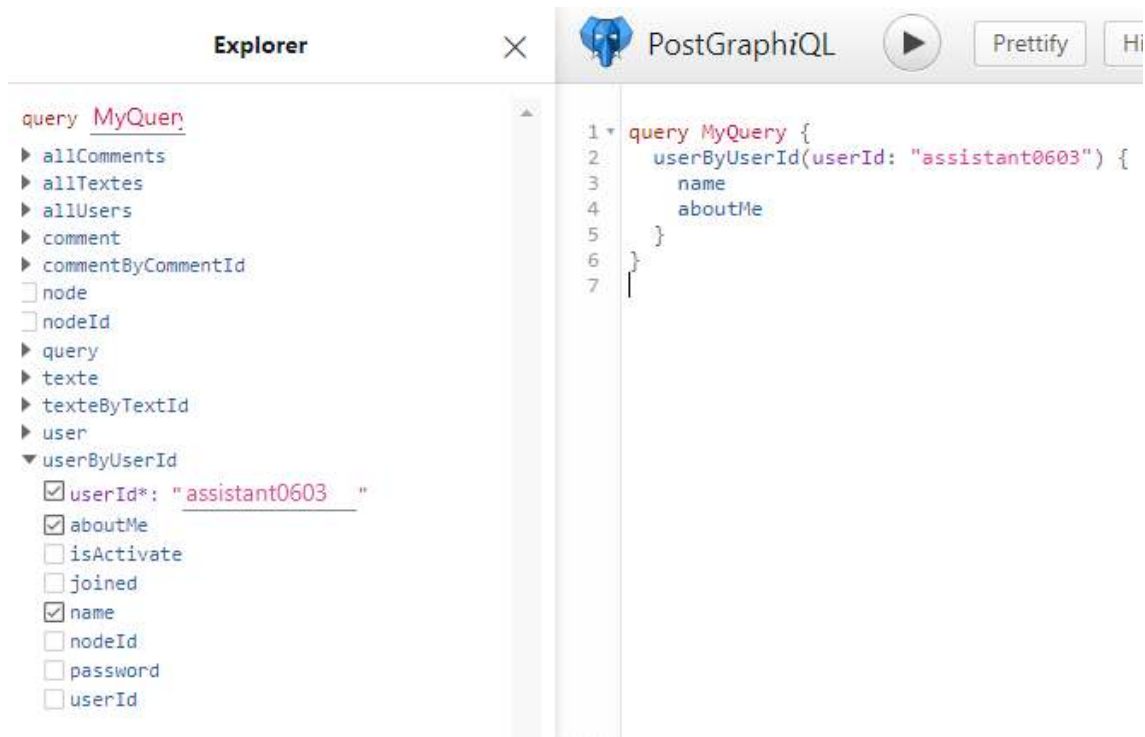


초보자에게는 왼쪽 함수목록이 핵심이다. 외우지 말라고 한 이유가 여기에 있다. 한 명의 유저 정보를 가져오고 싶기에 맨 아래 `userByUserId` 를 클릭하면,



옆에 쿼리가 뜬다. 우리가 가져오고자 하는 아이디는 `assistant0603` 이므로 입력해주자. 그리고 `name` 과 `aboutMe` 체크박스를 클릭해주면 다음과 같다.





작업하다가 코드가 안 예쁠 경우엔, 재생버튼 옆에 Prettify 버튼을 클릭해주면 코드가 예쁘게 정렬된다. 재생버튼 눌러보자.

```
{
 "data": {
 "userByUserId": {
 "name": "조교행님",
 "aboutMe": "안녕하세요 조교행님입니다."
 }
 }
}
```

오른쪽 화면에, 우리가 받을 JSON 이 출력된다.

외워야하는가? 외우지 마라! 이것저것 실행해보면서 컨닝해서, React에 복사 붙여넣기 하고, JSON 은 React에서 console.log() 찍어서 잘 들어오는지 확인할 것이다.

