

ICE4016 데이터베이스설계

<제목>

10만명의 학생 데이터에 index 적용하기

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.

2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.

3. 나는 보고서의 내용을 조작하지 않겠습니다.

4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.

5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2022년 11 월 06일

학부 정보통신공학

학년 3

성명 조하영

학번 12194111

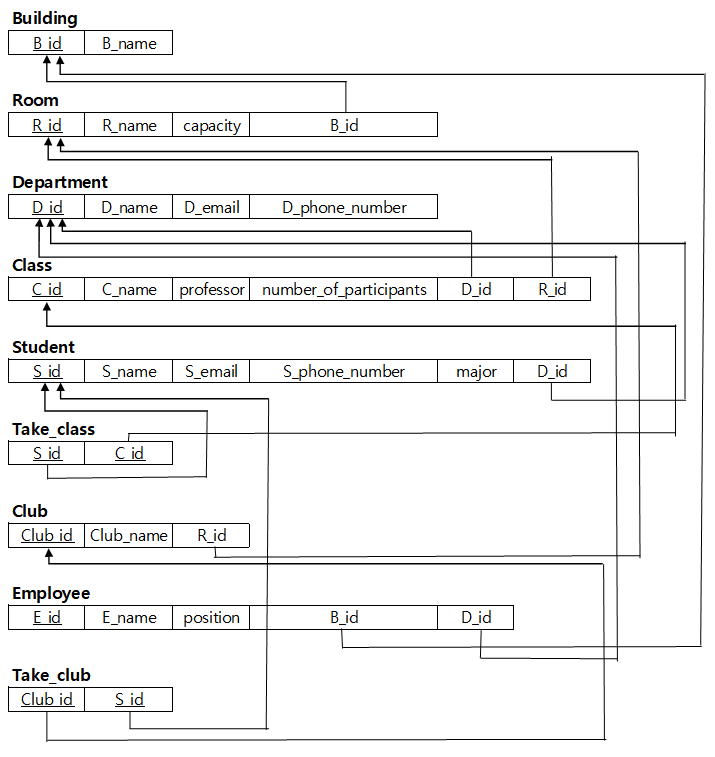


1. 개요

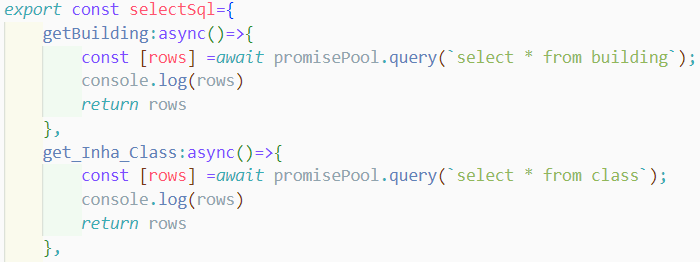
* 서버와 웹 페이지에서 INHA 데이터 베이스에 접근할 수 있도록 구현한다. 모든 테이블을 select로 조회할 수 있고, student 테이블의 경우 모든 attribute에 대해 update가 가능해야한다. 또한, student 테이블에 새로운 인스턴스를 insert할 수 있어야 한다.
* 임의로 학생 10만명을 만들어서 index 적용/미적용에 따른 결과를 제시할 수 있다. 이때, 특정 수업을 듣는 학생을 검색하는 조건으로 query를 작성한다.

1. 상세 설계 내용

**STEP 1. 웹에 INHA 데이터베이스 구현하기**

* 기존에 생성한 INHA 데이터베이스를 사용하여 위와 같은 구조로 작업하였다. (사진에는 babel.config.json, package-lock.json, package.json이 포함되어있지 않음)
* **sql.js**



* selectSql로 9개의 모든 테이블에 대해 select query를 진행할 수 있도록 하였다.



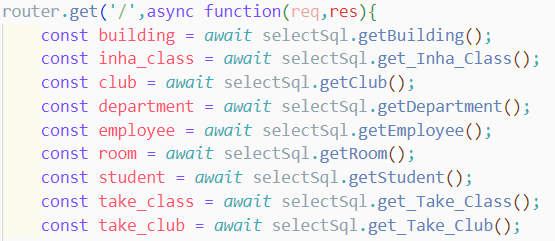
* insertSql로 student 테이블에 대해 insert query를 진행할 수 있도록 하였다.



