DBS Project 2023-1

소프트웨어학부 20204898 박소은

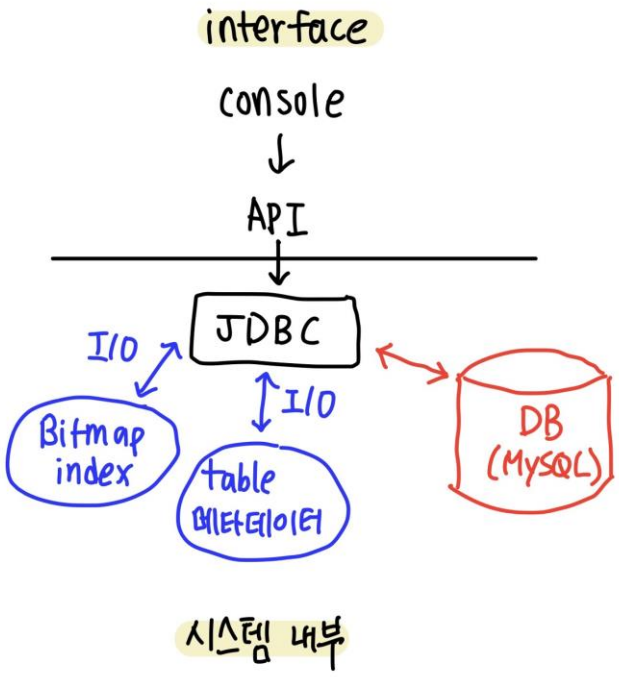
구현 언어 및 개발 환경

- 구현 언어: J ava 11.0.17
- Mysql 버전: 5.7.42
- IDE: IntelliJ IDEA 2022.3.1
- Runtime version: 17.0.5+1-b653.23 amd64
- OS: Window 11
- MySQL Connector J: mysql-connector-j-8.0.33.jar

설계 설명

- 주제: 고객 명단 리스트 관리 프로그램
- DBMS: MySQL 사용
- 테이블 스키마
  - id: INT (PK)
  - serialId: INT, 시스템 내부적으로 생성
  - name: VARCHAR(50) NOT NULL
  - gender: ENUM('male', 'female') NOT NULL
  - country: VARCHAR(50) NOT NULL
  - grade: INT NOT NULL CHECK (grade BETWEEN 1 AND 4)PK)

- 그림과 예시



- Implementation detail 및 이슈
  - Class
    - Customer: customer의 정보를 가지고 있는 클래스.
    - Bitmap: Bitmap index의 연산을 수행할 수 있는 클래스.
      - Controller 클래스에서 Bitmap 클래스의 인스턴스로 genderIndex, countryIndex, gradeIndex를 생성하여 사용한다.
      - Record 하나를 삽입하는 add 메소드, 다중질의문을 수행하는 processMultipleKeyQuery 메소드, 집계 질의문을 수행하는 processCountQuery 메소드를 가지고 있다.
    - JDBC: MySQL과 연결하여 SQL문을 실행시키는 클래스.
      - 레코드 하나를 삽입하는 insertRecord 메소드, B+tree index를 생성하는 createBPlusTreeIndex 메소드, Bitmap index와의 성능 비교를 위한 executeMultipleKeyQueryWithBPlusIndex 메소드와 executeGenderCountQueryWithBPlusIndex 메소드로 구성되어 있다.
    - Controller: Bitmap index를 관리하고 JDBC 인스턴스로 SQL문을 작동시키는 클래스이다.

- Bitmap Index를 사용할 때 필요한 serialId 변수를 관리한다.
- 자동으로 20000개의 튜플을 생성하는 addTuplesAuto 메소드를 가지고 있다.
- 그 외에도 multiple key query, count query를 수행하는 메소드와, Bitmap Index binary file을 읽고 쓰는 메소드를 가진다.

◆ Main: Scanner를 사용하여 input을 받아온다.

■ API 1. Record 삽입

레코드 하나를 생성하는 API이다.

레코드를 삽입할 때마다 Bitmap index를 업데이트할지, 마지막에 프로그램을 종료할 때 업데이트할지에 대한 이슈가 있었다.

