주제는 온라인 티켓 예매처이다. 설계 변경사항에 대해 먼저 논의한다. 첫 번째 설계 변경사항은 속성 항목이다. 설계 당시에는 속성이 고객의 아이디, 이름, 성별, 생년월일, 휴대폰번호, 공연제목, 날짜, 시간, 좌석 구역, 좌석 번호, 수령 방법이었다. 그러나, 직접 테이블을 제작하면서 대용량 테이블을 만들면서 모든 항목을 랜덤으로 작성하는데 간소화 하기 위해 최종적으로 고객의 인덱스, 이름, 주민번호 앞 6자리, 공연 종류, 공연 제목, 공연 날짜, 좌석 구역, 좌석 번호, 수령 방법으로 정하였다.
다음은 bitmap Index 컬럼이다. 설계 당시에는 고객이 특정날짜의 공연을 예매했는지 확인하는 것과 구역별 예매 인원수로 구상했는데 실제로 bitmap Index를 제작하면서 컬럼의 distinct 한 값이 너무 많아서 바꾸기로 하였다. 총 3개의 비트맵 인덱스를 만드는데, 컬럼은 seat_Area 컬럼, performance_type 컬럼, delivery_method 컬럼으로 정했다. 다음은 구현 설명이다.
class는 Area_bitmap.java, delivery_bitmap.java, ForProgram.java, QueryProgramming.java, type_bitmap.java로 총 5개이다. 다음은 각 코드에 대한 구현설명이다. 문서에 제시된 대로 주요 부분의 소스코드를 설명한다.
먼저, ForProgram.java이다. 테이블 생성, 레코드 삽입, bitmap Index 생성을 한 군데에 모아서 실행을 담당하는 코드이다.

```java
try {
    Class.forName( className: "com.mysql.cj.jdbc.Driver");   // JDBC 드라이버 로드
    Connection connection = DriverManager.getConnection(url, username, password); // 데이터베이스 연결
    Scanner sc = new Scanner(System.in);
    while (true) {
        System.out.println("1. Create Table");
        System.out.println("2. Insert Records");
        System.out.println("3. Create Bitmap index");
        System.out.println("0. Exit");
        System.out.println("Choose work>>" );

        int choice = sc.nextInt();

        if (choice == 1) { // 테이블 생성
            createTable(connection);
        } else if (choice == 2) { // 레코드 삽입
            System.out.println("Enter the record number>> ");
            int recordCount = sc.nextInt();
            insertRecords(connection, names, performanceTypes, performanceTitles, seatAreas, deliveryMethods, random, recordCount); // 레코드 삽입
        } else if (choice == 3){ // 비트맵 인덱스 생성 및 텍스트 파일 저장
            area.createAndSaveBitmapIndexes();
            deli.createAndSaveBitmapIndexes();
            type.createAndSaveBitmapIndexes();
        } else if (choice == 0) {
            break;
        } else {
            System.out.println("Wrong input.");
        }
```

```
ForProgram
C:\Users\USER\.jdks\openjdk-20\bin\
1. Create Table
2. Insert Records
3. Create Bitmap index
0. Exit
Choose work>>
1
Success to create table
1. Create Table
2. Insert Records
3. Create Bitmap index
0. Exit
Choose work>>
2
Enter the record number>>
100000
Success to insert records
1. Create Table
2. Insert Records
3. Create Bitmap index
0. Exit
Choose work>>
0
Exit program
```

입력을 받는 화면이다. 콘솔창은 오른쪽 그림처럼 등장한다.