## QUERY CASE2: 모든 유저 데이터를 가져올 경우

| USER          |          |      |                   |            |            |
|---------------|----------|------|-------------------|------------|------------|
| user_id       | password | name | about_me          | joined     | isactivate |
| assistant0603 | 123123   | 조교행님 | 안녕하세요 조교행님입니다.    | 2013-04-13 | TRUE       |
| corini        | 113366   | 코리니  |                   | 2015-12-10 | TRUE       |
| joayo         | joayo123 | 조아요맨 | 조아요~~             | 2013-08-21 | TRUE       |
| sujini        | 111111   | 수지니  | 리액트 배우고싶은 수지니예요~~ | 2013-04-14 | FALSE      |

### React에서 작성할 Query

```

1 query getAllUsers {
2 allUsers { 내가 정한 쿼리 이름
3 edges {
4 node {
5 name
6 aboutMe
7 } 내가 필요한 컬럼이름
8 }
9 }
10 totalCount
11 }
```



**React에서 받을 JSON**

```

1 {
2 "data": {
3 "allUsers": {
4 "edges": [
5 {
6 "node": {
7 "name": "조교행님",
8 "aboutMe": "안녕하세요 조교행님입니다."
9 }
10 },
11 {
12 "node": {
13 "name": "코리니",
14 "aboutMe": null
15 }
16 },
17 {
18 "node": {
19 "name": "조아요맨",
20 "aboutMe": "조아요~~"
21 }
22 },
23 {
24 "node": {
25 "name": "수지니",
26 "aboutMe": "리액트 배우고싶은 수지니예요~~"
27 }
28 }
29],
30 "totalCount": 4
31 }
32 }
33 }
```

만약 모든 유저의 닉네임과 자기소개를 가져오고싶은 경우라면?

아까처럼, 맨 앞엔 내가 정한 쿼리 이름을 쓰고,

allUsers 라는, 기본 제공되는 함수 다음에

edges 다음에 node 다음에 내가 필요한 컬럼이름을 쓴다.

totalCount 는 배열의 갯수가 필요한 경우를 위해 PostGraphile 에서 자동제공한다.

외워야하는가? 역시 컨닝할것이다.

\* edge 와 node 는 GraphQL 의 Graph 이론과 관련이 있다. 설명을 하면 굉장히 어렵기때문에 넘어가도록 한다. 우리 데이터만 잘 가져오면 되기 때문에 설명은 불필요하다고 판단된다.



```

query MyQuery {
 allComments
 allTextes
 allUsers {
 after:
 before:
 condition:
 first:
 last:
 offset:
 orderBy:
 edges
 nodes
 pageInfo
 totalCount
 }
 comment
 commentByCommentId
 node
 nodeId
 query
 texte
 texteByTextId
 user
 userByUserId
}

```

allUsers 만 클릭하고 재생버튼 눌러보자.

```

query MyQuery {
 allComments
 allTextes
 allUsers {
 after:
 before:
 condition:
 first:
 last:
 offset:
 orderBy:
 edges {
 cursor
 node {
 aboutMe
 isActivate
 joined
 name
 nodeId
 password
 userId
 }
 }
 nodes
 pageInfo
 totalCount
 }
 comment
 commentByCommentId
 node
 nodeId
 query
 texte
 texteByTextId
 user
 userByUserId
}

```

다음과 같이, 알아서 엣지와 노드가 쓰여진다. 자동으로 about\_me 가 들어갔는데, name, totalCount 까지 체크해보고 재생버튼 눌러보자.

```

{
 "data": {
 "allUsers": {
 "edges": [
 {
 "node": {
 "aboutMe": "안녕하세요 조교행님입니다.",
 "name": "조교행님"
 }
 },
 {
 "node": {
 "aboutMe": null,
 "name": "코리니"
 }
 },
 {
 "node": {
 "aboutMe": "조아요~~",
 "name": "조아요맨"
 }
 },
 {
 "node": {
 "aboutMe": "리액트 배우고싶은 수지니예요",
 "name": "수지니"
 }
 }
],
 "totalCount": 4
 }
 }
}

```

잘 받아온것을 확인할 수 있다. 외운 게 한 줄이라도 있는가? 하나도 없다! 그냥 클릭 몇 번 했을 뿐이다. 우리의 쿼리를 그대로 복사해서 React 에 붙여넣으면 되고, 위 이미지의 JSON 을 받아 console.log() 로 찍어볼 것이다.

