

ICE4016 데이터베이스설계

<실습과제 12>

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.

2023년 11 월 25 일

학부 정보통신공학

학년 4

성명 박준형

학번 12191767



1. 개요

이번 실습에서는 이론 시간에 배운 인덱스라는 데이터베이스 테이블에서 데이터를 빠르게 검색하기 위해 사용되는 데이터 구조에 대해서 과제를 진행하였다.

인덱스는 테이블의 여러 열에 대한 정렬된 값 목록으로 구성되고 이런 목록을 사용하여 값을 빠르게 찾는데 도움이 된다. 검색을 하는 방법 알고리즘으로는 자료구조 시간에 배운 탐색 기법들이 사용되는데 해쉬, 비트리, 어레이 등이 있는데 터미널과 sql에서 기본적으로 제공하는 방식은 btree 방식이다.

이를 이용하여 쿼리의 성능이 향상되고 실행시간이 단축되는 이점을 가질 수 있으며 insert, update, delete 등이 가능하고 필요한 열에 대해 적절한 인덱스를 생성하고 유지와 관리를 해야한다.

그래서 이번 과제에서 성능을 알아보기 위하여 직접 인덱스를 생성해보고 많은 데이터 중에서 원하는 데이터를 얼마나 단축된 시간에 뽑아 낼 수 있는지 알아보았다. 이때 기존에 있는 데이터 양으로는 시간 차이가 매우 미미하여 알아보기 힘들기 때문에 많은 양의 데이터를 삽입 해야하는데 기존 방식으로 한줄씩 짜서 삽입하는 것에는 한계가 있기 때문에 과제에서는 10만명의 데이터를 삽입하라는 조건이 있는데 이를 파이썬의 faker 라이브러리를 사용하여 반복문을 이용하여 10만명의 데이터를 삽입해보고 10만명의 데이터를 기준으로 여러가지 작업을 인덱스 적용 전과 후를 비교해보며 어떤 차이점이 있는지 알아본다.

1. 상세 설계 내용

**STEP 1 : 10만명 이상의 학생을 INHA\_DB에 INSERT하는 쿼리문 작성 (언어 사용 제한 없음)**

파이썬의 faker라이브러리를 이용하여 10만명의 학생 데이터를 insert하는 쿼리문을 작성하였다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

처음에 faker와 mysql을 이용할 수 있게 해준다. Faker는 가상 데이터를 생성하기 위한 라이브러리고 mysql.connector는 파이썬 코드를 실행하면 바로 데이터에 적용이 되게 해준다.

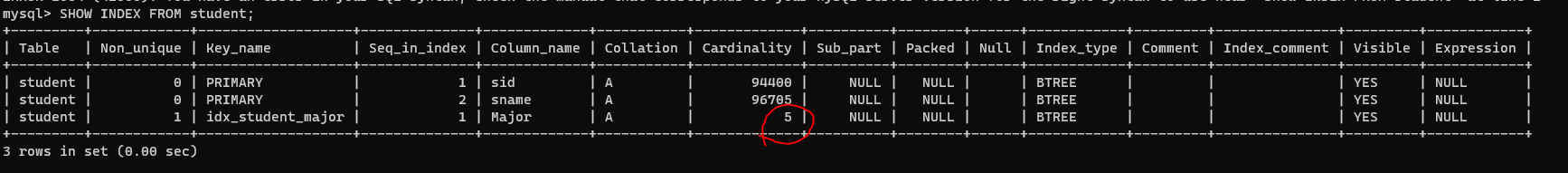
텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

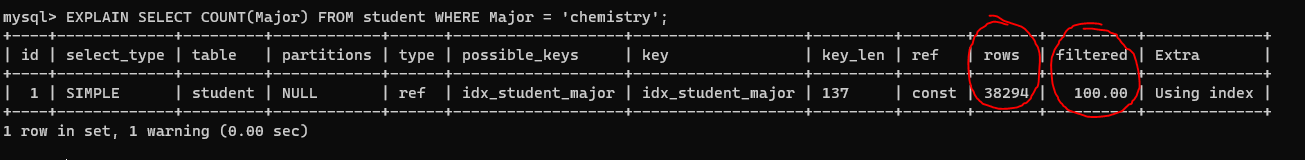
이후 insert문의 형식에 따라 쿼리문을 작성해주면 간단히 많은 양의 데이터를 생성해내고 데이터베이스에 적용해줄 수 있다. for \_ in range(100000)라는 반복문을 사용해서 10만개의 데이터를 생성하였다.

**STEP 2 : SHOW INDEX (Cardinality), EXPLAIN (rows, filtered) 조사**

Show index로 cardinality를 확인할 수 있고 이는 인덱스의 고유한 값을 나타낸다. 처음에 major에 관하여 인덱스를 생성하였기 때문에 이에 따른 show index로 CREATE INDEX idx\_student\_major ON student (Major); 라는 명령어로 확인을 해보면 과목이 5개 유형이 있기 때문에 cardinality가 5로 나온다.



explain에서는 explain select count(major) from student where major='chemistry';라는 명령어로 화학과의 학생들에 대한 설명을 살펴보았고 다음과 같다.



