# 데이터베이스 프로그래밍
# DB설계 과제 보고서

## 프로젝트명 : 온라인 음악 스트리밍 서비스

| | |
|---|---|
| 교과목명 | 데이터베이스 프로그래밍 02 |
| 담당교수 | 이용규 |
| 학 번 | 2019111985 |
| 이 름 | 남상원 |

# C O N T E N T S

1. 개요
 1.1 프로젝트 개요
 1.2 프로젝트 목표

2. 요구사항 파악
 2.1 기능 관련 요구사항
 2.2 데이터 요건 분석
 2.3 데이터 모델링

3. 데이터베이스 설계
 3.1 데이터베이스 논리적 설계
 3.2 데이터베이스 물리적 설계

4. 구현 결과
 4.1 구현된 데이터베이스

5. 평가 및 결론
 5.1 평가
 5.2 결론

6. 참고문헌

# 1. 개요

## 1.1 프로젝트 개요

온라인 음악 스트리밍 서비스는 한국저작권위원회에서 작성한 "디지털 음악 시장의 승자, 스트리밍 서비스"에 따르면 "2023년에서의 디지털 음악 시장의 총매출은 363억 달러(원화 약 49조원)에 이를 것으로 예상되며, 그 중 가장 큰 비중으로 차지하는 음악 스트리밍 시장 규모는 258억 4000만 달러로 추정된다."고 서술하고 있습니다.

사용자가 원하는 음악을 스마트폰과 같은 다양한 기기를 통해 언제 어디서나 쉽게 접근할 수 있는 편리성, 다양한 장르와 아티스트의 음악을 제공하여 자신의 취향에 맞는 음악을 쉽게 찾을 수 있는 접근성과 같은 장점들을 통해 앞으로도 온라인 음악 스트리밍 서비스는 현재 디지털 음악 소비의 중심 역할을 하고 있으며, 인기 또한 계속될 것으로 예상됩니다.

위에서 설명한 바와 같이 온라인 음악 스트리밍 서비스는 사용자에게 다양한 음악을 쉽게 탐색하고 스트리밍할 수 있는 기능을 제공합니다. 이러한 편리성과 접근성을 제공하는데 있어서 데이터베이스 설계는 해당 서비스의 성공에 필수적인 요소입니다. 따라서 효율적인 데이터베이스 설계를 통해 사용자에게 빠르고 정확한 검색 결과를 제공하고, 서비스 제공자 측면에서는 비용 절감과 운영 효율성 향상을 가능토록 하고자 해당 프로젝트를 진행하게 되었습니다.

## 1.2 프로젝트 목표

해당 프로젝트는 사용자들에게 온라인으로 다양한 음악을 쉽고 빠르게 스트리밍 서비스를 제공하는 것이 목표입니다. 이를 달성하기 위해서는 온라인 음악 스트리밍 서비스에 대한 효율적이고 확장 가능한 데이터베이스 설계를 개발하는 것입니다. 해당 설계는 다음과 같은 목표를 달성하기 위해 노력해야 합니다.

- 해당 프로젝트에서는 제공해야 하는 음악 트랙, 아티스트, 앨범, 사용자 정보, 플레이 리스트, 찜한 노래 등과 같은 데이터들을 효과적으로 저장 및 관리하기 위한 데이터베이스를 구체적으로 설계합니다.

- 해당 프로젝트에서는 다양한 음악과 아티스트를 사용자에게 제공하기 위해서 다양한 음악 장르와 아티스트를 포괄하는 음악 데이터베이스를 구축하도록 합니다.

- 해당 프로젝트에서는 대량의 데이터를 빠르고 효율적이게 접근하여 검색 및 스트리밍 등과 같은 기능을 제공하기 위해서 적절한 데이터 구조와 인덱스를 적용하여 성능을 최적화하도록 합니다.

이 프로젝트는 음악 스트리밍 서비스를 효과적으로 운영하기 위한 데이터베이스 설계에 중점을 두며, 데이터의 정확성, 성능, 다양성, 그리고 사용자 경험을 향상시키는 목표를 가지고 있습니다. 이러한 목표를 이루기 위한 데이터베이스의 효과적인 설계와 관리는 사용자에게 빠르고 정확한 검색 결과를 제공하고, 서비스 제공자 측면에서는 운영에 있어 효율성을 향상시키는 것과 같이 음악 스트리밍 서비스의 성공에 중요한 역할을 합니다.

## 2. 요구사항 파악

### 2.1 기능 관련 요구사항

온라인 음악 스트리밍 서비스는 아래와 같은 기능을 24시간 365일 사용자의 요청에 따라 즉시 제공해야 합니다.

- 사용자 계정 관리
- 사용자는 회원 가입 시 이름, 이메일 주소, 아이디, 비밀번호 등의 정보를 입력해야 합니다.
- 사용자는 로그인 시 아이디와 비밀번호를 입력해야 합니다.
- 사용자는 탈퇴 시 회원 정보를 삭제할 수 있어야 합니다.

- 음악 콘텐츠 관리
- 음악 트랙에는 제목, 아티스트, 장르, 작곡가, 작사가, 앨범 등의 정보가 포함되어야 합니다.
- 음악 트랙에는 여러 작곡가, 작사가와 장르가 포함될 수 있습니다.
- 아티스트에는 이름, 데뷔일 등의 정보가 포함되어야 합니다.
- 아티스트는 그룹일 수 있기 때문에 다수의 멤버를 포함되어야 합니다.
- 앨범에는 제목, 발매일 등의 정보가 포함되어야 합니다.

- 검색 및 추천
- 사용자는 음악 제목, 아티스트, 장르 등으로 음악을 검색할 수 있어야 합니다.
- 시스템은 사용자의 선호도에 따라 음악을 추천할 수 있어야 합니다.
- 사용자의 선호도는 즐겨 듣는 음악의 아티스트의 다른 음악 혹은 장르가 유사한 음악 등을 통해 결정할 수 있어야합니다.

- 플레이리스트
- 사용자는 원하는 음악을 모아서 재생 목록을 생성할 수 있어야 합니다.
- 재생 목록은 이름, 설명, 음악 트랙의 목록, 생성한 사용자 등으로 구성되어야 합니다.
- 사용자는 재생 목록을 공유 여부를 선택할 수 있어야 합니다.

- 사용자가 찜한 가수
- 사용자는 자신의 마음에 드는 가수를 찜할 수 있어야 합니다.
- 사용자들은 다양한 가수를 찜할 수 있어야 합니다.

- 사용자가 찜한 노래
- 사용자는 자신의 마음에 드는 노래를 찜할 수 있어야 합니다.
- 사용자들은 다양한 노래를 찜할 수 있어야 합니다.