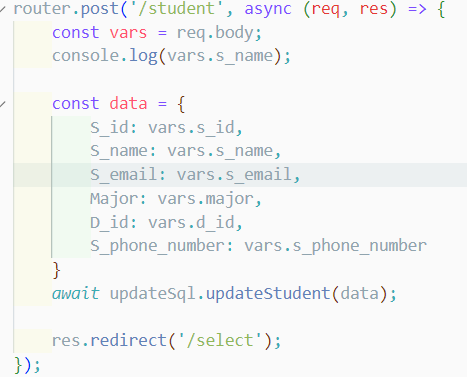
* updateSql로 student 테이블의 모든 attribute에 대해 update query를 진행할 수 있도록 하였다. 이때, PK인 S\_id를 기준으로 인스턴스를 찾아 query를 수행하기 때문에, S\_id가 반드시 입력되어야한다.
* **home.js**



* 웹은 home은 student 테이블에 대한 삽입 페이지이다. 따라서, insert query를 위해 사용자로부터 정보를 입력 받아 setStudent로 data를 넘겨주게 된다.
* **select.js**



* 조회할 수 있는 화면으로, 9개의 테이블에 대해 모든 인스턴스들이 출력되도록 하는 파일이다.
* **update.js**



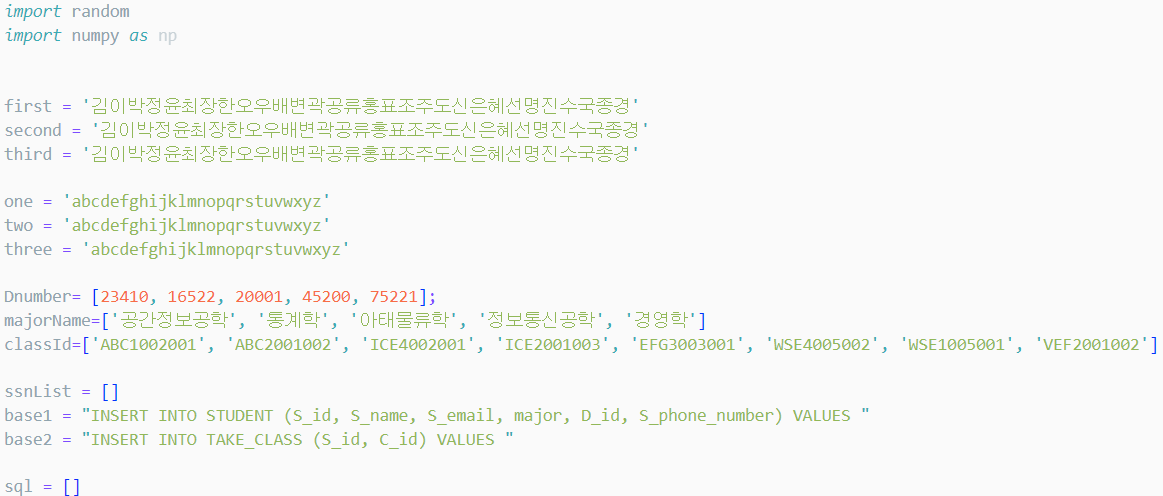
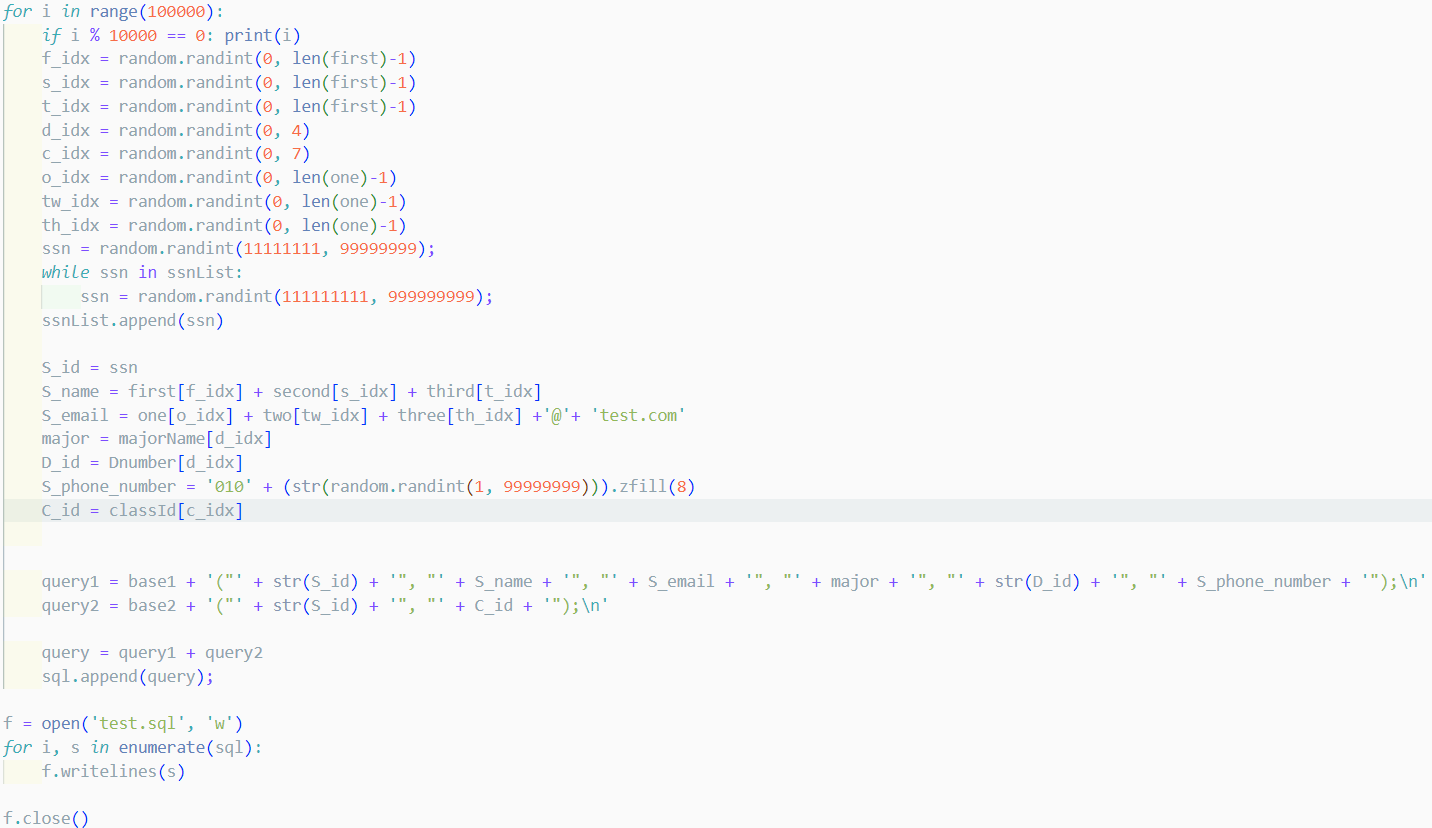
* student 테이블의 정보를 갱신하기 위한 파일이다. 사용자가 입력한 갱신 정보를 updateStudent에 넘겨주는 역할을 한다.
* **updateStudent.hbs**