쉽다고 해서 무시하지 말라. 초보자는 일단 쉽게 시작해서 나만의 앱을 만들어봐야한다. 세 개 정도 만들고나면 내가 부족한 게 보일 것이다. 그 때 새로운것을 배우면 된다.

## 4. 뮤테이션(mutation)

CRUD의 R(Read) 는 알았고, 나머지 CUD 를 배워보도록 하자. GraphQL 에션 이 작업을 query라고 하지 않고 mutation 이라고 부른다.

### Mutation CASE 1: create

#### React에서 작성할 Mutation

```
1 mutation { create {
 자동 생성된 함수 이름
 create {
 input: {
 user: {
 4 name: "newbie11"
 5 password: "gogo1234"
 6 name: "뉴비"
 7 joined: "2021-07-15"
 8 isactivate: true
 9 aboutMe: "새로 가입했어요~~"
 10 }
 11 }
 12 }
 13 }
 14 }
 15 }
 16 }
 17 }
 18 }
 19 }
 20 }
 21 }
```

input 내용

편의를 위해 mutation 안에 query 가 있을수도 있다

| USER          |          |      |                    |            |            |
|---------------|----------|------|--------------------|------------|------------|
| user_id       | password | name | about_me           | joined     | isactivate |
| assistant0603 | 123123   | 조고형님 | 안녕하세요 조고형님입니다.     | 2013-04-13 | TRUE       |
| corini        | 113366   | 코리니  |                    | 2015-12-10 | TRUE       |
| joayo         | joayo123 | 조야요맨 | 조야요~~              | 2013-08-21 | TRUE       |
| sujini        | 111111   | 수지니  | 리액트 배우고 싶은 수지니예요~~ | 2013-04-14 | FALSE      |
| newbie11      | gogo1234 | 뉴비   | 새로 가입했어요~~         | 2021-07-15 | TRUE       |

#### React에서 받을 JSON

```
1 {
2 "data": {
3 "createUser": {
4 "query": {
5 "userByUserId": {
6 "name": "뉴비",
7 "aboutMe": "새로 가입했어요~~"
8 }
9 }
10 }
11 }
12 }
```

왜 굳이 받을까?  
data fetch 성공 여부 알기 위해

- fetch : 가지고 오다
- patch : 프로그램을 수정하다

내가 만약 뉴비라는 사용자를 추가하고 싶다면 다음과 같이 할 수 있을 것이다. 상황에 따라 컬럼은 골라서 입력할 수 있다. 왜? 디폴트값이 정해져 있을수도 있고, 널값을 허용할 수도 있기 때문이다.

다른 건 신경쓸 거 없고, input 내용만 잘 넣어주면 된다. 그리고, 아래쪽에 쿼리를 하나 추가해줬다. 꼭 이렇게 할 필요는 없는데 왜 넣어줬을까? 데이터 패치 성공 여부를 알기 위해, 입력한 직후 해당 아이디를 받아서 잘 입력되었는지 확인하는것이다. 이처럼, 쿼리와 뮤테이션을 마음껏 조합할 수 있다!

\* patch 와 fetch 는 다른 것이다.

patch                  프로그램을 수정하다. ex) 게임 패치  
fetch                  데이터를 가지고 오다.