온라인 음악 스트리밍 서비스가 운용될 때 수익을 확보하기 위해 음악 감상 5번 마다 광고를 시청하여 음악을 스트리밍할 수 있는 무료 서비스와 광고 없이 음악을 스트리밍할 수 있는 유료 서비스와 같은 모드를 추가할 수 있습니다. 또한 신규 음악 콘텐츠의 추가에 따라 데이터베이스가 확장되거나 새로운 기능의 추가에 따라 시스템이 변경될 수 있어야 합니다.

## 2.2 데이터 요건 분석

(1) 관련 문서 수집

| No | 문서명 | 유형 | 설명 |
|---|---|---|---|
| 1 | 사용자 명부 | 문서 | 서비스를 사용하는 사용자들을 기록한 문서이다. |
| 2 | 앨범 목록 | 문서 | 발매된 앨범들을 기록한 문서이다. |
| 3 | 음악 목록 | 문서 | 발매된 음악들을 기록한 문서이다. |
| 4 | 플레이리스트 목록 | 문서 | 사용자가 생성한 플레이리스트를 기록한 문서이다. |
| 5 | 음악 장르 목록 | 문서 | 해당 음악의 장르를 분류하기 위한 문서이다. |
| 6 | 작곡가 목록 | 문서 | 해당 음악을 작곡한 작곡가를 기록한 문서이다. |
| 7 | 작사가 목록 | 문서 | 해당 음악을 작사한 작사가를 기록한 문서이다. |
| 8 | 가수(그룹) 목록 | 문서 | 해당 음악을 부른 가수(그룹)를 기록한 문서이다. |
| 9 | 멤버 목록 | 문서 | 해당 음악을 부른 가수의 멤버를 기록한 문서이다. |

[ 표 1 ]

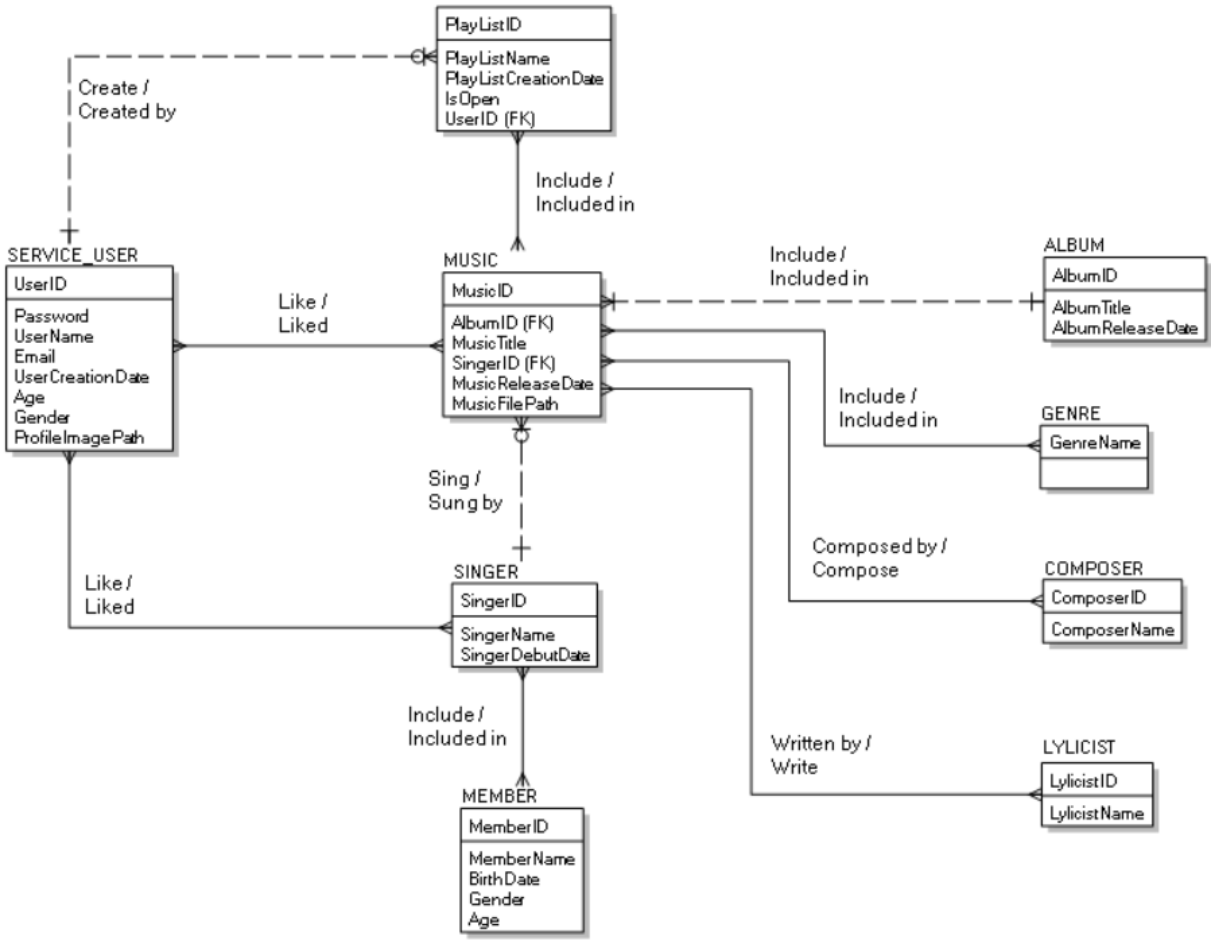
(2) 정보 항목 목록 작성

| No | 정보 항목군 | 정보 항목 |
|---|---|---|
| 1 | 사용자 정보 | 아이디, 비밀번호, 이름, 이메일, 생성일자, 나이, 성별, 프로필 이미지 경로 |
| 2 | 플레이리스트 정보 | 플레이리스트 명, 생성일자, 공개 여부, 생성자 정보 |
| 3 | 앨범 정보 | 앨범 정보, 발매일 |
| 4 | 음악 정보 | 음악 제목, 가수 정보, 앨범 정보, 발매일, 음악 파일 경로 |
| 5 | 음악 장르 정보 | 음악 장르명 |
| 6 | 작곡가 정보 | 작곡가 이름 |
| 7 | 작사가 정보 | 작사가 이름 |
| 8 | 가수 정보 | 가수명(그룹명), 데뷔일자 |
| 9 | 가수 멤버 정보 | 멤버명, 생년월일, 나이, 성별 |

[ 표 2 ]

## 2.3 데이터 모델링

<개념적 모델>




[ 그림 1 ]

<개체 테이블>

| 개체 | 식별자 | 속성 |
|---|---|---|
| SERVICE_USER | UserID | Password, UserName, Email, UserCreationDate, Age, Gender, PlofileImagePath |
| PLAYLIST | PlayListID | PlayListname, PlayListCreationDate, IsOpen, UserID |
| ALBUM | AlbumID | AlbumTitle, AlbumReleaseDate |
| MUSIC | MusicID | AlbumID, MusicTitle, SingerID, MusicReleaseDate, MusicFilePath |
| GENRE | GenreName | |
| COMPOSER | ComposerID | ComposerName |
| LYLICIST | LylicistID | LylicistName |
| SINGER | SingerID | SingerName, SingerDebutDate |
| MEMBER | MemberID | MemberName, BirthDate, Gender, Age |