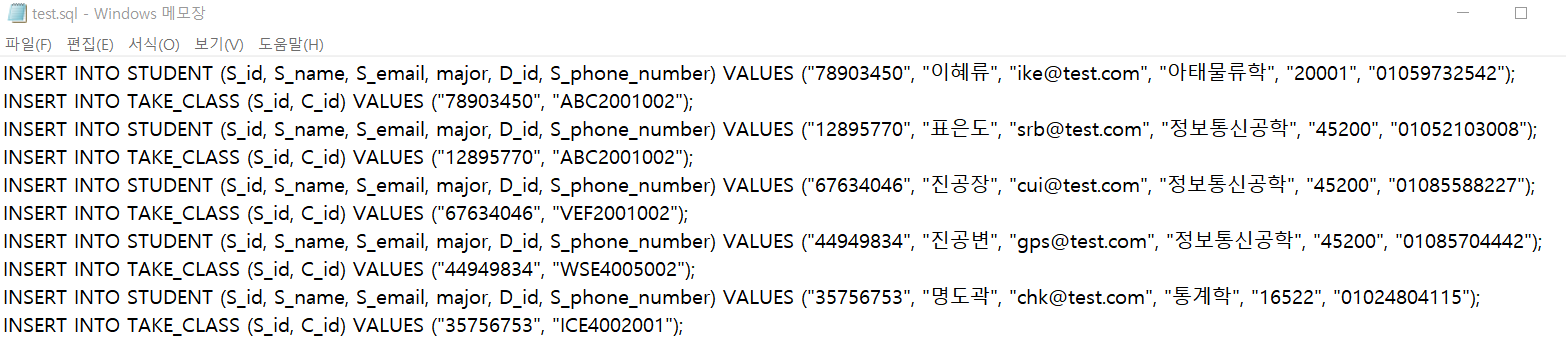
* student 테이블을 갱신하는 창을 구성하는 파일이다.

