데이터베이스시스템 설계과제 Ⅱ

20213563 김세연

1절. 데이터 및 테스트 시나리오

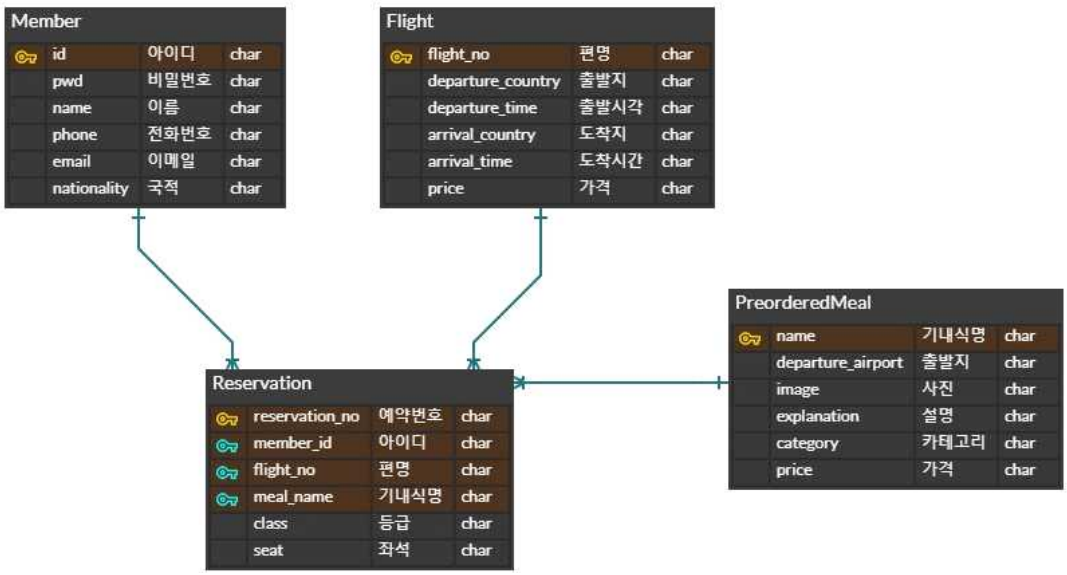
1. 선택 응용분야: 항공 예약 서비스
2. 생성할 테이블 스키마

Member(id, pwd, name, phone, email, nationality)

Flight(flight_no, departure_country, departure_time,
arriaval_country, arrival_time, price)

PreorderedMeal(name, departure_airport, image, explanation, category, price)

Reservation(reservation_no, member_id, flight_no, meal_name, class, seat)



3. 생성 테이블

1) member

ID	pwd	name	phone	email	nationality
traveler	ij4c8p	kim	01011112222	wlkbs@cau.ac.kr	Korea
dugod	3udsro	park	01012341234	q2f3hil@naver.com	Korea
vagabond	77naqb	lee	01008764982	09hsv@naver.com	Korea
rhror	ixlu1q	cho	01098723095	4hgip@gmail.com	Korea
whgdk	hj9urt	han	01029385092	pihsvd@icloud.com	Korea
member1	spa5nw	lim	01028736509	ihwvsp@naver.com	Korea
boat	8ke12m	jeong	01028764385	09svd@gmail.com	Korea
cnfqkf	pebswd	hwang	01009871301	0phva@gmail.com	Korea
resttime	1kw8g8	choi	01068463286	ohdv9@naver.com	Korea
home	crw487	kang	01098743287	p9h@naver.com	Korea
wave	osidvj	seo	01035908654	hyf9b@cau.ac.kr	Korea

2) reservation

reservation 테이블의 경우 member, flight, preorderedmeal의 값을 외래키로 하고 있으나, 테스트의 용이성을 위하여 flight_no, meal_name은 임의로 설정하여 튜플을 삽입하였다.