레코드를 삽입할 때마다 업데이트하는 경우, file I/O가 많이 발생하여 성능이 낮아졌다.

따라서 마지막 종료 시점에 한 번에 write하는 것으로 수정하였다.

■ API 2. Multiple-key 질의

예를 들어 gender='f' AND country='Korea'에 관한 질의를 처리할 때, multiple key 질의를 Bitmap 클래스 속에서 처리할지, Controller에서 처리할지에 대한 문제가 있었다.

Map 자료구조를 사용하여 Map<Bitmap, String>과 같은 형태로 비트맵 인덱스와 쿼리를 전달하여, genderIndex 인스턴스에서 처리하도록 하였다.

for문 속에서 여러 bitset 인스턴스들을 and처리하였다.

■ API 3. 집계함수 질의

BitSet 클래스를 사용하여, 라이브러리에 내장되어 있는 기능인 cardinality() 메소드를 사용하여 구현하였다.

■ API 4. 튜플 자동생성

성능을 비교하기 위해 많은 양의 레코드를 생성할 수 있는 API이다. 한 번에 20,000개의 레코드가 생성된다.

구현 설명

- Bitmap 클래스의 processMultipleKeyQuery 메소드

```
1 usage  soeun
public Bitmap processMultipleKeyQuery(Map<Bitmap, String> queries) {
    BitSet result = null;

    for(Map.Entry<Bitmap, String> query: queries.entrySet()) {
        Bitmap queryBitmap = query.getKey();    // ex. "Korea"
        BitSet bitmapIndex = queryBitmap.getIndex().get(query.getValue());  // Korea의 index를 가져온다

        if (bitmapIndex == null) {
            // 만들어둔 index가 없는 경우
            result = null;
            break;
        } else {
            if (result == null) {
                // for문의 첫 번째 반복인 경우
                result = bitmapIndex;
            } else {
                BitSet bitset = (BitSet) result.clone();
                bitset.and(bitmapIndex);
                result = bitset;
            }
        }
    }

    if (result != null) {
        Bitmap resultBitmap = new Bitmap( bitmapName: "Multiple key");
        resultBitmap.getIndex().put("queryResult", result);
        return resultBitmap;
    }

    return new Bitmap( bitmapName: "Multiple key");
}
```

1. Parameter로 Bitmap과 QueryString을 Map 형태로 받는다.
2. Map을 for문으로 탐색하며 다음 과정을 반복한다
    A. Bitmap 인스턴스를 가져온다.
    B. 가져온 Bitmap 인스턴스를 누적해서 and연산을 수행한다.
3. 마지막으로 and연산을 수행한 Bitmap 결과를 반환한다.

- Bitmap 클래스의 processCountQuery 메소드

```
4 usages  soeun
public int processCountQuery(String key) {
    if(index.get(key) == null) {
        return 0;
    } else {
        int result = index.get(key).cardinality();
        return result;
    }
}
```

Parameter로 받은 key값으로 Bitmap을 cardinality() 메소드를 활용하여 연산한 결과를 반환한다.

- JDBC 클래스의 createBPlusTreeIndex

```java
public void createBPlusTreeIndex() {
    try {
        Statement stmt = con.createStatement();

        // idx_customer_gender 인덱스가 이미 존재하는지 확인
        ResultSet resultSet = stmt.executeQuery( sql: "SHOW INDEX FROM Customer WHERE Key_name = 'idx_customer_gender'
        if (resultSet.next()) {
            // 인덱스가 이미 존재하면 삭제
            stmt.executeUpdate( sql: "DROP INDEX idx_customer_gender ON Customer");
        }

        // idx_customer_country 인덱스가 이미 존재하는지 확인
        resultSet = stmt.executeQuery( sql: "SHOW INDEX FROM Customer WHERE Key_name = 'idx_customer_country'");
        if (resultSet.next()) {
            // 인덱스가 이미 존재하면 삭제
            stmt.executeUpdate( sql: "DROP INDEX idx_customer_country ON Customer");
        }

        // idx_customer_grade 인덱스가 이미 존재하는지 확인
        resultSet = stmt.executeQuery( sql: "SHOW INDEX FROM Customer WHERE Key_name = 'idx_customer_grade'");
        if (resultSet.next()) {
            // 인덱스가 이미 존재하면 삭제
            stmt.executeUpdate( sql: "DROP INDEX idx_customer_grade ON Customer");
        }

        // 새로운 인덱스 생성
        stmt.executeUpdate( sql: "CREATE INDEX idx_customer_gender ON Customer (gender) USING BTREE");
        stmt.executeUpdate( sql: "CREATE INDEX idx_customer_country ON Customer (country) USING BTREE");
        stmt.executeUpdate( sql: "CREATE INDEX idx_customer_grade ON Customer (grade) USING BTREE");


        System.out.println("B+tree index 생성 완료.");

    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

동일한 B+tree index를 생성하지 않도록 인덱스가 존재하는지 검사한 후, 인덱스가 이미 존재한다면 삭제하고 새로 B+tree index를 생성한다.

- JDBC 클래스의 executeMultipleKeyQueryWithBPlusIndex,

  executeGenderCountQueryWithBPlusIndex