**STEP 2. index 적용하기**

* **createStudent.py**

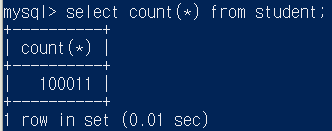
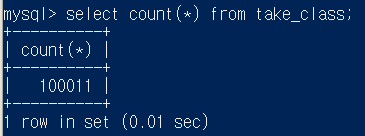
 

* 임의로 학생 10만명의 데이터를 생성하기 위한 파이썬 코드이다. 생성한 sql문은 test.sql에 저장하도록 하였다.
* student 테이블에 데이터를 삽입하기 위해서는 s\_id, s\_name, s\_email, major, d\_id, s\_phone\_number에 해당하는 정보가 필요하므로, 각 조건에 맞게 난수 정보를 생성하였다.
* 또, 특정 수업을 듣는 학생을 검색하는 조건으로 query를 작성해야하기 때문에 학생들이 듣는 수업 관련 데이터가 필요하다. 즉, Take\_class 테이블에 데이터를 삽입해야하므로, 생성된 10만명의 학생에 대하여 현재 class 테이블에 있는 8개의 수업 중 1개를 랜덤으로 고르도록 구현하였다.

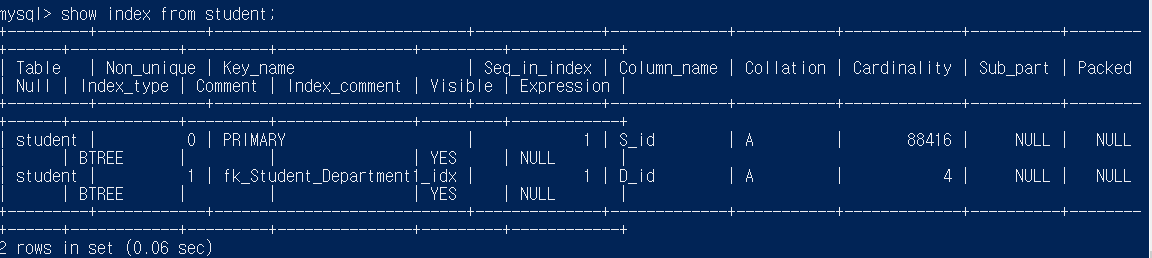


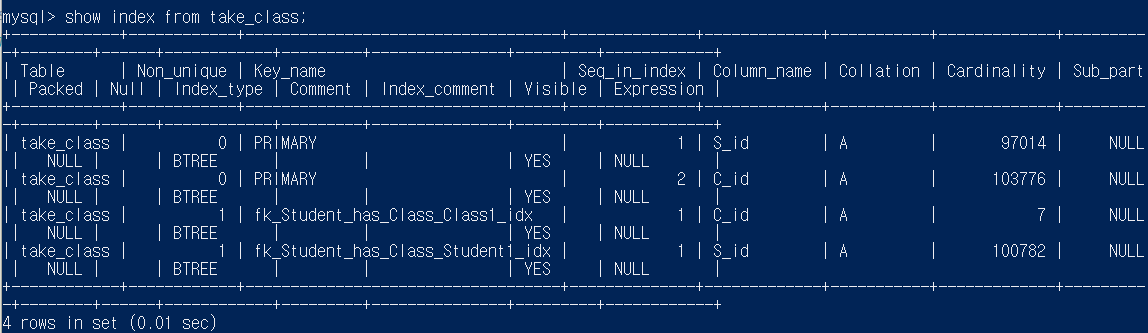


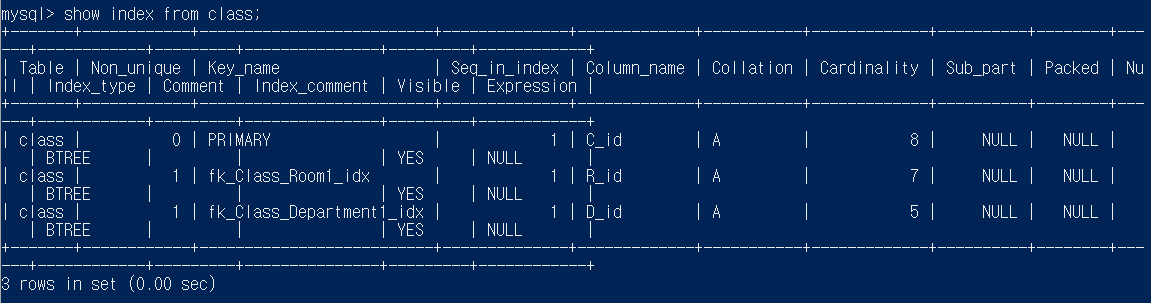
* test.sql 파일에 위와 같이 저장된 것을 확인하였고, 이를 데이터베이스에 삽입하였다.