reservation_no	member_id	flight_no	meal_name	class	seat
ASW430	whgdk	P2465	baby	economy	j08
EAO249	dugod	A4503	bibimbap	economy	k03
DAR634	traveler	P2819	seafood	business	d03
AER398	boat	S7048	cupramen	economy	k01
SDF359	home	W3024	low-sodium	first	a01
ERB451	member1	R3909	diabetic	business	e04
REN349	wave	B6039	vegan	economy	p06

4. 테스트 시나리오

1) 테이블 생성: member, reservation

2) 테이블별 튜플 삽입

3) 조인

'select * from member, reservation where member.id = reservation.member_id'의 연산을 진행하고, 결과를 출력한다.

2절. 설계

1. 구현 세부사항

1) 설계과제1의 수정

ㄱ. 코드의 재사용성을 위한 리팩토링

- DBManager를 MetadataManager로 이름을 변경하여 메타데이터를 관리하는 객체임을 명시했다.
- MemberManager를 DatabaseManager로 이름을 변경하여 타 테이블을 생성하고, 데이터를 삽입과 삭제하는 데 활용할 수 있도록 하였다.

ㄴ. 튜플 삽입 시 입력받은 데이터를 컬럼별로 파싱하는 과정에서
split(". ")에서 split(",").strip()로 변경하여 입력의 제한을 줄였다.

ㄷ. block단위 IO 구현 변경: readline()에서 버퍼선언+filereader.read(buffer, 0, len)으로 변경하여 원하는 블록만 가져오도록 구성하였다.

2) 해시함수

ㄱ. 파티셔닝을 위한 해시함수

: 컬럼의 byte로 치환한 후 모두 더하여 mod 5하여 구하였다.

ㄴ. join 짝을 찾기 위한 해시함수

: .hashCode() 함수를 사용하여 hash값을 구한 후, mod 3한 양수 값을 사용했다.

3) 파티셔닝 파일 저장 위치

프로젝트 폴더 아래에 **tmp 폴더를 미리 생성**한 후, 해당 폴더에 테이블명0 ~ 테이블명4의 파일을 생성한다. 버킷의 블록은 2로 설정하여 버킷이 찬 경우 디스크로 쓰도록 구성한다.

4) hash index 구성

columnValue와 index를 가지는 BuchetStructure 클래스를 선언하고 리스트를 선언하였다. 해시 결과를 버킷의 index로 사용하여, 비교하고자 하는 column의 값과 몇 번째 record인지 그 index를 저장한다.

2. 인터페이스 설계

```
Select a command you want to execute.
(1.CREATE TABLE  2.INSERT  3.DELETE  4.SELECT  5.JOIN  6.EXIT)
>> 5
Enter first table for the record to be selected.
>> member
Enter the column of the first table.
>> id
Enter second table for the record to be selected.
>> reservation
Enter the column of the second table.
>> member_id
```

3절. 구현

```
2 usages
void partitioning(String tableName, String column) throws IOException {
    //임시 파일 생성
    for(int i=0; i<5; i++) {
        FileOutputStream fos = new FileOutputStream( name: System.getProperty("user.dir") + "\\tmp\\" + tableName + i + ".txt");
        fos.close();
    }
```

파티셔닝: tmp 폴더 아래에 테이블 이름0 ~ 테이블이름4 파일을 생성한다.
이때, 구현 세부사항에 기술한 것처럼, tmp 폴더를 미리 생성해야 오류가 발생하지 않는다.

```
BufferedReader fr = new BufferedReader(new FileReader(location),  sz: length*3);

char[] buffer = new char[length*3];
while(true) {
    int charsRead = fr.read(buffer,  off: 0,  len: length*3);
    if(charsRead == -1)
        break;

    String block2 = new String(buffer,  offset: 0, charsRead);
    String[] subStringArray = block2.split( regex: "(?<=\\G.{" + length + "})");
    for(String string : subStringArray){
```