```
1 usage  soeun
public void executeMultipleKeyQueryWithBPlusIndex(String sql) {
    try {
        long startTime = System.currentTimeMillis();

        Statement stmt = con.createStatement();
        stmt.executeQuery(sql);

        long endTime = System.currentTimeMillis();
        long executionTime = endTime - startTime;

        System.out.println("- B+tree index를 사용한 시간: " + executionTime + " ms");

    } catch (SQLException error) {
        error.printStackTrace();
    }

}
```

executeMultipleKeyQueryWithBPlusIndex: MySQL의 기능을 사용하여 Multiple key query를 수행한다. Bitmap index와의 결과 비교를 위하여 수행된다.

```
2 usages  soeun
public void executeGenderCountQueryWithBPlusIndex(String gender) {
    try {
        String sql = "SELECT COUNT(*) AS total_count FROM customer WHERE gender = ?";
        long startTime = System.currentTimeMillis();

        PreparedStatement pstmt = con.prepareStatement(sql);
        pstmt.setString( parameterIndex: 1, gender);
        ResultSet resultSet = pstmt.executeQuery();

        if (resultSet.next()) {
            int totalCount = resultSet.getInt( columnLabel: "total_count");
            System.out.println("Total Count(B+tree): " + totalCount);
        }

        long endTime = System.currentTimeMillis();
        long executionTime = endTime - startTime;

        System.out.println("B+tree index를 사용한 시간: " + executionTime + " ms");

    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

executeGenderCountQueryWithBPlusIndex 또한 성능 비교를 위하여 수행된다. Count query의 경우 gender에 관하여 성능 비교를 할 수 있다.

- Controller 클래스의 saveBitmapIndex, readBitmapIndex

```
3 usages  soeun
static void saveBitmapIndex(Bitmap bitmap, String path) {
    try (FileOutputStream fileOutputStream = new FileOutputStream(path)) {
        ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);
        objectOutputStream.writeObject(bitmap);
        System.out.println(path+": index 저장");

    } catch (IOException error) {
        System.out.println(error);
        error.printStackTrace();
    }
}

3 usages  soeun
static Bitmap readBitmapIndex(String path) {
    try (FileInputStream fileInputStream = new FileInputStream(path)) {
        ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);
        Bitmap bitmap = (Bitmap) objectInputStream.readObject();
        System.out.println(path+": index 읽어오기");
        return bitmap;
    } catch (IOException | ClassNotFoundException error) {
        System.out.println(error);
        error.printStackTrace();
    }
    return null;
}
```

프로그램이 처음 시작될 때 readBitmapIndex를 사용하여 Bitmap index binary file을 읽어오고, 종료될 때 saveBitmapIndex를 사용하여 binary file에 저장한다.