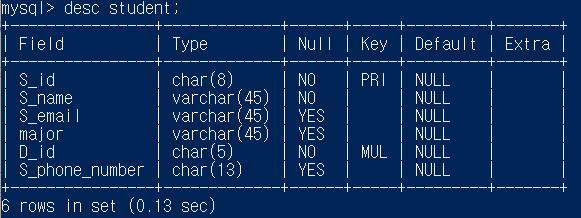
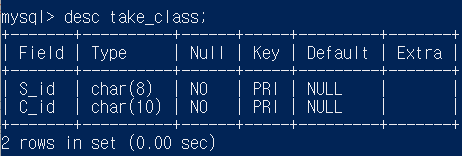
* 기존 데이터까지 포함하여 총 100,011개의 데이터가 student와 take\_class 테이블에 각각 존재함을 알 수 있다.

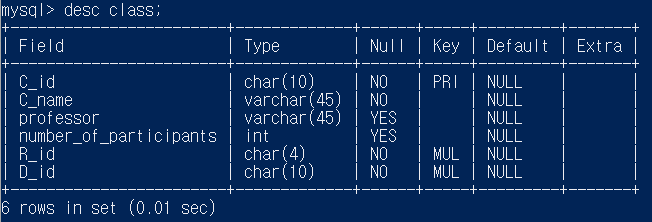




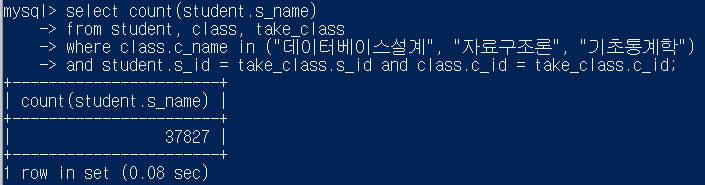


* 추가 index를 설정하기 전, student, take\_class, class 테이블의 index이다.

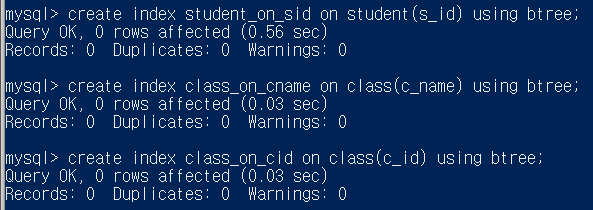
 



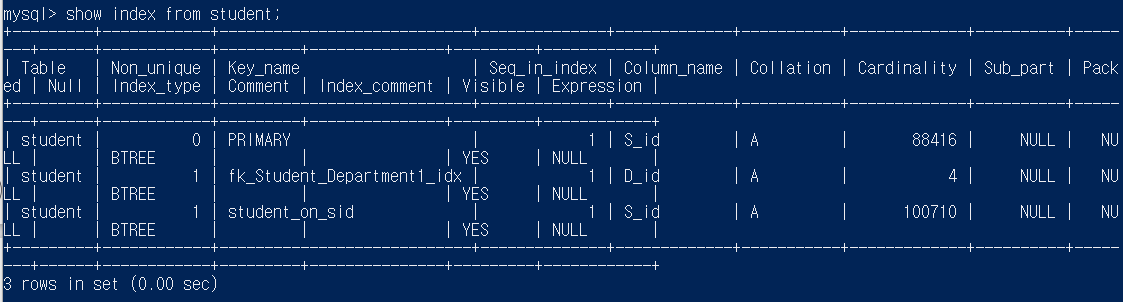
* 또한, 추가 index를 설정하기 이전 3개의 테이블에 대한 정보이다.