테이블에서 한 블록씩 가져온다.

```
String str = string;
String CheckColumn = "";
for(MetadataManager.AttributesInformation information: attributesInformations) {
    if (information.getAttribute().toString().equals(column)) {
        //column 파싱 및 해시 적용
    }
    else {
        CheckColumn = str.substring(information.getLength());
        str = CheckColumn;
    }
}
```

CheckColumn을 사용하여 join 조건의 column인지 확인한다.

```
if (information.getAttribute().toString().equals(column)) {
    //column 파싱 및 해시 적용
    String columnValue = CheckColumn.substring(0, information.getLength().intValue()).trim();
    columnValue = Arrays.toString(columnValue.getBytes());
    int sum = Arrays.stream(columnValue.substring(1, columnValue.length() - 1).split( regex: ",")) Stream<String>
            .map(String::trim)
            .mapToInt(Integer::parseInt) IntStream
            .sum();
    int hash = sum%5;

    //해당 버퍼에 저장
    bucket[hash].add(string);
    //버퍼가 가득차면 디스크로 씀
    if(bucket[hash].size() == 2){
        try {
            String toWrite = "";
            for (String elements: bucket[hash]){
                toWrite = toWrite+elements;
            }
            FileWriter writer = new FileWriter( fileName: System.getProperty("user.dir") + "\\tmp\\" + tableName + hash + ".txt",  append: true);
            writer.write(toWrite);
            writer.close();
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
        bucket[hash].clear();
    }

    //다음 컬럼부터는 확인하지 않음
    break;
}
```

join 조건의 column인 것이 확인되면, 해당 column의 값에 대하여 해시함수를 적용한다. 도출된 해시값을 인덱스로 하여 버킷에 column값을 저장한다. 버킷이 가득 차면 디스크에 쓰고

버킷을 초기화한다.

파티셔닝이 끝나면 join짝을 찾는다. join짝을 찾을 때 사용할 hash index를 구현하기 위하여 column값과 index를 필드로 하는 클래스를 선언하였다.

```
public class BucketStructure{
    2 usages
    String columnValue;

    2 usages
    Integer index;

    2 usages
    public String getColumnValue() { return columnValue; }

    2 usages
    public void setColumnValue(String columnValue) { this.columnValue = columnValue; }

    2 usages
    public Integer getIndex() { return index; }

    2 usages
    public void setIndex(Integer index) { this.index = index; }
}
```

join짝을 찾을 때 BucketStructure의 리스트를 선언하고 초기화하며 시작한다.

```
for(int i=0; i<5; i++){
    //build input, probe input 결정
    String location1 = System.getProperty("user.dir") + "\\tmp\\" + table1 + i + ".txt";
    Path path1 = Paths.get(location1);
    String location2 = System.getProperty("user.dir") + "\\tmp\\" + table2 + i + ".txt";
    Path path2 = Paths.get(location2);

    if(Files.size(path1) < Files.size(path2)) {
```

각 파티션에 대하여 작은 쪽을 build input으로 사용하기 위하여 사이즈를 비교한다.

```
if(Files.size(path1) < Files.size(path2)) {
    //build input은 전부 메모리에 적재
    List<String> buildInput = new ArrayList<>();
    BufferedReader fr = new BufferedReader(new FileReader(location1),  sz: length1 * 2);

    char[] buffer = new char[length1 * 2];
    while (true) {
        int charsRead = fr.read(buffer,  off: 0,  len: length1 * 2);
        if (charsRead == -1)
            break;

        String block1 = new String(buffer,  offset: 0, charsRead);
        for (int j = 0; j < block1.length(); j += length1) {
            buildInput.add(block1.substring(j, j + length1));
        }
    }
    fr.close();
```