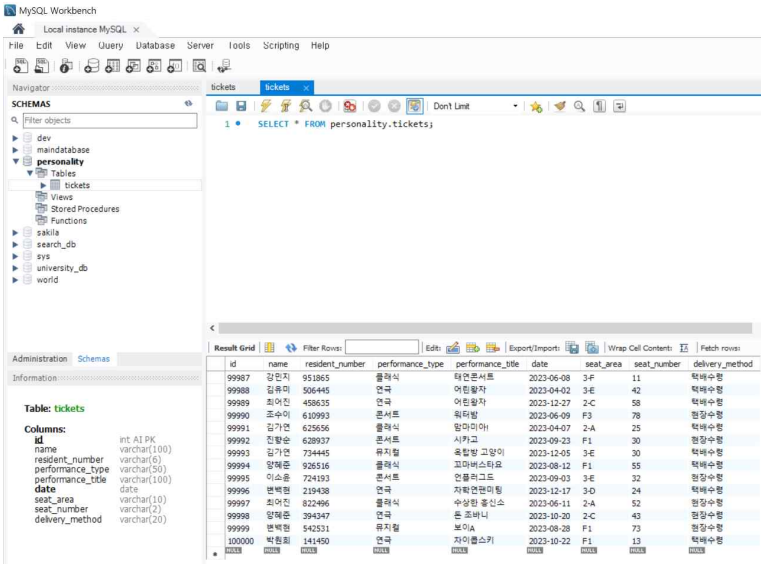
```java
private static void createTable(Connection connection) throws SQLException {
    String createTableQuery = "CREATE TABLE tickets (" +
            "id INT PRIMARY KEY AUTO_INCREMENT," +
            "name VARCHAR(100) ," +
            "resident_number VARCHAR(6)," +
            "performance_type VARCHAR(50) ," +
            "performance_title VARCHAR(100)," +
            "date DATE," +
            "seat_area VARCHAR(10) ," +
            "seat_number VARCHAR(2)," +
            "delivery_method VARCHAR(20)" +
            ")";
    Statement statement = connection.createStatement();
    statement.executeUpdate(createTableQuery);
    System.out.println("Success to create table");
}
```

위의 코드는 테이블 생성을 위한 코드이다. SQL문으로 작성되었고, 각 속성당 형태도 지정해주어 이대로 MySQL Workbench에서 실행되어 테이블이 생성된다.

```java
private static void insertRecords(Connection connection, String[] names, String[] performanceTypes, String[] performanceTitles,
                                  String[] seatAreas, String[] deliveryMethods, Random random, int recordCount) throws SQLException {
    String insertQuery = "INSERT INTO tickets (name, resident_number, performance_type, performance_title, date, seat_area," +
            "seat_number, delivery_method) VALUES (?, ?, ?, ?, ?, ?, ?)";
    PreparedStatement preparedStatement = connection.prepareStatement(insertQuery);

    LocalDate startDate = LocalDate.of( year: 2023,  month: 1,  dayOfMonth: 1);
    LocalDate endDate = LocalDate.of( year: 2023,  month: 12,  dayOfMonth: 31);

    // 레코드를 반복적으로 삽입
    for (int i = 0; i < recordCount; i++) {
        // 랜덤한 데이터 생성
        String name = names[random.nextInt(names.length)];
        String residentNumber = String.valueOf( i: random.nextInt( bound: 900000) + 100000);
        String performanceType = performanceTypes[random.nextInt(performanceTypes.length)];
        String performanceTitle = performanceTitles[random.nextInt(performanceTitles.length)];
        String date = generateRandomDate(startDate, endDate);
        String seatArea = seatAreas[random.nextInt(seatAreas.length)];
        String seatNumber = String.valueOf( i: random.nextInt( bound: 99) + 1);
        String deliveryMethod = deliveryMethods[random.nextInt(deliveryMethods.length)];

        preparedStatement.setString( parameterIndex: 1, name);
        preparedStatement.setString( parameterIndex: 2, residentNumber);
        preparedStatement.setString( parameterIndex: 3, performanceType);
        preparedStatement.setString( parameterIndex: 4, performanceTitle);
        preparedStatement.setString( parameterIndex: 5, date);
        preparedStatement.setString( parameterIndex: 6, seatArea);
        preparedStatement.setString( parameterIndex: 7, seatNumber);
        preparedStatement.setString( parameterIndex: 8, deliveryMethod);
```

```java
private static String generateRandomDate(LocalDate startDate, LocalDate endDate) {
    long startEpochDay = startDate.toEpochDay();
    long endEpochDay = endDate.toEpochDay();
    long randomEpochDay = ThreadLocalRandom.current().nextLong(startEpochDay, endEpochDay);

    return LocalDate.ofEpochDay(randomEpochDay).toString();
}
```

레코드를 삽입하는 코드이다. 대용량 데이터로 만들고 내가 지정한 레코드 수는 100,000개이다. for문으로 삽입한다. 100,000개의 데이터를 추가하기 위해 모든 값에 대한 범위를 지정해주고 랜덤으로 값을 주도록 하였다. 날짜에 대해 랜덤 값을 부여하기 위해 generateRandomDate 메소드를 추가하였다.
아래의 화면은 생성된 테이블과 삽입된 레코드의 모습이다.