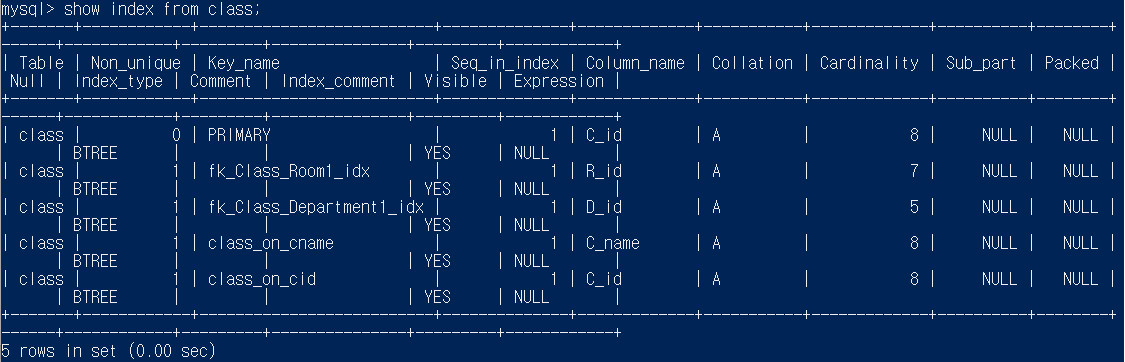


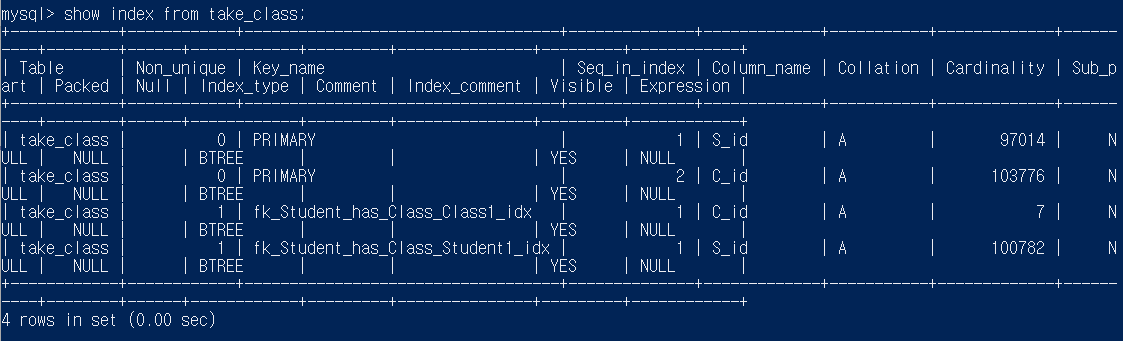
* index의 성능을 테스트하기 위한 query를 위와 같이 작성하였다. 특정 수업을 듣는 학생을 검색해야하므로, 수업의 이름이 데이터베이스설계, 자료구조론, 기초통계학인 경우의 학생 이름을 count하도록 하였다. 이를 실행하는 데, 0.08초가 소요되었다.



* 이전 select 문에서 조인 연산에 활용되는 student의 s\_id, class의 c\_id 2개의 column에 대해 각각 student\_on\_sid, class\_on\_cid의 이름으로 btree를 이용하는 index를 생성하였다. 또한, class의 c\_name에서 수업 이름을 탐색하는 조건이 있기 때문에 이에 대해 추가로 class\_on\_cname의 이름으로 index를 생성하였다. take\_class 테이블에 대해서 추가로 index를 설정하지 않은 이유는 이미 S\_id, C\_id 두 column에 대한 index가 설정되어있기 때문이다.



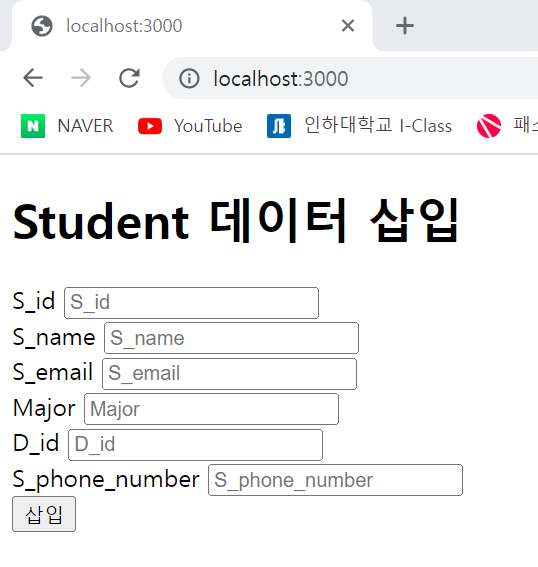




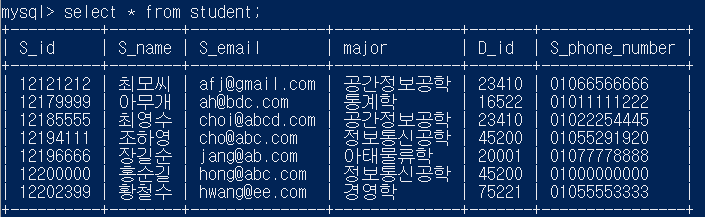
* 각 테이블에 대해 show index 명령어를 실행한 결과, 정상적으로 index가 생성되었음을 확인할 수 있다.