## Mutation CASE 2: update

| USER          |          |      |                   |            |            |
|---------------|----------|------|-------------------|------------|------------|
| user_id       | password | name | about_me          | joined     | isactivate |
| assistant0603 | 123123   | 조교형님 | 안녕하세요 조교형님입니다.    | 2013-04-13 | TRUE       |
| corini        | 113366   | 코리니  |                   | 2015-12-10 | TRUE       |
| joayo         | joayo123 | 조아요맨 | 조아요~~             | 2013-08-21 | TRUE       |
| sujini        | 111111   | 수지니  | 리액트 배우고싶은 수지니예요~~ | 2013-04-14 | FALSE      |
| newbie11      | gogo1234 | 뉴비뉴빗 | 새로 가입했어요~         | 2021-07-15 | TRUE       |

### React에서 작성할 Mutation

```

1 mutation updateUserByUserId {
2 updateUserByUserId(
3 input: {
4 userPatch: {
5 name: "뉴비뉴빗"
6 },
7 userId: "newbie11"
8 }
9) {
10 query {
11 userByUserId(userId: "newbie11") {
12 name
13 }
14 }
15 }
16 }

```



### React에서 받을 JSON

```

1 {
2 "data": {
3 "updateUserByUserId": {
4 "query": {
5 "userByUserId": {
6 "name": "뉴비뉴빗"
7 }
8 }
9 }
10 }
11 }

```

뉴비의 닉네임을 뉴비뉴빗으로 바꾸고 싶은 경우다. 외우지 않을 것이다. 나중에 짚어보면서 익혀보자.

## Mutation CASE 3: delete

| USER          |          |      |                   |            |            |
|---------------|----------|------|-------------------|------------|------------|
| user_id       | password | name | about_me          | joined     | isactivate |
| assistant0603 | 123123   | 조교형님 | 안녕하세요 조교형님입니다.    | 2013-04-13 | TRUE       |
| corini        | 113366   | 코리니  |                   | 2015-12-10 | TRUE       |
| joayo         | joayo123 | 조아요맨 | 조아요~~             | 2013-08-21 | TRUE       |
| sujini        | 111111   | 수지니  | 리액트 배우고싶은 수지니예요~~ | 2013-04-14 | FALSE      |
| newbie11      | gogo1234 | 뉴비뉴빗 | 새로 가입했어요~         | 2021-07-15 | TRUE       |

### React에서 작성할 Mutation

```

1 mutation deleteUserByUserId {
2 deleteUserByUserId(
3 input: {
4 userId: "newbie11"
5 }
6) {
7 query {
8 allUsers {
9 edges {
10 node {
11 name
12 }
13 }
14 }
15 }
16 }
17 }

```



하나 삭제되었으니  
나머지 이름들 받아보기

```

1 {
2 "data": {
3 "deleteUserByUserId": {
4 "query": {
5 "allUsers": {
6 "edges": [
7 {
8 "node": {
9 "name": "조교형님"
10 }
11 },
12 {
13 "node": {
14 "name": "코리니"
15 }
16 },
17 {
18 "node": {
19 "name": "조아요맨"
20 }
21 },
22 {
23 "node": {
24 "name": "수지니"
25 }
26 }
27]
28 }
29 }
30 }
31 }
32 }

```

### React에서 받을 JSON

마지막은 delete다. 뉴비뉴빗 유저를 지워버릴것이다. 역시 핵심은 id를 아는 것이다. 생성이든 수정이든 삭제가든, mutation의 핵심은 id를 아는 것이라고 할 수 있다.

쿼리로 뭘 받아오면 좋을까? 삭제가 잘 되었는지 확인하기 위해 전체 유저 정보를 받아오면 좋을 것이다. 정해진 규칙이 아니다! 내가 필요하면 추가하는 것이고, 필요없으면 빼면 된다.

\* id 말고 name 이나 가입일자를 나타내는 joined 로 접근할 순 없는 것인가?  
가능하긴한데, 그 경우 여러분들이 직접 함수를 작성해줘야하는 번거로움이 있다.  
난이도가 어려워진다. 개인적으로, PostGraphile 을 쓰는 가장 큰 이유가, 서버  
코드 한 줄 쓸 필요없이 편하게 제공된다는 것이라고 생각한다. 만약 제대로 된  
서버쪽 GraphQL 을 쓰고 싶다면 PostGraphile 대신 Apollo Backend 를 배워보길  
추천한다.