[ 표 3 ]

<관계 테이블>

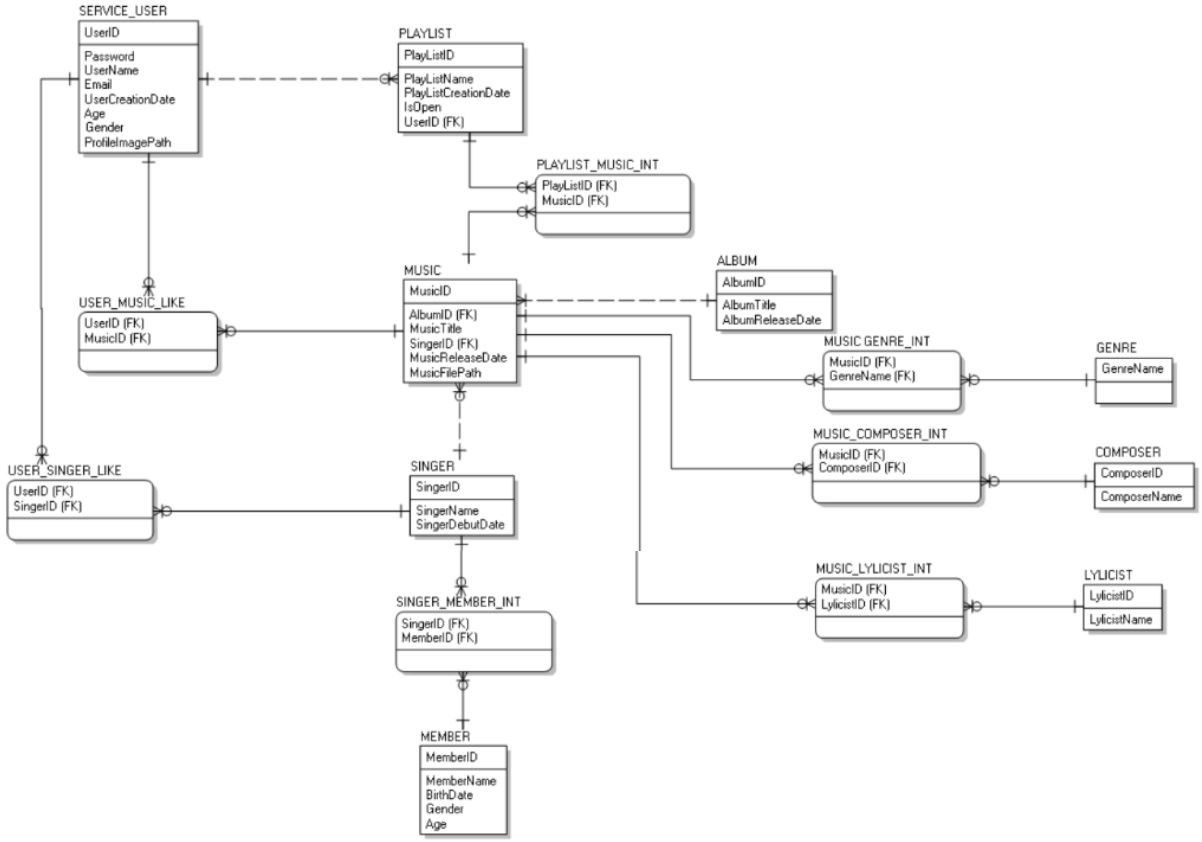
| 개체1 | 개체2 | 식별/비식별 | 최대 카디널리티 | 최소 카디널리티 |
|---|---|---|---|---|
| SERVICE_USER | PLAYLIST | 비식별 | 1:N | M-O |
| SERVICE_USER | MUSIC | 식별 | N:M | M-O |
| SERVICE_USER | SINGER | 식별 | N:M | M-O |
| ALBUM | MUSIC | 비식별 | 1:N | M-M |
| MUSIC | GENRE | 식별 | N:M | M-M |
| MUSIC | COMPOSER | 식별 | N:M | M-M |
| MUSIC | LYLICIST | 식별 | N:M | M-M |
| MUSIC | SINGER | 식별 | N:M | M-M |
| SINGER | MEMBER | 식별 | N:M | M-M |

[ 표 4 ]

## 3. 데이터베이스 설계

### 3.1 데이터베이스 논리적 설계

<논리적 모델>




[ 그림 2 ]

<SERVICE_USER 테이블>

| 칼럼명 | 타입 | 키형태 | NULL | 비고 |
|---|---|---|---|---|
| UserID | VARCHAR2(20) | | NOT NULL | |
| Password | VARCHAR2(20) | | NOT NULL | |
| UserName | VARCHAR2(20) | | NOT NULL | |
| Email | VARCHAR2(40) | | NULL | |
| UserCreation Date | Date | | NULL | |
| Age | INTEGER | | NULL | Age > 0 AND Age < 200 |
| Gender | BOOLEAN | | NULL | Gender In (1, 0) |
| ProfileImage Path | VARACHAR(300) | | | |

[ 표 5 ]

<PLAYLSIT 테이블>

| 칼럼명 | 타입 | 키형태 | NULL | 비고 |
|---|---|---|---|---|
| PlayListID | VARCHAR2(20) | PK | NOT NULL | |
| PlayListName | VARCHAR2(30) | | | DEFAULT PlayListName = 'Play List' |
| PlayList CreationDate | DATE | | | |
| IsOpen | BOOLEAN | | | DEFAULT IsOpen = 1 |
| UserID | VARCHAR2(20) | FK <SERVICE_USER (UserID)> | NOT NULL | |

[ 표 6 ]

<ALBUM 테이블>

| 칼럼명 | 타입 | 키형태 | NULL | 비고 |
|---|---|---|---|---|
| AlbumID | VARCHAR2(30) | PK | NOT NULL | |
| AlbumTitle | VARCHAR2(30) | | NOT NULL | DEFAULT AlbumTitle = 'Unidentified' |
| Album ReleaseDate | DATE | | NULL | |

[ 표 7 ]

<MUSIC 테이블>

| 칼럼명 | 타입 | 키형태 | NULL | 비고 |
|---|---|---|---|---|
| MusicID | VARCHAR2(20) | PK | NOT NULL | |
| MusicTitle | VARCHAR2(30) | | NOT NULL | DEFAULT MusicTitle = 'Unidentified' |
| Music ReleaseDate | DATE | | NULL | |
| MusicFilePath | VARCHAR2(300) | | NOT NULL | |
| SingerID | VARCHAR2(20) | FK <SINGER (SingerID)> | NOT NULL | |
| AlbumID | VARCHAR2(30) | FK <ALBUM (AlbumID)> | NOT NULL | |

