중앙대학교 소프트웨어학부

혈액 보관소 DB

데이터베이스 시스템

20181807 김찬경
2023-5-12

프로젝트 중간 보고서

프로젝트 요약

보고 날짜	프로젝트 이름	작성자
2023 년 5 월 12 일	혈액 보관소 DB 시스템	김찬경(20181807)

개발 환경

운영체제: WINDOWS 11

개발 언어: JAVA

IDE: INTELLIJ IDEA

DB: 관계형 데이터베이스 사용(그 중 MySQL)

프로젝트 설명

이 프로젝트의 목표는 혈액 보관소들의 데이터를 모아놓은 혈액 보관소 DB 시스템을 구축하는 것이다. 이를 위해 4 가지의 기능(테이블 생성, 레코드 삽입, bitmap index 생성, bitmap index 를 이용한 질의 처리)을 구현할 예정인데 이들을 각각 text UI 형태에서 기능별 호출을 가능하게 구현하려고 한다.

설계, 구현 상에서 고려한 내용

1. 설정 및 가정한 내용

테이블은 하나로 설정하고 테이블 하나에 대량의 레코드를 삽입한다고 결정하였다.

테이블 스키마는 다음과 같이 설정했다.

serial_num	center_id	donor_id	donor_name	donor_gender	blood_type	donation_date

serial_num 을 제외한 모든 컬럼은 랜덤으로 설정할 계획이다. 그 이유는 대량의 데이터 삽입시 매 record 마다 값을 결정하기에는 힘들기 때문이다.

serial_num 은 일련의 번호로 B+ 트리 인덱싱을 위한 컬럼으로 일련번호 부여와 인덱싱을 위해 SERIAL 설정을 하였다. 또한 MySQL 에서는 primary key 에 대하여 기본적으로 B+ 트리 인덱싱을 적용하므로 PRIMARY KEY 설정을 하였다.

```
serial_num SERIAL PRIMARY KEY
```

center_id 는 보관소 ID 로 INT 형으로 잡았고 범위는 1 부터 100 까지로 설정하였다.

donor_id 는 기부자 ID 로 INT 형으로 잡았고 대량의 데이터를 삽입할 것이므로 1 부터 1000000 까지로 설정하였다.

donor_name 은 기부자 이름으로 VARCHAR 형태이며 최소 3 개의 문자에서 7 개의 문자까지로 설정하였다.

donor_gender 은 성별로 여기에 BITMAP INDEX 를 설정할 것이다. 표시는 'M', 'F'로 설정할 예정이다.

blood_type 은 혈액형으로 +, -도 구분해야하므로 총 8 개의 타입으로 { "A+", "A-", "B+", "B-", "O+", "O-", "AB+", "AB-"}로 설정하였다.

donation_date 은 헌혈한 날짜로 DATE 형태로 설정하였고 근 10 년동안의 날짜 중에서 고르도록 하였다. 그리하여 2013-5-7 부터 2023-5-7 일까지 중에서 고르도록 하였다.

그리하여 다음과 같이 설정하였다.

```
int centerId=rand.nextInt(100)+1;
int donorId=rand.nextInt(1000000)+1;
String donorName="";
for(int j=0;j<rand.nextInt(5)+3;j++){
    donorName+=(char)('a'+rand.nextInt(26));
}
String donorGender=genders[rand.nextInt(2)];
String bloodType=bloodTypes[rand.nextInt(8)];
```

```
long startDate = Date.valueOf("2013-05-07").getTime();
long endDate = Date.valueOf("2023-05-07").getTime();
long randomTime = startDate + (long) (rand.nextDouble() * (endDate - startDate));
Date donationDate= new Date(randomTime);
```

2. 구현 상 고려한 이슈들

I. 현실 세계 값과의 차이

현실 세계의 값은 굉장히 다양하기에 구현할 시 괴리가 클 수 밖에 없었고 최대한 현실 세계를 반영하려 값의 범위를 위처럼 설정하였고 다양성을 위해 랜덤하게 레코드를 삽입하도록 하였다.

II. 기부자 이름의 다양성 부족

랜덤하게 값을 삽입 시 정해진 리스트에서 뽑아 기부자 이름을 정하려 했으나 대량의 데이터를 삽입한다는 가정하에 다양성이 떨어질 것을 예상하여 위의 코드와 같이 여러 문자를 랜덤으로 정해 연결하는 방식으로 수정하여 다양성을 늘렸다.