여기서 rows는 검색된 행 수로 쿼리 실행 중에 예상되는 검색된 행 수를 나타낸다. 여기서는 쿼리 결과로 38294개의 행이 검색되었으며 모든 행이 결과에 포함되었다는 걸 알 수 있다. 또한 filtered는 필터링된 비율로 조건을 거친 후 실제 결과에 대한 예상 비율을 나타낸다. 여기서 100은 필터링 없이 모든 행이 결과에 포함되었다는 것을 의미한다. 여기서 rows가 낮을수록 빠른 쿼리 실행이 가능하고 filtered의 값이 높을수록 필터링이 효과적으로 이루어졌다는 것을 알 수 있다.

또한 explain을 인덱스 생성 전과 후로 비교해보면 다음과 같다. extra에서 index를 사용했는지 안 했는지도 확인이 가능하다.

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

인덱스가 없을 때 major에 관한 explain을 보면 filtered가 10이고 rows는 103534이다. 이때 type은 all로 표시되어 있고 인덱스를 사용하지 않고 풀 테이블 스캔을 수행하는 모습이다. 그래서 rows가 103534로 전체 레코드를 전부 스캔한 모습이다. filtered값은 where 절의 조건에 따라 필터링 된 레코드 비율을 나타내며 이때 10퍼센트가 필터링 되었다.

그러나 index를 만든 후 explain결과에서는 type이 ref로 표시되고 possible keys로 만들어진 index가 표시되며 이는 조건문을 기반으로 인덱스를 사용하여 필요한 레코드만 찾아온다는 걸 의미한다. Rows는 39710으로 인덱스 생성 전보다 훨씬 적은 양의 필요한 레코드만 검색하고 필터도 100퍼센트로 해당 조건에 일치하는 모든 레코드를 필터링해서 결과에 지장은 없고 필요한 것만 검색하여 더 효율적인 작업을 진행하였다는 것을 의미한다.

결론적으로 이를 통하여 인덱스가 필요한 레코드만을 검색하여 결과를 찾아내므로 더 효율적인 쿼리실행을 진행하고 이러한 근거로 인하여 실행시간이 짧아진다는 것을 알아볼 수 있었다.

**STEP 3 : 10만명 이상의 학생을 INHA\_DB에 INSERT 한 뒤 5개 이상의 INDEX 적용 전후 비교 및**

**분석**

이제 이렇게 완성한 코드들을 바탕으로 실제로 실행하여 구현 설계를 한 내용과 같이 페이지가 잘 구현되었는지 확인해본다.

CREATE INDEX idx\_student\_major ON student (Major); 와 같이 한 attribute에 대하여도 인덱스를 만들어보는게 가능하고 CREATE INDEX idx\_student\_sname\_semail ON student (sname, semail);나 CREATE INDEX idx\_student\_major\_class\_id ON student (Major, class\_id); 처럼 복합적인 두가지 이상의 속성으로 인덱서를 생성하는 복합 인덱스 생성도 가능하다. 그렇게 생성하면 아래와 같이 show index로 확인이 가능하다.

텍스트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

이처럼 복합 인덱스는 인덱스 번호로 두가지 attribute를 보인다.

이렇게 만들어진 인덱스를 바탕으로 여러가지 검색을 5번 이상 진행해보며 나온 결과를 실행화면 부분에서 설명해주었다.

이러한 인덱스가 영향을 주는 관련된 검색 명령어를 통하여 얼마나 시간이 소요되는지 그리고 결과값은 정확하게 나오고 둘이 같게 나오는지 등을 적용하기 전과 후에 같은 명령어를 입력하고 살펴보았다. 예를 들면 위의 class\_id관련 인덱스를 활용하여 아래에서와 같이 해당 수업 아이디를 가지는 학생이 몇 명인지 찾는 명령어다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 실행 화면

**STEP 1 : 10만명 이상의 학생을 INHA\_DB에 INSERT하는 쿼리문 작성 (언어 사용 제한 없음)**

. 텍스트, 스크린샷, 메뉴, 폰트이(가) 표시된 사진

자동 생성된 설명

10만개의 데이터를 삽입하기 전의 데이터 모습이다. Student가 17개 있다는 것을 확인이 가능하다.

이제 파이썬에서 faker를 이용하여 10만개의 데이터를 생성하여 삽입시켜주고 실행해주면 select \* from student로 정보를 확인하기에는 계속해서 정보가 출력되기 때문에 count명령어를 통해서도 몇 개의 데이터가 있는지 확인해봤는데 다음과 같았다.