[ 표 8 ]

<SINGER 테이블>

| 칼럼명 | 타입 | 키형태 | NULL | 비고 |
|---|---|---|---|---|
| SingerID | VARCHAR2(20) | PK | NOT NULL | |
| SingerName | VARCHAR2(30) | | NOT NULL | DEFAULT SingerName = 'Unidentified' |
| SingerDebutDate | DATE | | NULL | |

[ 표 9 ]

<MEMBER 테이블>

| 칼럼명 | 타입 | 키형태 | NULL | 비고 |
|---|---|---|---|---|
| MemberID | VARCHAR2(20) | PK | NOT NULL | |
| MemberName | VARCHAR2(20) | | NOT NULL | DEFAULT MemberName = 'Unidentified' |
| BirthDate | DATE | | NULL | |
| Gender | BOOLEAN | | NULL | Gender In (1, 0) |
| Age | INTEGER | | NULL | Age > 0 AND Age < 200 |

[ 표 10 ]

<GENRE 테이블>

| 칼럼명 | 타입 | 키형태 | NULL | 비고 |
|---|---|---|---|---|
| MemberID | VARCHAR2(30) | PK | NOT NULL | |

[ 표 11 ]

<COMPOSER 테이블>

| 칼럼명 | 타입 | 키형태 | NULL | 비고 |
|---|---|---|---|---|
| ComposerID | VARCHAR2(20) | PK | NOT NULL | |
| ComposerName | VARCHAR2(30) | | | DEFAULT ComposerName= 'Unidentified' |

[ 표 12 ]

<LYLICIST 테이블>

| 칼럼명 | 타입 | 키형태 | NULL | 비고 |
|---|---|---|---|---|
| LylicistID | VARCHAR2(20) | PK | NOT NULL | |
| LylicistName | VARCHAR2(30) | | | DEFAULT LylicistName= 'Unidentified' |

[ 표 13 ]

<USER_MUSIC_LIKE 테이블>

| 칼럼명 | 타입 | 키형태 | NULL | 비고 |
|---|---|---|---|---|
| UserID | VARCHAR2(20) | PK & FK <SERVICE_USER (UserID)> | NOT NULL | |
| MusicID | VARCHAR2(20) | PK & FK <MUSIC (MusicID)> | NOT NULL | |

[ 표 14 ]

<USER_SINGER_LIKE 테이블>

| 칼럼명 | 타입 | 키형태 | NULL | 비고 |
|---|---|---|---|---|
| UserID | VARCHAR2(20) | PK & FK <SERVICE_USER (UserID)> | NOT NULL | |
| SingerID | VARCHAR2(20) | PK & FK <SINGER (SingerID)> | NOT NULL | |

[ 표 15 ]

<PLAYLIST_MUSIC_INT 테이블>

| 칼럼명 | 타입 | 키형태 | NULL | 비고 |
|---|---|---|---|---|
| PlayListID | VARCHAR2(20) | PK & FK <PLAYLIST (PlayListID)> | NOT NULL | |
| MusicID | VARCHAR2(20) | PK & FK <MUSIC (MusicID)> | NOT NULL | |

[ 표 16 ]

<SINGER_MEMBER_INT 테이블>

| 칼럼명 | 타입 | 키형태 | NULL | 비고 |
|---|---|---|---|---|
| SingerID | VARCHAR2(20) | PK & FK <SINGER (SingerID)> | NOT NULL | |
| MemberID | VARCHAR2(20) | PK & FK <MEMBER (MemberID)> | NOT NULL | |

[ 표 17 ]

<MUSIC_GENRE_INT 테이블>

| 칼럼명 | 타입 | 키형태 | NULL | 비고 |
|---|---|---|---|---|
| MusicID | VARCHAR2(20) | PK & FK <MUSIC (MusicID)> | NOT NULL | |
| GenreName | VARCHAR2(30) | PK & FK <GENRE (GenreName)> | NOT NULL | |

[ 표 18 ]

<MUSIC_COMPOSER_INT 테이블>

| 칼럼명 | 타입 | 키형태 | NULL | 비고 |
|---|---|---|---|---|
| MusicD | VARCHAR2(20) | PK & FK <MUSIC (MusicID)> | NOT NULL | |
| ComposerID | VARCHAR2(20) | PK & FK <COMPOSER (ComposerID)> | NOT NULL | |

[ 표 19 ]