그럼 방법이 아예 없단 말인가? 추천하는 방법으로, 우선 쿼리를 통해 id 정보를  
JSON 으로 가져오고, 파악한 id 로 뮤테이션을 하면 될 것이다. 정석이라곤 할 수  
없는데, 초보자 수준에서 이 정도면 충분하다고 생각한다.

## 실습



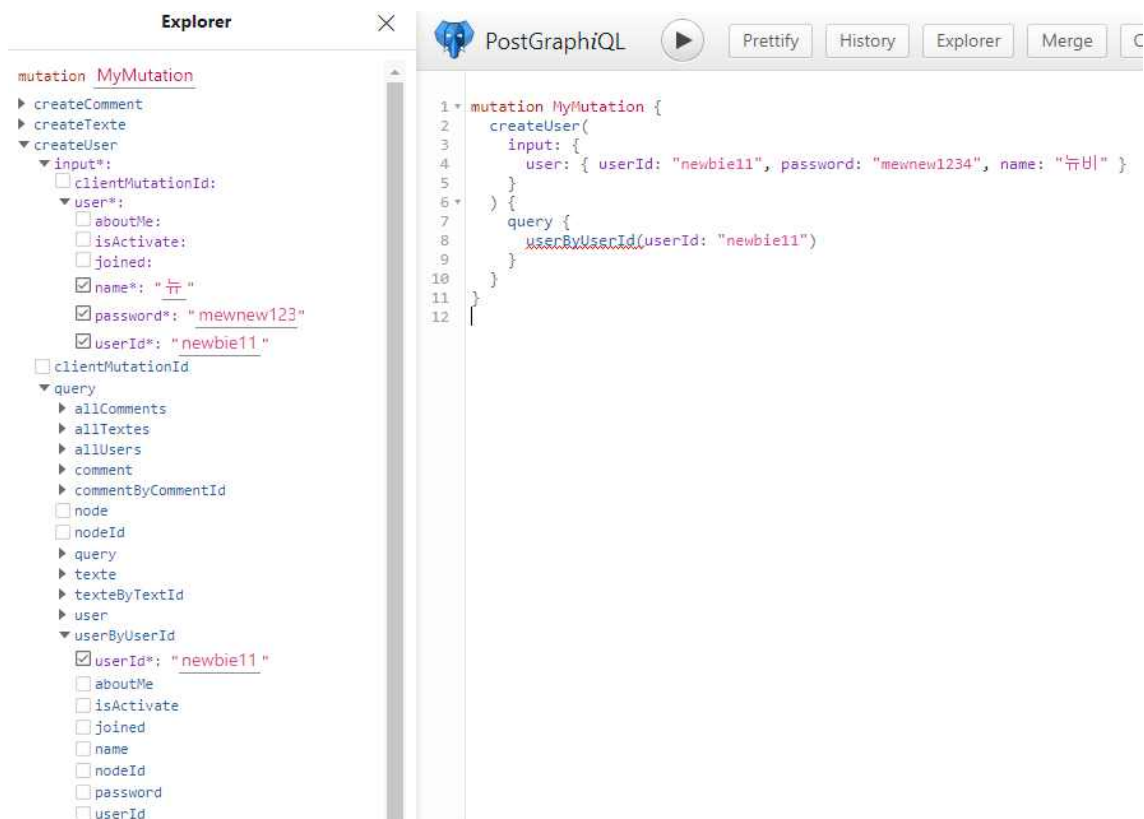
왼쪽 아래에 보면, Query 와 Mutation을 추가할 수 있다. Mutation 을 추가하자.



기존에 있던 쿼리는 삭제하고, 뮤테이션만 남겨두었다.