III. 기부자가 같은데 기부자 이름, 성별, 혈액형이 다르게 삽입되는 경우

위에서 언급한 것처럼 모든 레코드의 내용이 랜덤으로 설정하기에 같은 기부자 ID 를 가지고 있음에도 불구하고 이름과 성별 그리고 혈액형이 다르게 설정되는 문제가 발생하였다. 이 문제를 해결하기위해 추가적으로 함수를 만들어 기부자 ID 가 같으면 기부자 이름, 성별, 혈액형을 같게 만들도록 설정하였다.

함수:

```
public static void updateDuplicateDonors(Connection conn) throws
SQLException {
    String sql="SELECT donor_id, donor_name, donor_gender,
blood_type FROM blood_donations GROUP BY donor_id HAVING COUNT(*)
> 1";
    PreparedStatement pstmt=conn.prepareStatement(sql);
    ResultSet rs=pstmt.executeQuery();

    while(rs.next()){
        int donorId=rs.getInt("donor_id");
        String donorName=rs.getString("donor_name");
        String donorGender=rs.getString("donor_gender");
```

```
        String bloodType=rs.getString("blood_type");
        sql="UPDATE blood_donations SET donor_name = ?,
donor_gender = ?, blood_type = ? WHERE donor_id = ?";
        pstmt=conn.prepareStatement(sql);
        pstmt.setString(1, donorName);
        pstmt.setString(2, donorGender);
        pstmt.setString(3, bloodType);
        pstmt.setInt(4, donorId);
        pstmt.executeUpdate();

    }
    System.out.println("Duplicate donors updated successfully!");
}
```

사용자가 UI 에서 레코드 삽입 기능을 실행하면 이 함수도 같이 호출되어 이러한 문제가 발생하지 않도록 바꾸었다.

이전:

```
mysql> select donor_id, donor_name, donor_gender, blood_type from blood_donations group by donor_id having count(*)>1;
+----------+------------+--------------+------------+
| donor_id | donor_name | donor_gender | blood_type |
+----------+------------+--------------+------------+
|    43087 | qrdv       | F            | A-         |
|   212610 | cnk        | F            | B+         |
|   643653 | drg        | M            | B-         |
|   597651 | gzs        | M            | A-         |
|   668181 | trqae      | F            | A-         |
|   890079 | vtzt       | M            | B-         |
+----------+------------+--------------+------------+
6 rows in set (0.00 sec)

mysql> select * from blood_donations where donor_id=43087;
+------------+-----------+----------+------------+--------------+------------+---------------+
| serial_num | center_id | donor_id | donor_name | donor_gender | blood_type | donation_date |
+------------+-----------+----------+------------+--------------+------------+---------------+
|        118 |        66 |    43087 | qrdv       | F            | A-         | 2022-09-21    |
|        481 |        12 |    43087 | sbgole     | M            | AB+        | 2016-08-29    |
+------------+-----------+----------+------------+--------------+------------+---------------+
2 rows in set (0.00 sec)
```

함수 추가하기 전 다음과 같이 같은 기부자 id 를 가지고 있음에도 불구하고 이름과 성별, 혈액형이 다른 것을 볼 수 있다.

이후:

```
mysql> select * from blood_donations where donor_id=43087;
+------------+-----------+----------+------------+--------------+------------+---------------+
| serial_num | center_id | donor_id | donor_name | donor_gender | blood_type | donation_date |
+------------+-----------+----------+------------+--------------+------------+---------------+
|        118 |        66 |    43087 | qrdv       | F            | A-         | 2022-09-21    |
|        481 |        12 |    43087 | qrdv       | F            | A-         | 2016-08-29    |
+------------+-----------+----------+------------+--------------+------------+---------------+
2 rows in set (0.00 sec)
```

함수 시행 후 통일된 것을 볼 수 있다.

IV. BITMAP INDEX 설정 시 어떤 컬럼에 대해 BITMAP INDEX 를 만들 것인지에 대한 고민

donor_gender 를 bitmap index 로 설정하기로 정하였다. 그 이유는 성별이 적은 수의 고유한 값을 갖기 때문에 비트맵 인덱스의 크기가 작아지고 검색 속도가 매우 빨라질 것이라 판단했기 때문이다.

일정

5/13~5/20 : 테이블 생성 및 삽입 기능 완성

5/21~5/28: bitmap index 생성 및 질의 처리 기능 완성

5/29~6/1: 검토 및 최종 레포트 작성