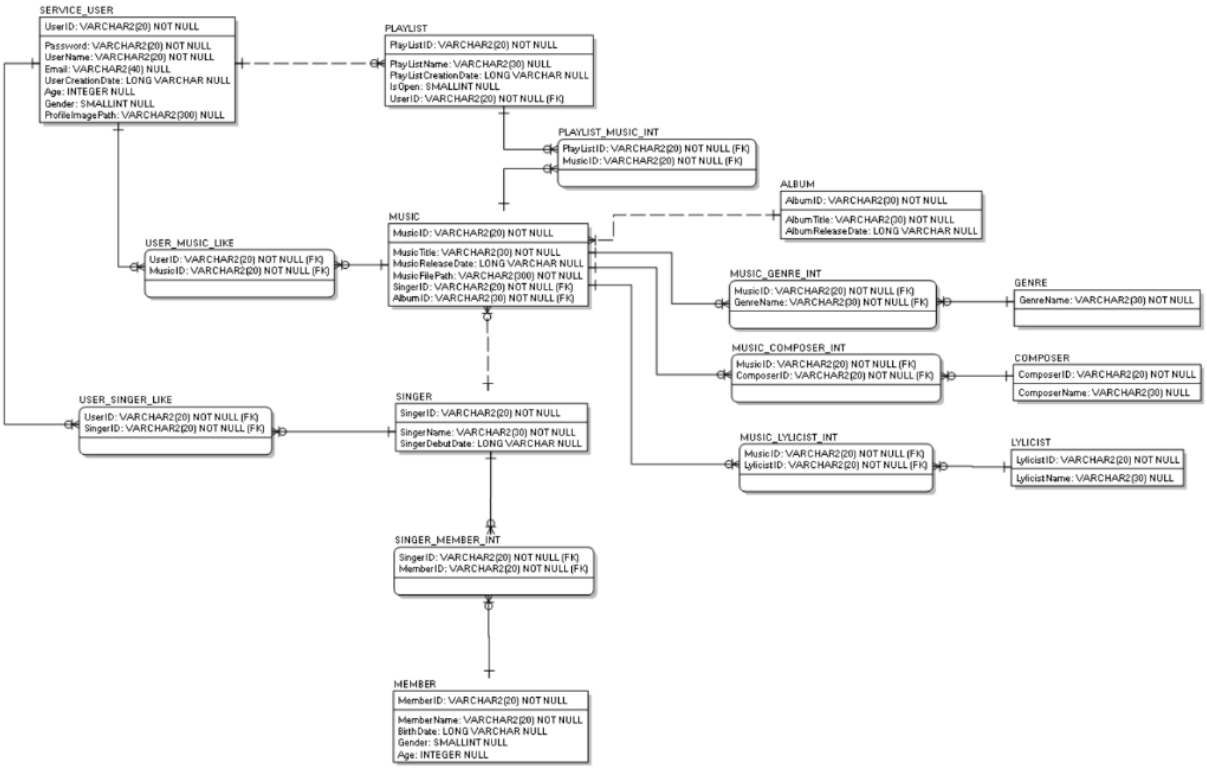
<MUSIC_LYLICIST_INT 테이블>

| 칼럼명 | 타입 | 키형태 | NULL | 비고 |
|---|---|---|---|---|
| MusicD | VARCHAR2(20) | PK & FK <MUSIC (MusicID)> | NOT NULL | |
| LylicistID | VARCHAR2(20) | PK & FK <LYLICIST (LylicistID)> | NOT NULL | |

[ 표 20 ]

## 3.2 데이터베이스 물리적 설계

<물리적 모델>




[ 그림 3 ]

## 4. 구현 결과

### 4.1 구현된 데이터베이스

(1) 테이블 스키마 구조, 데이터 확인

[그림 1]에서의 테이블들은 서비스를 이용하기 위한 사용자, 서비스에서 제공하는 음악과 앨범, 사용자가 원하는 음악만을 모아 은놓 플레이리스트, 해당 음악에 속하는 장르, 해당 음악을 제작한 작곡가와 작사가, 해당 음악을 부른 가수와 그의 멤버들로 구성된 테이블입니다. 해당 테이블들은 해당 프로젝트에서 제시하는 서비스를 제공하는데 있어서 필수적인 데이터들을 저장하기 위한 테이블입니다.

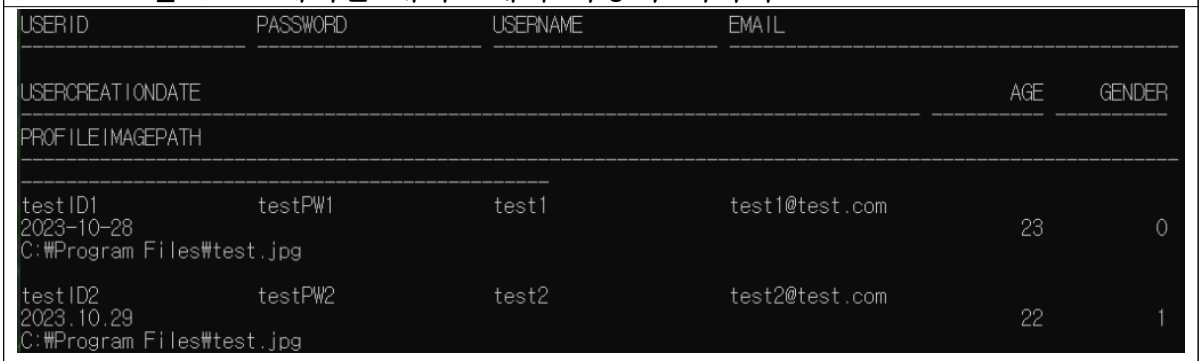
다음 테이블을 아래의 설명과 같이 관계 지어서 해당 프로젝트에서의 서비스를 빠르고 효율적이게 제공합니다.

- 사용자 정보 테이블과 플레이리스트 테이블 간의 1:N 관계 구성: 한 명의 사용자가 여러 개의 플레이리스트를 생성하여 관리할 수 있습니다.
- 사용자 정보 테이블과 음악 테이블 간의 N:M 관계 구성: 사용자들은 자신의 취향에 따라 좋아하는 음악을 찜할 수 있습니다. 이를 통해 음악의 인기도를 측정하거나 해당 음악과 비슷한 장르나 동일한 가수, 작곡가, 작사가를 지닌 다른 음악을 추천하는 용도로 사용할 수 있습니다.
- 사용자 정보 테이블과 가수 테이블 간의 N:M 관계 구성: 사용자들은 자신의 취향에 따라 좋아하는 가수를 찜할 수 있습니다. 이를 통해 가수의 인기도를 측정하거나 해당 가수가 부른 다른 음악들을 추천하는 용도로 사용할 수 있습니다.
- 앨범 테이블과 음악 테이블 간의 1:N 관계 구성: 한 앨범은 1개 이상의 노래로 구성합니다. 단일 음악만 등록하더라도 해당 데이터베이스에서는 앨범으로 구성하도록 합니다.
- 음악 테이블과 음악 장르 테이블 간의 N:M 관계 구성: 음악들은 다양한 장르를 지닐 수 있습니다. 이를 통해 음악 간의 유사도를 판단할 수 있습니다.
- 음악 테이블과 작곡가 테이블 간의 N:M 관계 구성: 음악들은 다양한 작곡가 정보를 지닙니다. 공동 작곡으로 인해 음악이 만들어 질 수 있기 때문에 하나의 음악에는 1명 이상의 작곡가가 포함될 수 있습니다.
- 음악 테이블과 작사가 테이블 간의 N:M 관계 구성: 음악들은 다양한 작사가 정보를 지닙니다. 공동 작사로 인해 음악이 만들어 질 수 있기 때문에 하나의 음악에는 1명 이상의 작사가가 포함될 수 있습니다.
- 가수 테이블과 음악 테이블 간의 1:N 관계 구성: 한 명의 가수(그룹)가 다양한 노래를 부를 수 있습니다. 하나의 음악에는 한 명의 가수(그룹)만이 포함됩니다. 이를 통해 해당 가수가 부른 다양한 음악을 사용자에게 추천할 수 있습니다.
- 가수 테이블과 멤버 테이블 간의 N:M 관계 구성: 가수(그룹)들은 한 명 이상의 멤버들로 구성됩니다. 솔로 가수일지라도 한 명의 멤버를 지닌 가수(그룹)으로 구성되도록 합니다. 다음과 같이 구성한 이유는 아이돌 그룹에서 유닛 그룹을 구성하여 노래를 발매하는 것과 같이 한 명의 가수가 여러 그룹에 속할 수 있기 때문입니다.
- 플레이리스트 테이블과 음악 테이블 간의 N:M 관계 구성 : 플레이리스트들은 다양한 음악들을 포함시킬 수 있습니다. 음악들은 다양한 플레이리스트에 중복적으로 포함될 수 있기 때문에 다음과 같이 구성하였습니다.

다음은 [그림 3]을 만족하도록 구성된 데이터베이스에 데이터를 추가한 결과입니다.