기능 동작 정확성 검증 결과

- Multiple key query

```
< 고객 관리 Application >
1. record 삽입
2. DB에 있는 bitmap index 확인
3. multiple-key 질의
4. 집계함수 count(*) 처리
5. 튜플 자동 생성(20000개의 튜플 생성)
6. 종료
숫자를 입력해주세요: 3
gender에 대한 질의를 생성하시겠습니까?
gender에 대한 질의 생성: 1 입력
1
SELECT *
FROM CUSTOMER
WHERE gender=?
gender='m'인 질의를 선택: 1 입력
gender='f'인 질의를 선택: 2 입력
2
country에 대한 질의를 생성하시겠습니까?
country에 대한 질의 생성: 1 입력
1
SELECT *
FROM CUSTOMER
WHERE country=?
WHERE절 안에 들어갈 country를 입력하세요: Korea
grade에 대한 질의를 생성하시겠습니까?
grade에 대한 질의 생성: 1 입력
2
grade에 대한 질의를 생성하지 않음.
< Result >
SELECT * FROM CUSTOMER WHERE gender='f' AND country="Korea"

- Bitmap index를 사용한 시간: 0 ms
- B+tree index를 사용한 시간: 24 ms

{0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 4
```

genderIndex, countryIndex, gradeIndex를 조합하여 multiple key query를 수행할 수 있다.

수행 결과, Bitmap index가 B+tree index에 비해 빠른 것을 확인할 수 있다.

- Count query

```
< 고객 관리 Application >
1. record 삽입
2. DB에 있는 bitmap index 확인
3. multiple-key 질의
4. 집계함수 count(*) 처리
5. 튜플 자동 생성(20000개의 튜플 생성)
6. 종료
숫자를 입력해주세요: 4
집계 함수를 실행할 column을 선택하세요.
1. gender(성별)
2. country(나라)
3. grade(고객 등급)
1
gender key값을 입력하세요: 남자는 1, 여자는 2를 선택
1
Total Count(Bitmap):10000
Bitmap index를 사용한 시간: 0 ms
Total Count(B+tree): 10000
B+tree index를 사용한 시간: 15 ms
```

Count query에 대하여 성능 비교를 한 결과, Bitmap index가 월등히 빠른 것을 확인할 수 있다.

- 새로운 고객 생성

```
< 고객 관리 Application >
1. record 삽입
2. DB에 있는 bitmap index 확인
3. multiple-key 질의
4. 집계함수 count(*) 처리
5. 튜플 자동 생성(20000개의 튜플 생성)
6. 종료
숫자를 입력해주세요: 1
< 생성할 고객 정보를 입력해주세요 >
고객 이름을 입력하세요: CustomerName
성별을 선택하세요: 남자는 1, 여자는 2를 입력
1
고객이 거주 중인 나라를 입력하세요: Japan
고객 등급을 입력하세요: 1~4 중에서 입력
2
이름: CustomerName, 성별: m, 국가: Japan, 고객 등급: 2
< 고객 생성 완료 >
```

실행 파일 생성 방법 설명

- MySQL을 local로 실행한 뒤, JDBC 클래스의 password 변수에 비밀번호를 입력한다.
- Bitmap Index binary file의 경우, 빈 bitmap이 저장되어 있는 binary file을 프로젝트 폴더 경로에 넣어주어야 한다.

  만약 binary file을 넣지 않은 경우, 다음과 같이 주석을 풀어 binary file을 초기화한다.

```
no usages   soeun *
public static void main(String[] args) {
//        Bitmap genderIndex = Controller.readBitmapIndex(Controller.genderIndexFileN
//        Bitmap countryIndex = Controller.readBitmapIndex(Controller.countryIndexFil
//        Bitmap gradeIndex = Controller.readBitmapIndex(Controller.gradeIndexFileNam
    Bitmap genderIndex = new Bitmap( bitmapName: "genderIndex");
    Bitmap countryIndex = new Bitmap( bitmapName: "countryIndex");
    Bitmap gradeIndex = new Bitmap( bitmapName: "gradeIndex");
```

- 처음 실행하면 테이블이 비어 있으므로, 5번을 클릭하여 튜플을 자동 생성한다.

```
< 고객 관리 Application >
1. record 삽입
2. DB에 있는 bitmap index 확인
3. multiple-key 질의
4. 집계함수 count(*) 처리
5. 튜플 자동 생성(20000개의 튜플 생성)
6. 종료
```

- External library로 mysql-connector-j-8.0.33.jar를 사용한다. (제출 파일에 함께 첨부)