다음은 Area_bitmap.java이다. 좌석 구역 컬럼에 대한 bitmap Index를 생성하는 코드이다.

```java
public static int getRCount() {
    int rcount = 0;

    try (Connection connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
         PreparedStatement statement = connection.prepareStatement( sql: "SELECT COUNT(*) AS count FROM tickets")) {
        ResultSet resultSet = statement.executeQuery();

        if (resultSet.next()) {
            rcount = resultSet.getInt( columnLabel: "count");
            System.out.println("Record count: " + rcount);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return rcount;
}
```

레코드 수를 세는 메소드이다. 우리는 100,000개라고 이미 알고 있지만, 한 번 더 확인하는 것이다

```java
public static BitSet[] createBitmapIndexes(int rcount) {
    BitSet[] bm = new BitSet[9];

    try (Connection connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
         PreparedStatement statement = connection.prepareStatement( sql: "SELECT seat_area FROM tickets")) {
        ResultSet resultSet = statement.executeQuery();

        while (resultSet.next()) {
            String seatArea = resultSet.getString( columnLabel: "seat_area");

            int index = getSeatAreaIndex(seatArea);

            if (bm[index] == null) {
                bm[index] = new BitSet(rcount);
            }
            bm[index].set(resultSet.getRow() - 1);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return bm;
}
```

비트맵 인덱스를 생성하는 메소드이다. JDBC에 연결해서 데이터베이스에서 seat_area컬럼의 값을 받아와 생성하는 것이다.

```java
public static int getSeatAreaIndex(String seatArea) {
    switch (seatArea) {
        case "F1":
            return 0;
        case "F2":
            return 1;
        case "F3":
            return 2;
        case "2-A":
            return 3;
        case "2-B":
            return 4;
        case "2-C":
            return 5;
        case "3-D":
            return 6;
        case "3-E":
            return 7;
        case "3-F":
            return 8;
        default:
            throw new IllegalArgumentException("Invalid seat area: " + seatArea);
    }
}
```

seat-area컬럼의 distinct한 값을 switch-case문으로 나눈 것이다.

```java
public static void saveBitmapIndexes(BitSet[] bitmaps) {
    for (int i = 0; i < bitmaps.length; i++) {
        BitSet bm = bitmaps[i];
        String filename = (i + 1) + "_seat_area_bitmap.txt";

        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename))) {
            for (int j = 0; j < bm.size(); j++) {
                int bit = bm.get(j) ? 1 : 0;
                writer.write(String.valueOf(bit));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    System.out.println("Success to save files.");
}
```

계산한 비트맵 인덱스를 txt파일에 저장하는 메소드이다.

다음은 delivery_bitmap.java이다. 수령 방법 컬럼에 대한 bitmap Index를 생성하는 코드이다. 기본적인 틀은 Area_bitmap.java와 유사하므로 많이 다른 부분만 제시한다. 아래는 수령 방법을 switch-case문으로 나눈 코드이다. 나머지는 거의 같다.

```java
public static int getDeliveryIndex(String delivery) {
    switch (delivery) {
        case "현장수령":
            return 0;
        case "택배수령":
            return 1;
        default:
            throw new IllegalArgumentException("Invalid delivery: " + delivery);
    }
}
```

다음은 type_bitmap.java이다. 공연 종류 컬럼에 대한 bitmap Index를 생성하는 코드이다. 기본적인 틀은 Area_bitmap.java와 유사하므로 많이 다른 부분만 제시한다. 아래는 공연 종류를 switch-case문으로 나눈 코드이다. 나머지는 거의 같다.

```java
public static int getTypeIndex(String type) {
    switch (type) {
        case "콘서트":
            return 0;
        case "연극":
            return 1;
        case "뮤지컬":
            return 2;
        case "클래식":
            return 3;
        default:
            throw new IllegalArgumentException("Invalid type: " + type);
    }
}
```