**데이터베이스 내의 데이터 [표 21]**

SERVICE_USER 테이블 내의 2개의 사용자 데이터



| USERID | PASSWORD | USERNAME | EMAIL | AGE | GENDER |
|---|---|---|---|---|---|
| testID1 | testPW1 | test1 | test1@test.com | 23 | 0 |
| 2023-10-28 | | | | | |
| C:₩Program Files₩test.jpg | | | | | |
| testID2 | testPW2 | test2 | test2@test.com | 22 | 1 |
| 2023.10.29 | | | | | |
| C:₩Program Files₩test.jpg | | | | | |

[ 그림 6 ]

ALBUM 테이블 내의 2개의 앨범 데이터

| ALBUMID | ALBUMTITLE | ALBUMRELEASEDATE |
|---|---|---|
| ALBUM1 | 가을 우체국 앞에서 | 1994.12.01 |
| ALBUM2 | Goodbye, grief. | 2013.10.14 |

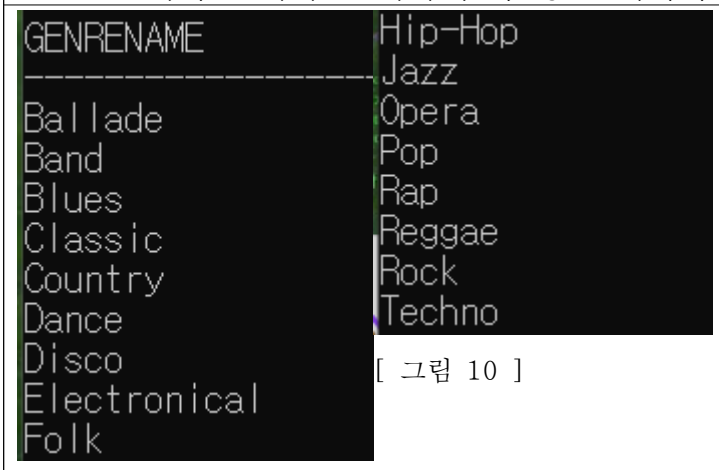
[ 그림 7 ]

MUSIC 테이블 내의 4개의 음악 데이터

| MUSICID | MUSICTITLE | MUSICRELEASEDATE |
|---|---|---|
| MUSICFILEPATH | | |
| SINGERID | ALBUMID | |
| MUSIC1 | 너를 보내고 | 1994.12.01 |
| C:₩Program Files₩0_0.mp4 | | |
| SINGER1 | ALBUM1 | |
| MUSIC2 | 사랑 two | 1994.12.01 |
| C:₩Program Files₩0_1.mp4 | | |
| SINGER1 | ALBUM1 | |
| MUSIC3 | 가을 우체국 앞에서 | 1994.12.01 |
| C:₩Program Files₩0_2.mp4 | | |
| SINGER1 | ALBUM1 | |
| MUSIC4 | 스물다섯, 스물하나 | 2013.10.14 |
| C:₩Program Files₩1_3.mp4 | | |
| SINGER2 | ALBUM2 | |

[ 그림 8 ]

GENRE 테이블 내의 17개의 음악 장르 데이터

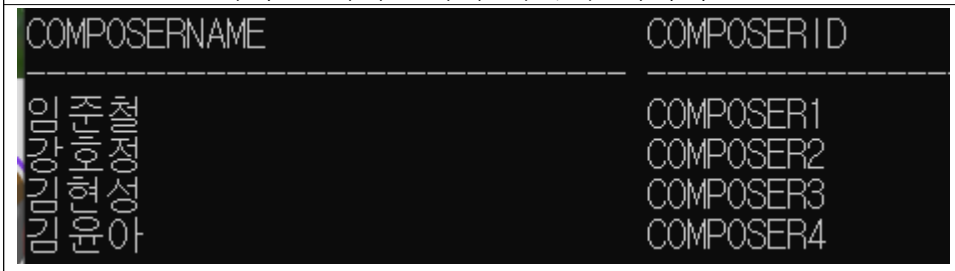


| GENRENAME |
| --- |
| Ballade |
| Band |
| Blues |
| Classic |
| Country |
| Dance |
| Disco |
| Electronical |
| Folk |
| Hip-Hop |
| Jazz |
| Opera |
| Pop |
| Rap |
| Reggae |
| Rock |
| Techno |

[ 그림 9 ]

[ 그림 10 ]

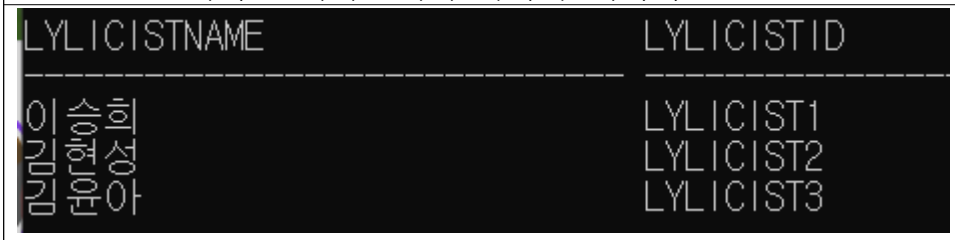
COMPOESR 테이블 내의 4개의 작곡가 데이터



| COMPOSERNAME | COMPOSERID |
| --- | --- |
| 임준철 | COMPOSER1 |
| 강호정 | COMPOSER2 |
| 김현성 | COMPOSER3 |
| 김윤아 | COMPOSER4 |

[ 그림 11 ]

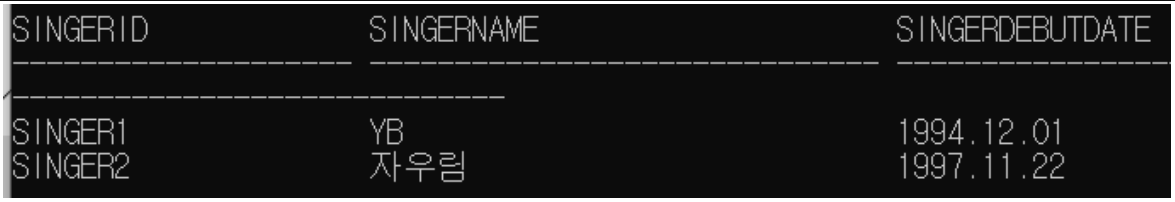
LYLICIST 테이블 내의 3개의 작사가 데이터



| LYLICISTNAME | LYLICISTID |
| --- | --- |
| 이승희 | LYLICIST1 |
| 김현성 | LYLICIST2 |
| 김윤아 | LYLICIST3 |

[ 그림 12 ]

SINGER 테이블 내의 2개의 가수(그룹) 데이터



| SINGERID | SINGERNAME | SINGERDEBUTDATE |
| --- | --- | --- |
| SINGER1 | YB | 1994.12.01 |
| SINGER2 | 자우림 | 1997.11.22 |