build input의 경우 모두 메모리에 적재한다. 한 블록씩 가져와 row로 나누어 리스트에 추가하여 index를 활용할 수 있도록 하였다.

```
//hash index 구성
for (int a=0; a< buildInput.size(); a++) {
    //column parsing
    String str = buildInput.get(a);
    String CheckColumn = "";
    for (MetadataManager.AttributesInformation information : attributesInformations1) {
        if (information.getAttribute().toString().equals(column1)) {
            //column 파싱 및 해시 적용
            String columnValue = CheckColumn.substring(0, information.getLength().intValue()).trim();
            //columnValue = Arrays.toString(columnValue.getBytes());
            int hash = (columnValue.hashCode() % 3 + 3) % 3;

            BucketStructure bucketStructure = new BucketStructure();
            bucketStructure.setColumnValue(columnValue);
            bucketStructure.setIndex(a);

            bucket[hash].add(bucketStructure);
        } else {
            CheckColumn = str.substring(information.getLength());
            str = CheckColumn;
        }
    }
}
```

파티셔닝과 비슷한 방법으로 해시를 구하여 인덱스로 사용하고, 해당 column값과 row의 인덱스를 버킷에 저장한다.

hash index를 모두 구성하고 나면 probe input을 한 블록씩 가져와 row에 대하여 검증을 진행한다. 해당 row의 column값을 파싱하고 해시함수를 적용하여 버킷에 접근한다. 해당 버

```
fr = new BufferedReader(new FileReader(location2),  sz: length2 * 2);

buffer = new char[length2 * 2];
while (true) {
    int charsRead = fr.read(buffer,  off: 0,  len: length2 * 2);
    if (charsRead != -1) {
        //probe input은 한 블록씩 가져와서 hash index에 동일한 내용이 있는지 확인
        String block2 = new String(buffer,  offset: 0, charsRead);
        for (int j = 0; j < block2.length(); j += length2) {
            String probeInput = block2.substring(j, j + length2);

            //column parsing
            String str = probeInput;
            String CheckColumn = "";
            for (MetadataManager.AttributesInformation information : attributesInformations2) {
                if (information.getAttribute().toString().equals(column2)) {
                    //column 파싱 및 해시 적용
                    String columnValue = CheckColumn.substring(0, information.getLength().intValue()).trim();
                    //columnValue = Arrays.toString(columnValue.getBytes());
                    int hash = (columnValue.hashCode() % 3 + 3) % 3;

                    //해시 인덱스를 사용하여 동일한 값이 있는지 확인
                    for (BucketStructure bucketStructure : bucket[hash]) {
                        //동일한 값이 있으면 출력
```

킷에 column값과 동일한 값이 있을 경우 join 결과를 출력한다.

```
//동일한 값이 있으면 출력
if (columnValue.equals(bucketStructure.getColumnValue())) {
    String buildinput = buildInput.get(bucketStructure.getIndex());
    for(MetadataManager.AttributesInformation information1: attributesInformations1){
        System.out.print("|" + buildinput.substring(0, information1.getLength()));
        buildinput = buildinput.substring(information1.getLength());
    }
    for(MetadataManager.AttributesInformation information2: attributesInformations2){
        System.out.print("|" + probeInput.substring(0, information2.getLength()));
        probeInput = probeInput.substring(information2.getLength());
    }
    System.out.println("|");
}
```

출력은 'select *'를 기준으로 하여 모든 column에 대한 값을 출력한다. column에 대한 길이값을 메타데이터를 통해 확인하고 포맷을 맞추어 출력한다.

table2를 build input으로 사용하는 경우에도 로직은 동일하나, 출력 포맷을 맞추기 위하여 함수를 사용하지 않고 동일한 코드에서 table1과 관련된 변수의 table2와 관련된 변수의 위치를 치환하였다.