다음은 QueryProgramming.java이다. 이 코드는 질의처리를 할 수 있는 코드이다. 내가 이해한 바로는 multi-key로 하여 count함수를 사용하면 되는 것으로 알고 있다. bitmap Index로 고른 컬럼들을 합쳐서 사용자가 입력하여 작동하는 API 및 UI에 대한 코드이다.

```
System.out.println("Which one do you want to see? 1. Performance type - Delivery method    2. Performance type - Seat Area");
int num = sc.nextInt();

String performanceType = "";
String deliveryMethod = "";
String seatArea = "";

if (num == 1) {
    System.out.print("Enter performance type (Concert, Play, Musical, Classic): ");
    performanceType = sc.next();
    System.out.print("Enter delivery method (Pick up, Post): ");
    deliveryMethod = sc.next();
} else if (num == 2) {
    System.out.print("Enter performance type (Concert, Play, Musical, Classic): ");
    performanceType = sc.next();
    System.out.print("Enter seat area (F1, F2, F3, 2-A, 2-B, 2-C, 3-D, 3-E, 3-F): ");
    seatArea = sc.next();
    int areaIndex = getSeatAreaIndex(seatArea);
    if (areaIndex == -1) {
        System.out.println("Invalid seat area.");
        return;
    }
    if (areaBitmaps[areaIndex] == null) {
        System.out.println("Invalid seat area.");
        return;
    }
} else {
    System.out.println("Wrong input.");
    return;
}
```

응용 UI코드이다. 콘솔 창에는 아래 화면과 같다.

```
C:\Users\USER\.jdks\openjdk-20\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.1\
Which one do you want to see? 1. Performance type - Delivery method    2. Performance type - Seat Area
2
```

```
if (typeBitmaps[typeIndex] != null) {
    BitSet result = (BitSet) typeBitmaps[typeIndex].clone();
    if (num == 1) {
        int deliveryIndex = getDeliveryIndex(deliveryMethod);
        if (deliveryIndex == -1) {
            System.out.println("Invalid delivery method.");
            return;
        }
        if (deliveryBitmaps[deliveryIndex] == null) {
            System.out.println("Invalid delivery method.");
            return;
        }
        result.and(deliveryBitmaps[deliveryIndex]);
    } else if (num == 2) {
        int areaIndex = getSeatAreaIndex(seatArea);
        result.and(areaBitmaps[areaIndex]);
    }
    int count = result.cardinality();
    System.out.println("Count: " + count);
} else {
    System.out.println("Invalid performance type, delivery method, or seat area.");
}
```

한 번 작동할 때 두 개의 bitmap Index만 사용한다. 그러면 하나의 bitmap Index는 사용하지 않으므로 이를 처리하기 위한 코드이다.

```
public static BitSet[] loadBitmapIndexes(String filenamePrefix, int bitmapCount) {
    BitSet[] bitmaps = new BitSet[bitmapCount];

    try {
        for (int i = 0; i < bitmapCount; i++) {
            String filename = filenamePrefix + (i + 1) + ".txt";
            BufferedReader reader = new BufferedReader(new FileReader(filename));

            String line = reader.readLine();
            BitSet bitmap = new BitSet(line.length());
            for (int j = 0; j < line.length(); j++) {
                int bit = Integer.parseInt(String.valueOf(line.charAt(j)));
                if (bit == 1) {
                    bitmap.set(j);
                }
            }
            bitmaps[i] = bitmap;
            reader.close();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    return bitmaps;
}
```

비트맵 인덱스를 txt파일에서 읽어와 load하는 메소드이다.

```java
public static int getTypeIndex(String type) {
    switch (type) {
        case "콘서트":
            return 0;
        case "연극":
            return 1;
        case "뮤지컬":
            return 2;
        case "클래식":
            return 3;
        default:
            throw new IllegalArgumentException("Invalid type: " + type);
    }
}

1 usage
public static int getDeliveryIndex(String delivery) {
    switch (delivery) {
        case "현장수령":
            return 0;
        case "택배수령":
            return 1;
        default:
            throw new IllegalArgumentException("Invalid delivery: " + delivery);
    }
}
```

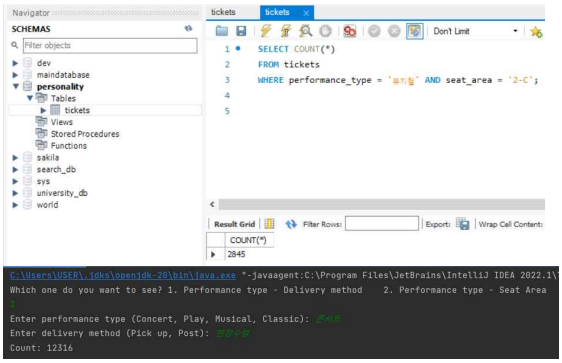
```java
public static int getSeatAreaIndex(String seatArea) {
    switch (seatArea) {
        case "F1":
            return 0;
        case "F2":
            return 1;
        case "F3":
            return 2;
        case "2-A":
            return 3;
        case "2-B":
            return 4;
        case "2-C":
            return 5;
        case "3-D":
            return 6;
        case "3-E":
            return 7;
        case "3-F":
            return 8;
        default:
            throw new IllegalArgumentException("Invalid area:" + seatArea);
    }
}
```

왼쪽과 오른쪽 화면은 콘솔에서 입력받은 것에 대한 결과를 도출하게 하기 위한 코드이다.