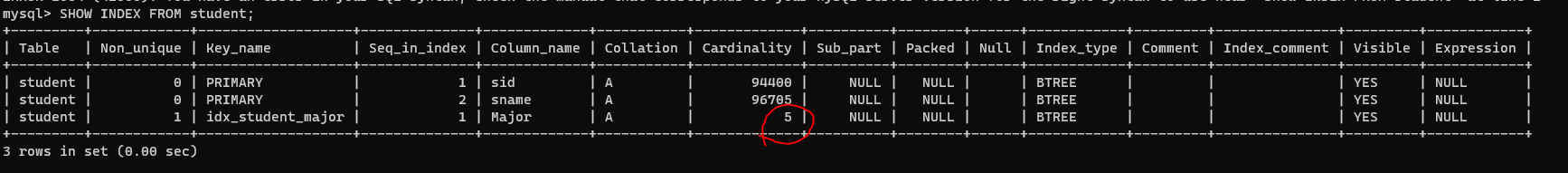
텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

이를 통해서 10만개의 학생 데이터가 추가되었다는 것을 확인 할 수 있다.

**STEP 2 : SHOW INDEX (Cardinality), EXPLAIN (rows, filtered) 조사**



스크린샷이(가) 표시된 사진

자동 생성된 설명

<Show index를 통한 cardinality 조사 결과 화면>

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

<explain을 통한 rows,filtered 조사 결과화면>

2번의 설계 단계에서 자세한 설명과 조사 내용을 적어 두었습니다.

**STEP 3 : 10만명 이상의 학생을 INHA\_DB에 INSERT 한 뒤 5개 이상의 INDEX 적용 전후 비교 및**

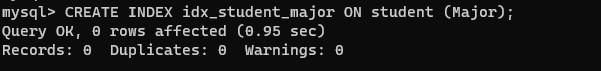
**분석**

먼저 index를 생성하기 전에 미리 chemistry 전공을 가진 학생이 10만명 중 몇 명인지 확인해보는 작업을 진행하였고 다음과 같이 0.03초가 걸리고 20048명이 있다는 것을 알 수 있었다.

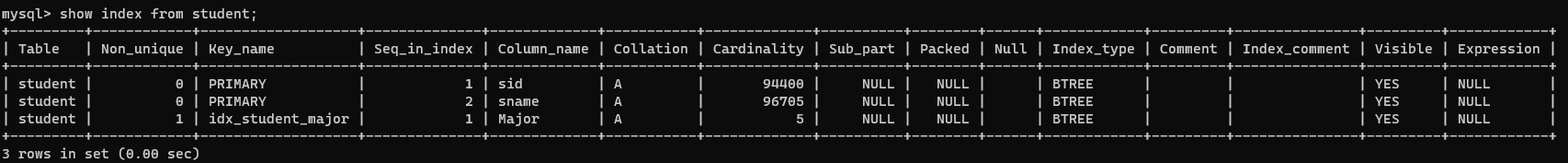
텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

이후 아래와 같이 major와 관련된 index를 만들어 보았다.



이는 show index를 통해서도 확인이 가능하다.



이후 변화를 보기 위하여 똑 같은 명령어를 실행해 보았다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

다른 과목에 대해서도 검색을 시도해보았고 이때도 시간은 역시나 더 짧아진 시간이였다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명🡪텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

왼쪽이 인덱스 생성 전이고 오른쪽이 인덱스 생성 후인데 결과 값은 똑같지만 걸린 시간은 계속해서 차이가 난다는 것을 알 수 있다. 이를 통해 소요시간이 3배나 단축된 모습을 확인할 수 있다. 이를 더 명확하게 하기 위하여 drop명령어를 통하여 index를 삭제하고 다시 해보면 다시 시간이 늘어난 것을 알 수 있다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

이렇게 명확하게 차이가 있다는 것을 확인할 수 있었다. 다른 index를 생성하여도 진행을 해보았다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명 🡪텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

인덱스 적용 전에는 왼쪽 처럼 뜨는데 CREATE INDEX idx\_student\_major\_class\_id ON student (Major, class\_id);를 통하여 class\_id에 관한 인덱스가 생성된 후를 보면 오른쪽 처럼 시간이 줄어든 것을 알 수 있다.

1. 결론

이번 과제에서 인덱스에 관한 실습을 진행하였는데 인덱스가 어떤 목적으로 사용되며 어떤 기능인지를 먼저 알아보고 인덱스를 만들고 검색을 진행하였을 때 와 만들기 전의 비교를 step3에서 하였고 step2에서는 그러한 인덱스 전후의 차이가 어떻게 발생하는지 알아보기 위하여 몇 가지 속성들을 살펴보고 explain을 통하여 검색을 한 쿼리문에 대한 정보를 알아볼 수 있었다. 또한 이러한 차이점들을 눈으로 잘 살피기 위하여 데이터를 많이 늘려야 했는데 이를 위하여 step1에서 10만명의 데이터를 삽입하는 쿼리문을 작성하는 방법도 익히는 연습을 하였다.

이론 시간에 배운 내용을 내가 이전까지 과제 하며 만들어 둔 디비에 데이터를 확장 시켜 넣어 확인해보고 실제로 시간 단축이 잘 이루어지고 정확하게 나오는지 그리고 어떻게 그러한 차이가 생기는지 원리를 알아볼 수 있는 과제였다.