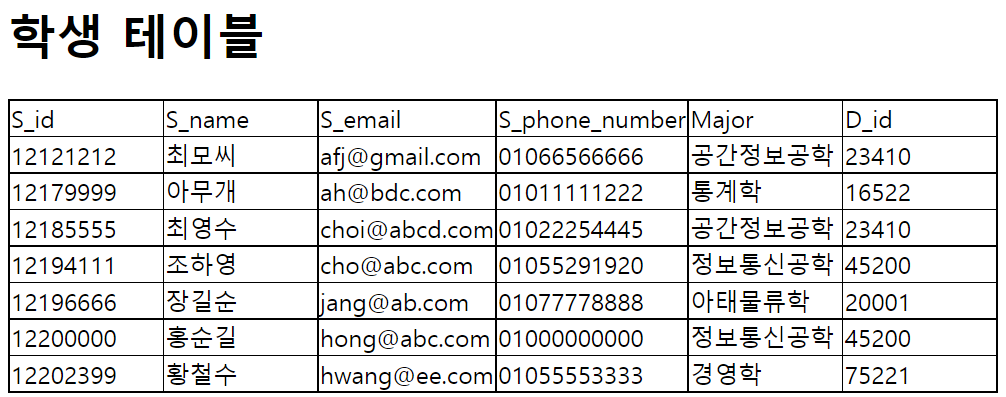
1. 실행 화면

**STEP 1. 웹에 INHA 데이터베이스 구현하기**



* 처음 localhost:3000에 접속했을 때의 home 화면으로, 데이터 삽입을 진행할 수 있다.



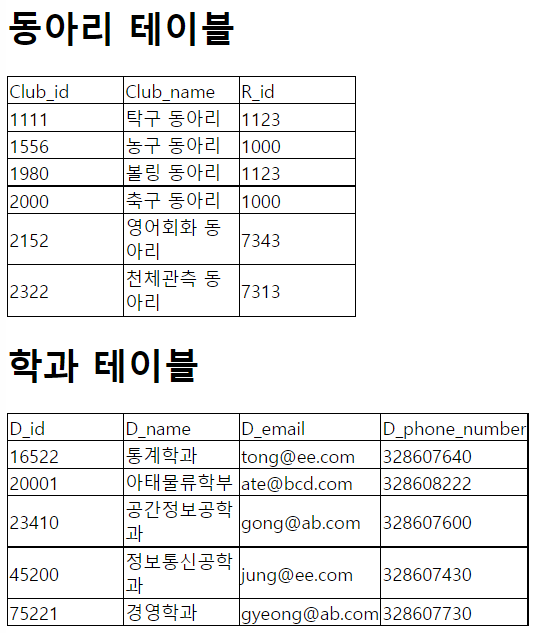


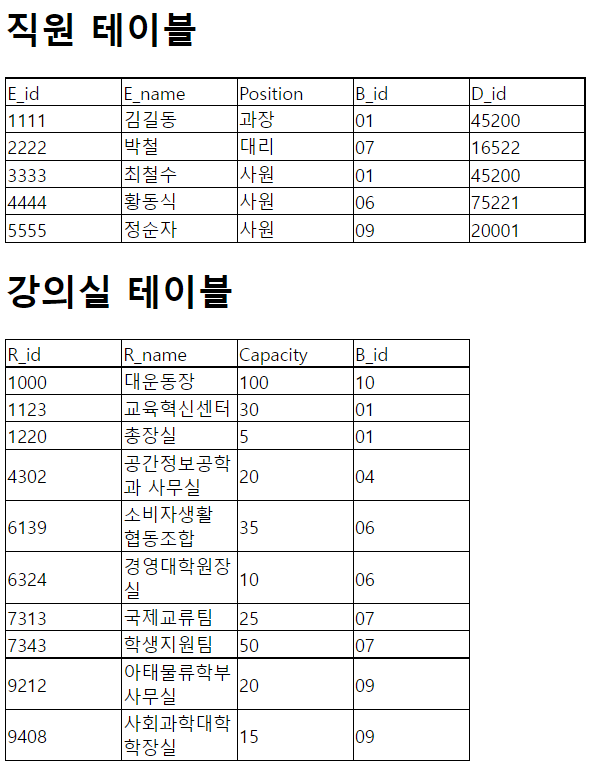
* 데이터 삽입 전 student 테이블의 인스턴스들이다. powershell과 웹 모두 동일한 데이터를 가지고 있다.

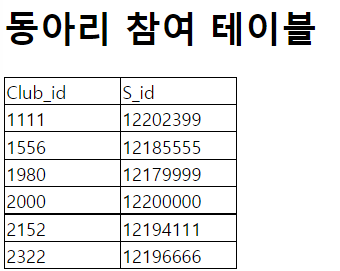




* 위와 같은 데이터를 삽입한 결과, 정상적으로 수행된 것을 확인할 수 있다.



* 또한, select 창에서 9개 테이블에 대한 모든 정보가 출력되는 것을 확인할 수 있다.



* update 창의 경우, 처음 접속하였을 때 위와 같은 형태를 보인다.