여기서, 주목할 부분은 맨 끝에 id가 있는것들이다. create 는 id 가 따로 없는데,  
테이블에서 행 전체를 추가할 것이기 때문에 우리가 직접 아이디를 “입력”  
해주거나, 다른 테이블의 경우 SERIAL 타입으로 자동증가할것이기 때문이다.

createUser 눌러보자.



내가 필요한것들만 클릭하고 입력해보았다. aboutMe 는 널값을 허용하니깐 체크하지 않을 것이고, isActivate 는 DEFAULT 값이 true, 그리고 joined 는 현재 날짜가 DEFAULT 로 정해져있다.  
그리고 쿼리 하나를 추가해주는데, id를 통해 user를 가져오도록 한다.  
재생버튼 클릭해보자.

```
{
 "data": {
 "createUser": {
 "query": {
 "userByUserId": {
 "aboutMe": null,
 "name": "뉴비"
 }
 }
 }
 }
}
```

쿼리로 인한 JSON 이 정상적으로 받아진 것을 확인할 수 있다. TablePlus 에서 새로고침해서 확인해보면,

| user_id       | password   | name  | about_me    | joined     | is_activate |
|---------------|------------|-------|-------------|------------|-------------|
| assistant0603 | 123123     | 조교... | 안녕하세요 조교... | 2013-04-13 | TRUE        |
| corini        | 113366     | 코리니   | NULL        | 2015-12-10 | TRUE        |
| joayo         | joayo123   | 조아... | 조아요~~       | 2013-08-21 | TRUE        |
| newbie11      | mewnew1... | 뉴비    | NULL        | 2021-07-22 | TRUE        |
| sujini        | 111111     | 수지니   | 리액트 배우고싶... | 2013-04-14 | TRUE        |

다음과 같이 새 유저가 정상적으로 추가되었다.

다음은 update 이다.

```

▼ updateUserById
 ▼ input*:
 ☐ clientMutationId:
 ☒ userId*: "newbie11"
 ▼ userPatch*:
 ☐ aboutMe:
 ☐ isActivate:
 ☐ joined:
 ☒ name: "뉴비"
 ☐ password:
 ☐ userId:
 ☐ clientMutationId
 ▼ query
 ▶ allComments
 ▶ allTextes
 ▶ allUsers
 ▶ comment
 ▶ commentByCommentId
 ☐ node
 ☐ nodeId
 ▶ query
 ▶ texte
 ▶ texteByTextId
 ▶ user
 ▼ userById
 ☒ userId*: "newbie11"
 ☐ aboutMe
 ☐ isActivate
 ☐ joined
 ☒ name
 ☐ nodeId
 ☐ password
 ☐ userId

```

```

1 ▾ mutation MyMutation {
2 updateUserById(
3 input: { userPatch: { name: "뉴비뉴빗" }, userId: "newbie11" }
4) {
5 query {
6 userById(userId: "newbie11") {
7 name
8 }
9 }
10 }
11 }
12

```

수정하고자 하는 id를 입력해주고, 수정하고자 하는 이름은 뉴비뉴빗으로 설정했다. 그리고 쿼리 하나를 추가해서, 이름만 받아오도록 하자.

```

{
 "data": {
 "updateUserById": {
 "query": {
 "userById": {
 "name": "뉴비뉴빗"
 }
 }
 }
 }
}

```

쿼리의 결과로 인한 JSON 을 정상적으로 받았고, TablePlus 에서 새로고침해보면

| user_id         | password          | name        | about_me    | joined            | is_activate |
|-----------------|-------------------|-------------|-------------|-------------------|-------------|
| assistant0603   | 123123            | 조교형님        | 안녕하세요 조교... | 2013-04-13        | TRUE        |
| corini          | 113366            | 코리니         | NULL        | 2015-12-10        | TRUE        |
| joayo           | joayo123          | 조아요맨        | 조아요~~       | 2013-08-21        | TRUE        |
| <b>newbie11</b> | <b>mewnew1...</b> | <b>뉴비뉴빗</b> | <b>NULL</b> | <b>2021-07-22</b> | <b>TRUE</b> |
| sujini          | 111111            | 수지니         | 리액트 배우고싶... | 2013-04-14        | TRUE        |

다음과 같이, 뉴비에서 뉴비뉴빗으로 바뀐것을 확인할 수 있다.