4절. 테스트 및 정확성 검증

```
Select a command you want to execute.
(1.CREATE TABLE  2.INSERT  3.DELETE  4.SELECT  5.JOIN  6.EXIT)
>> 4
Enter a table for the record to be selected.
>> member
Select an option.
(1. Select 1 record.  2. Select all records)
>> 2
|email               |id            |name      |nationality|phone          |pwd       |
|hyf9b@cau.ac.kr     |wave          |seo       |Korea      |01035908654    |osidvj    |
|p9h@naver.com       |home          |kang      |Korea      |01098743287    |crw487    |
|ohdv9@naver.com     |resttime      |choi      |Japan      |01068463286    |1kw8g8    |
|0phva@gmail.com     |cnfqkf        |hwang     |France     |01009871301    |pebswd    |
|09svd@gmail.com     |boat          |jeong     |Australia  |01028764385    |8ke12m    |
|ihwvsp@naver.com    |member1       |lim       |Singapore  |01028736509    |spa5nw    |
|pihsvd@icloud.com   |whgdk         |han       |Korea      |01029385092    |hj9urt    |
|4hgip@gmail.com     |rhror         |cho       |Korea      |01098723095    |ixlu1q    |
|09hsv@naver.com     |vagabond      |lee       |Korea      |01008764982    |77naqb    |
|q2f3hil@naver.com   |dugod         |park      |Korea      |01012341234    |3udsro    |
|wlkbs@cau.ac.kr     |traveler      |kim       |Korea      |01011112222    |ij4c8p    |
```

member table

```
Select a command you want to execute.
(1.CREATE TABLE  2.INSERT  3.DELETE  4.SELECT  5.JOIN  6.EXIT)
>> 4
Enter a table for the record to be selected.
>> reservation
Select an option.
(1. Select 1 record.  2. Select all records)
>> 2
|class     |flight_no |meal_name          |member_id      |reservation_no|seat |
|economy   |P2465     |baby               |whgdk          |ASW430    |j08  |
|economy   |A4503     |bibimbap           |dugod          |EA0249    |k03  |
|business  |P2819     |seafood            |traveler       |DAR634    |d03  |
|economy   |S7048     |cupramen           |boat           |AER398    |k01  |
|first     |W3024     |low-sodium         |home           |SDF359    |a01  |
|business  |R3909     |diabetic           |member1        |ERB451    |e04  |
|economy   |B6039     |vegan              |wave           |REN349    |p06  |
```

reservation table

```
Select a command you want to execute.
(1.CREATE TABLE  2.INSERT  3.DELETE  4.SELECT  5.JOIN  6.EXIT)
>> 5
Enter first table for the record to be selected.
>> member
Enter the column of the first table.
>> id
Enter second table for the record to be selected.
>> reservation
Enter the column of the second table.
>> member_id
|email              |id            |name      |nationality|phone          |pwd      |class     |flight_no |meal_name          |member_id      |reservation_no|seat |
|p9h@naver.com      |home          |kang      |Korea      |01098743287    |crw487   |first     |W3024     |low-sodium         |home           |SDF359    |a01  |
|hyf9b@cau.ac.kr    |wave          |seo       |Korea      |01035908654    |osidvj   |economy   |B6039     |vegan              |wave           |REN349    |p06  |
|q2f3hil@naver.com  |dugod         |park      |Korea      |01012341234    |3udsro   |economy   |A4503     |bibimbap           |dugod          |EA0249    |k03  |
|ihwvsp@naver.com   |member1       |lim       |Singapore  |01028736509    |spa5nw   |business  |R3909     |diabetic           |member1        |ERB451    |e04  |
|09svd@gmail.com    |boat          |jeong     |Australia  |01028764385    |8ke12m   |economy   |S7048     |cupramen           |boat           |AER398    |k01  |
|pihsvd@icloud.com  |whgdk         |han       |Korea      |01029385092    |hj9urt   |economy   |P2465     |baby               |whgdk          |ASW430    |j08  |
|wlkbs@cau.ac.kr    |traveler      |kim       |Korea      |01011112222    |ij4c8p   |business  |P2819     |seafood            |traveler       |DAR634    |d03  |
```

join 결과

5절. Java/JDBC 프로그램 소스코드 전체