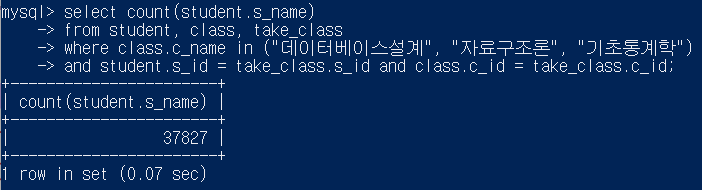


* 박모씨의 email, 전공, 학과 id, 핸드폰 전화 정보를 위와 같이 변경한다.

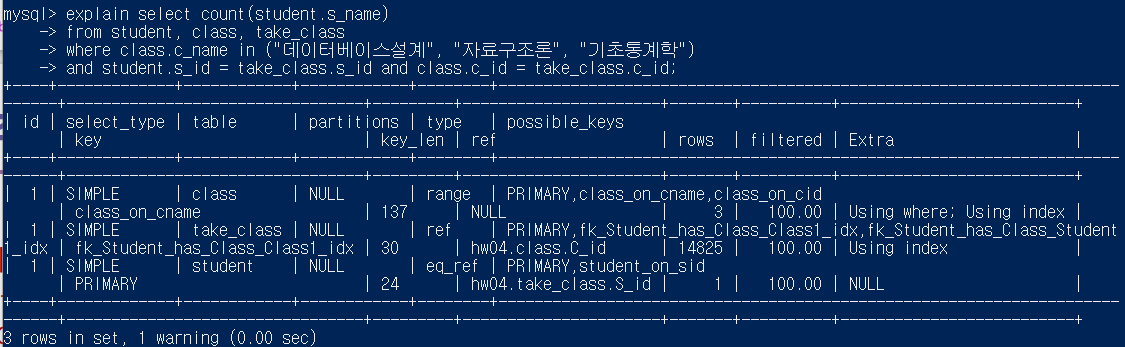


* 박모씨의 정보가 정상적으로 변경된 것을 확인할 수 있다.

**STEP 2. index 적용하기**



* 추가 index를 설정한 후, 특정 수업을 듣는 학생을 검색하는 동일한 select문을 실행하였다. 기존 소요 시간은 0.08초였으나, index 설정 이후 0.07초로 감소한 것을 알 수 있다.



* explain을 사용하여 select문 실행 시 index가 적용되었는지 확인하였다. 추가로 설정한 class\_on\_cname, class\_on\_cid, student\_on\_sid 3개의 index가 모두 possible\_keys의 목록에 존재하고, class\_on\_cid는 실제로 사용되었다.
* 즉, select문 소요 시간을 0.01초 감소시킨 데에 class\_on\_cid index가 기여하였음을 알 수 있다.

1. 결론

* 웹에 INHA 데이터베이스를 구현하는 첫 번째 문제의 경우, SQL과 서버 간의 데이터 교환이 정상적으로 이루어져 insert, select, update 모두 가능하였다.
* 특히, student 테이블에 대해서는 3가지 기능이 모두 적용되었는데, 새로운 인스턴스를 insert하면 INHA 데이터베이스에 바로 반영이 되었다. 또한, 기존 인스턴스의 정보를 수정하는 경우, 데이터베이스를 update하였는데, 이때 student의 s\_id를 where절에 조건으로 삼기 때문에 s\_id 정보를 변경하지 않더라도 반드시 기존 s\_id 정보를 입력해야한다는 주의사항이 있었다.
* 임의로 학생 10만명을 만들어서 index 적용 결과를 비교하는 두 번째 문제의 경우, index 3개를 추가 설정하기 이전과 이후의 소요 시간 차이가 0.01초로 비교적 적었다. 소요시간의 차이를 극대화하기 위해 다소 복잡한 select query를 작성하였음에도 불구하고 0.01초 차이가 나는 점에 대해서는 아쉬움을 느낀다. 다만, index 설정 이후 약 10번의 동일한 select query를 실행하였는데, 단 한 번도 0.08초 이상 소요된 경우는 없었으므로, index가 소요 시간을 감소시키는 데에 도움이 되었다고 확신할 수 있다.
* 또한, explain을 통해 index 적용 결과를 확인하는데, class\_on\_cname index가 mysql이 판단하였을 때 가장 적합하다고 생각되어 연산에 활용되었음을 알 수 있었다. where절로 수업의 이름을 검색하는 과정에서 사용된 것이다. 이때, class 테이블에서 c\_id와 c\_name을 다중 index로 설정하지 않고 각각 설정한 이유는 다중으로 c\_id는 조인을 위한 용도로, c\_name은 수업 이름 구분 용도로 각각 다르게 사용되기 때문이다.