마지막은 delete다.



```
1 mutation MyMutation {
2 deleteUserByUserId(input: { userId: "newbie11" }) {
3 query {
4 allUsers {
5 edges {
6 node {
7 name
8 }
9 }
10 }
11 }
12 }
13 }
14
```

삭제하고자 하는 유저 아이디를 입력하고, 전체 유저를 불러오는 쿼리를 하나 추가해줬다.

```

{
 "data": {
 "deleteUserById": {
 "query": {
 "allUsers": {
 "edges": [
 {
 "node": {
 "name": "조교행님"
 }
 },
 {
 "node": {
 "name": "코리니"
 }
 },
 {
 "node": {
 "name": "조아요맨"
 }
 },
 {
 "node": {
 "name": "수지니"
 }
 }
]
 }
 }
 }
 }
}

```

다음과 같이, 쿼리에 대한 결과로 JSON 이 정상적으로 들어온 것을 확인할 수 있는데, 뉴비뉴빔은 사라졌다.

| user_id       | password | name | about_me    | joined     | is_activate |
|---------------|----------|------|-------------|------------|-------------|
| assistant0603 | 123123   | 조교행님 | 안녕하세요 조교... | 2013-04-13 | TRUE        |
| corini        | 113366   | 코리니  | NULL        | 2015-12-10 | TRUE        |
| joayo         | joayo123 | 조아요맨 | 조아요~~       | 2013-08-21 | TRUE        |
| sujini        | 111111   | 수지니  | 리액트 배우고싶... | 2013-04-14 | TRUE        |

TablePlus에서 확인한 결과, 뉴비뉴빔은 더이상 존재하지 않는다.

쿼리든, 뮤테이션이든, 외우려고 하지 마라. PostGraphile 의 장점은 이렇게 컨닝하면서 내가 원하는 쿼리와 JSON 을 확인할 수 있다는 것이다.

## 5. 마치며

《ver.2021 노베이스 모던 웹개발》 코스는, 복싱을 배우는 것과 비슷하게 이루어져있다. 복싱장 가면 적어도 3개월까진 절대 링 위로 올려보내지 않는다. 거울보면서 십자가 그려놓고 기본적인 자세연습 시키고, 줄넘기, 달리기, 근력운동하고, 샌드백 치면서 기초체력부터 길러놓는다. 그리고 어느 정도 되었다 싶을 때 링 위로 올라간다. 지금까지, 이론만 주구장창 배웠다. 하지만 꼭 필요한 것들만 가르쳤다. 하나의 프로젝트하기에 필요한 것들만 가르친 것이다.

JavaScript, TypeScript 기초

React: component, props, useState

당장 필요한 CSS

Bootstrap 공식문서 구경

SQL DB, ERD

GraphQL query, mutation

더 어렵게, 더 많이 가르칠수도 있었지만 그렇게 하지 않았다. 여러분들은 어짜피 까먹을 것이고, 심지어 지금까지 코스를 진행하면서도 머릿속에 남아있는 지식이라곤 20% 밖에 없을것이라고 생각된다. 그래도, 링 위에 올라가야한다. 언제까지고 링 아래에 있을수는 없다. 이론을 완벽하게 가르친 적이 없기때문에, 전혀 새로운 개념들도 앞으로 배우게 될 것이다. 그러나, 그것들은 지금까지 배운 기초에 비하면, 부가적인 지식들일 뿐이다.

딱 하나의 수업만 더 하고 코스를 마치겠다. 지금까지 배운 것들을 종합해보는 실전개발 수업이다.