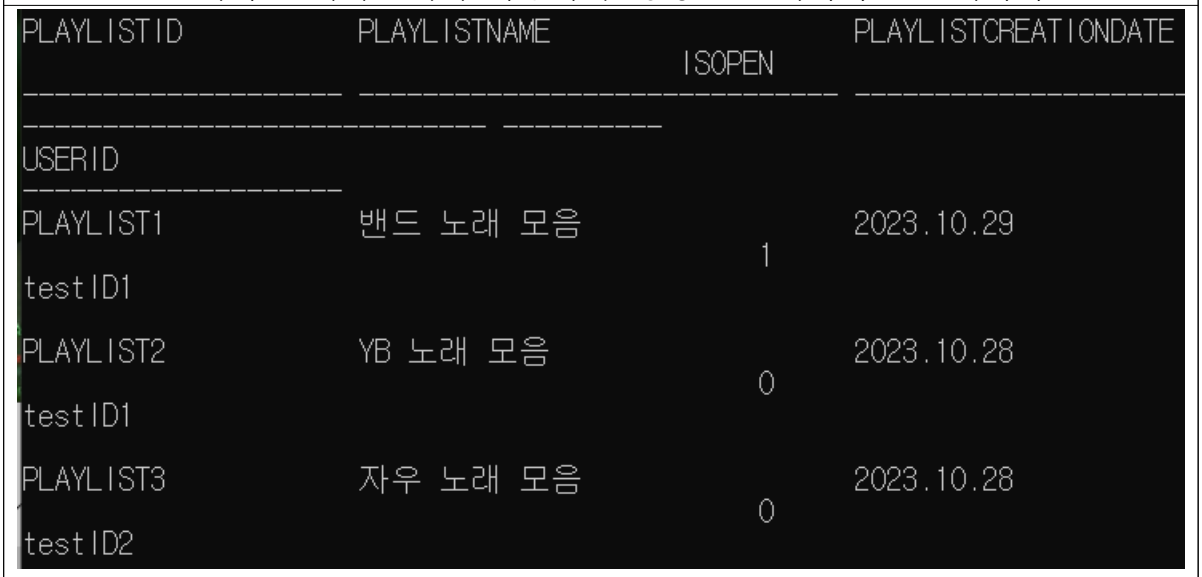
[ 그림 13 ]

MEMBER 테이블 내의 8개의 가수(그룹) 멤버 데이터

| MEMBERID | MEMBERNAME | GENDER | BIRTHDATE | AGE |
|---|---|---|---|---|
| MEMBER1 | 윤도현 | 0 | 1972.02.03 | 51 |
| MEMBER2 | 박태희 | 0 | 1969.09.19 | 53 |
| MEMBER3 | 김진원 | 0 | 1970.02.23 | 53 |
| MEMBER4 | 허준 | 0 | 1974.09.17 | 49 |
| MEMBER5 | Scott | 0 | 1977.02.19 | 46 |
| MEMBER6 | 이선규 | 0 | 1971.08.28 | 52 |
| MEMBER7 | 김진만 | 0 | 1972.02.25 | 51 |
| MEMBER8 | 김윤아 | 1 | 1974.03.11 | 49 |

[ 그림 14 ]

PLAYLIST 테이블 내의 3개의 사용자가 생성한 플레이리스트 데이터



| PLAYLISTID | PLAYLISTNAME | ISOPEN | PLAYLISTCREATIONDATE | USERID |
|---|---|---|---|---|
| PLAYLIST1 | 밴드 노래 모음 | 1 | 2023.10.29 | testID1 |
| PLAYLIST2 | YB 노래 모음 | 0 | 2023.10.28 | testID1 |
| PLAYLIST3 | 자우 노래 모음 | 0 | 2023.10.28 | testID2 |

[ 그림 15 ]

USER_MUSIC_LIKE 테이블 내의 5개의 사용자가 찜한 음악 데이터

| USERID | MUSICID |
|---|---|
| testID1 | MUSIC1 |
| testID1 | MUSIC2 |
| testID1 | MUSIC4 |
| testID2 | MUSIC3 |
| testID2 | MUSIC4 |

[ 그림 16 ]

USER_SINGER_LIKE 테이블 내의 3개의 사용자가 찜한 가수 데이터

| USERID | SINGERID |
|---|---|
| testID1 | SINGER1 |
| testID1 | SINGER2 |
| testID2 | SINGER2 |

[ 그림 17 ]

PLAYLIST_MUSIC_INT 테이블 내의 8개의
플레이리스트에 저장된 음악 데이터

| PLAYLISTID | MUSICID |
|---|---|
| PLAYLIST1 | MUSIC1 |
| PLAYLIST1 | MUSIC2 |
| PLAYLIST1 | MUSIC3 |
| PLAYLIST1 | MUSIC4 |
| PLAYLIST2 | MUSIC1 |
| PLAYLIST2 | MUSIC2 |
| PLAYLIST2 | MUSIC3 |
| PLAYLIST3 | MUSIC4 |

[ 그림 18 ]

SINGER_MEMBER_INT 테이블 내의 8개의
가수(그룹)에 속한 멤버 데이터

| SINGERID | MEMBERID |
|---|---|
| SINGER1 | MEMBER1 |
| SINGER1 | MEMBER2 |
| SINGER1 | MEMBER3 |
| SINGER1 | MEMBER4 |
| SINGER1 | MEMBER5 |
| SINGER2 | MEMBER6 |
| SINGER2 | MEMBER7 |
| SINGER2 | MEMBER8 |

[ 그림 19 ]

MUSIC_COMPOSER_INT 테이블 내의 5개의
음악을 작곡한 작곡가 데이터

| MUSICID | COMPOSERID |
|---|---|
| MUSIC1 | COMPOSER1 |
| MUSIC2 | COMPOSER1 |
| MUSIC2 | COMPOSER2 |
| MUSIC3 | COMPOSER3 |
| MUSIC4 | COMPOSER4 |

[ 그림 20 ]

MUSIC_LYLICIST_INT 테이블 내의 4개의
음악을 작사한 작곡가 데이터

| MUSICID | LYLICISTID |
|---|---|
| MUSIC1 | LYLICIST1 |
| MUSIC2 | LYLICIST1 |
| MUSIC3 | LYLICIST2 |
| MUSIC4 | LYLICIST3 |

[ 그림 21 ]

MUSIC_GENRE_INT 테이블 내의 11개의 해당 음악 장르 데이터

| MUSICID | GENRENAME |
|---|---|
| MUSIC1 | Band |
| MUSIC1 | Blues |
| MUSIC1 | Rock |
| MUSIC2 | Band |
| MUSIC2 | Rock |
| MUSIC3 | Band |

[ 그림 22 ]

| | |
|---|---|
| MUSIC3 | Folk |
| MUSIC3 | Rock |
| MUSIC4 | Ballade |
| MUSIC4 | Band |
| MUSIC4 | Rock |