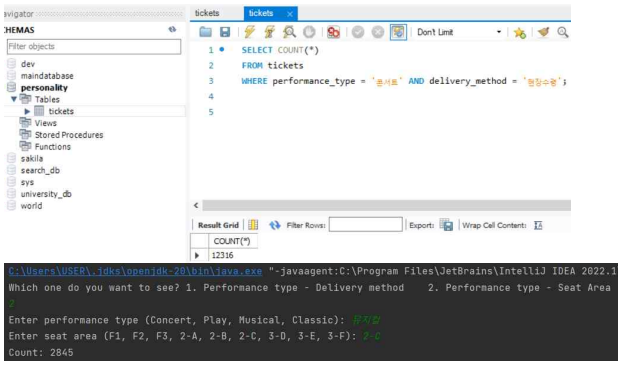
bitmap Index를 디스크 기반으로 저장하기 위한 파일은 .txt파일로 선택하였다. 엑셀 파일은 인덱스를 쉽게 확인할 수 있지만 큰 크기의 인덱스를 처리하는 경우에는 성능이 저하할 수 있다. 텍스트 파일은 간단하고 가볍지만 데이터를 단순히 저장하고 검색하는 용도로 사용해야 하기에 텍스트 파일로 선택하였다.

다음은 제약 조건에 대한 설명이다.
1. 현재 내가 쓰는 인텔리제이에서 콘솔창에 한글이 깨지는 현상이 발생하여, 입력을 안내하는 문구나 오류 문구들을 전부 영어로 작성하고, 기능을 위한 UI에서 직접 입력할 때는 한국어로 입력하여 읽을 수 있도록 하였다.
2. 이름의 수는 대략 50개로 한정, 공연 종류는 콘서트, 연극, 뮤지컬, 클래식으로 한정, 공연 제목은 40개로 한정, 좌석 구역은 플로어 구역의 F1, F2, F3, 2층 구역은 2-A, 2-B, 2-C, 3층 구역은 3-D, 3-E, 3-F로 한정, 수령방법은 현장수령, 택배수령으로 한정한다. 한정한 내용들은 for문을 통해 랜덤으로 부여한다. 주민번호 앞자리 6글자는 숫자로 이루어지고, 날짜는 DATE 형식으로 2023-01-01부터 2023-12-31사이이며 좌석 번호는 두자리 숫자로 이루어진다.

다음은 기능 동작에 대한 정확성을 검증한다. 응용 동작 시나리오는 다음과 같다.
가. 1. 공연 종류와 수령방법을 입력받아서 count 하기 2. 공연 종류와 좌석 구역을 입력받아서 count 하기 중 선택한다.
나. 한글로 각 항목에 대해 보고싶은 결과를 입력한다.
다. 결과를 확인한다.
이에 대해 검증하기 위해 Mysql Workbench로 SQL구문을 작성하여 확인하였다. 1번과 2번에 대한 예시 화면과 결과 출력 그리고 MySQL로 검증한 화면은 다음과 같다.
1번 예시)



2번 예시)



마지막으로 실행화일 생성 방법에 대한 설명이다. 나는 IDE를 IntelliJ를 사용하였고, 현재 나의 IntelliJ는 한글을 인식하지 못하여 부득이하게 영어로 작성하고, 값을 입력받을 때만 한글을 사용하였다.
실행화일은 jar파일이다. ForProgram.jar과 QueryProcessing.jar이다. 이들을 실행하는 방법은 두 jar 파일이 존재하는 파일의 경로에 명령 프롬프트를 실행하고 “java -jar FromProgram.jar FromProgram” 명령을 실행하거나 “java -jar QueryProcessing.jar QueryProcessing” 명령을 실행한다. 실행하는 컴퓨터에서 한글을 인식하고 허용해야 이를 실행할 수 있다.