[ 그림 23 ]

위의 [ 표 21 ]은 데이터베이스에 아래와 같은 정보의 데이터가 저장된 결과이다.

| | |
|---|---|
| 사용자 | 사용자 1 – testID1, testPW1, test1, test1@test.com, 2023.10.28, 23, 남성(0), (C:\Program Files\test.jpg) |
| | 사용자 2 – testID2, testPW2, test2, test2@test.com, 2023.10.29, 22, 여성(1), (C:\Program Files\test.jpg) |
| 앨범 및 음악 정보 | 앨범 1– (Album ID = ALBUM1) 가을 우체국 앞에서 (1994.12.01)<br>음악 1–1 – (album id = ALBUM1) 너를 보내고 (1994.12.01) (music id = MUSIC1)<br>(singer id = SINGER1), (C:\Program Files\0_0.mp4)<br>장르 – band, rock, blues<br>작곡가 – 임준철(COMPOESR1)<br>작사가 – 이승희 (LYLICIST1)<br>음악 1–2 – (album id = ALBUM1) 사랑 two (1994.12.01) (music id = MUSIC2)<br>(singer id = SINGER1), (C:\Program Files\0_1.mp4)<br>장르 – band, rock<br>작곡가 – 임준철(COMPOESR1), 강호정(COMPOESR2)<br>작사가 – 이승희(LYLICIST1)<br>음악 1–3 – (album id = ALBUM1) 가을 우체국 앞에서 (1994.12.01) (music id = MUSIC3)<br>(singer id = SINGER1), (C:\Program Files\0_2.mp4)<br>장르 – band, rock, folk<br>작곡가 – 김현성(COMPOESR3)<br>작사가 – 김현성(LYLICIST2)<br>가수(그룹) – (singer id = SINGER1) YB (1994.12.01)<br>멤버 – (member id = MEMBER1) 윤도현, 1972.02.03, 남성, 51<br>– (member id = MEMBER2) 박태희, 1969.09.19, 남성, 53<br>– (member id = MEMBER3) 김진원, 1970.02.23, 남성, 53<br>– (member id = MEMBER4) 허준, 1974.09.17, 남성, 49<br>– (member id = MEMBER5) Scott, 1977.02.19, 남성, 46 |
| | 앨범 2– (Album ID = ALBUM2) Goodbye, grief. (2013.10.14)<br>음악 2–1 – (album id = ALBUM2) 스물다섯, 스물하나 (2013.10.14) (music id = MUSIC4)<br>(singer id = SINGER2), (C:\Program Files\1_3.mp4)<br>장르 – band, rock, ballade<br>작곡가 – 김윤아(COMPOSER4)<br>작사가 – 김윤아(LYLICIST3)<br>가수(그룹) – (singer id = SINGER2) 자우림 (1997.11.22)<br>멤버 – (member id = MEMBER6) 이선규, 1971.08.28, 남성, 52<br>– (member id = MEMBER7) 김진만, 1972.02.25, 남성, 51<br>– (member id = MEMBER8) 김윤아, 1974.03.11, 여성, 49 |

| | |
|---|---|
| 플레이 리스트 | 사용자 1 플레이 리스트(PLID = PLAYLIST1) – 밴드 노래 모음, 2023.10.29, Open(1), 사용자 1 |
| | 사용자 1 플레이 리스트(PLID = PLAYLIST2) – YB 노래 모음, 2023.10.28, Not Open(0), 사용자 1 |
| | 사용자 2 플레이 리스트(PLID = PLAYLIST3) – 자우림 노래 모음, 2023.10.28, NOT Open(0), 사용자 2 |
| 사용자 찜한 노래 | 사용자 1 좋아요한 노래 – 너를 보내고, 사랑 two, 스물다섯, 스물하나 |
| | 사용자 2 좋아요한 노래 – 가을 우체국 앞에서, 스물다섯, 스물하나 |
| 사용자 찜한 가수 | 사용자 1 좋아요한 가수 – YB, 자우림 |
| | 사용자 2 좋아요한 가수 – 자우림 |

[ 표 22 ]

(2) PL/SQL 생성 및 테스트 결과

**[ 회원 탈퇴 프로시저 ]**

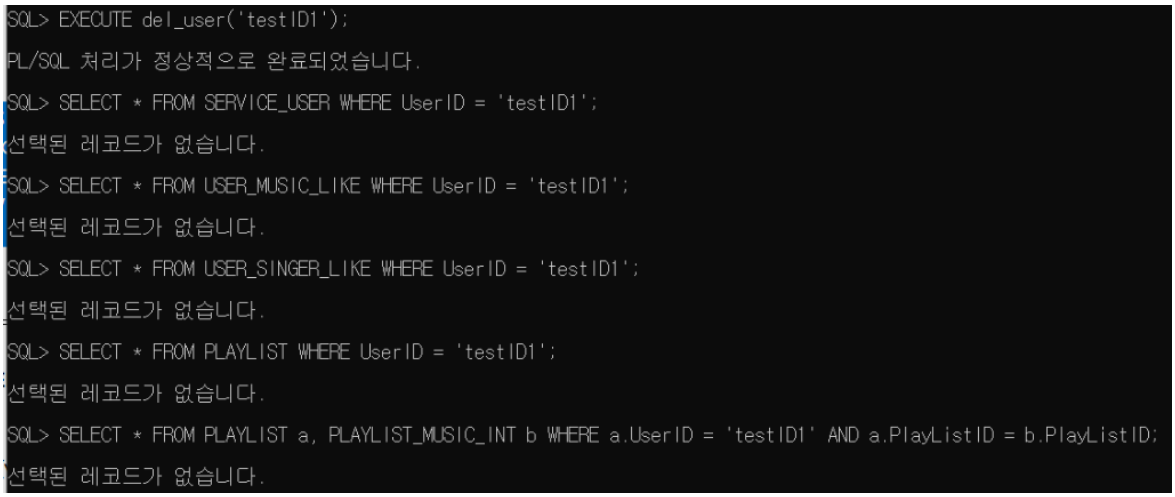
```
-- 회원 탈퇴 프로시저
CREATE OR REPLACE PROCEDURE del_user(v_userID IN VARCHAR2)
IS
BEGIN
        del_user_music_like(v_userID);
        del_user_singer_like(v_userID);
        del_playlists_of_user(v_userID);

        DELETE FROM SERVICE_USER
        WHERE UserID = v_userID;
        COMMIT;
EXCEPTION
        WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('ERR MESSAGE' || SQLERRM);
END;
/
```

```
-- 유저 삭제 -> 찜한 노래 삭제 프로시저
CREATE OR REPLACE PROCEDURE del_user_music_like(v_userID IN VARCHAR2)
IS
BEGIN
        DELETE FROM USER_MUSIC_LIKE
        WHERE UserID = v_userID;
        COMMIT;
EXCEPTION
        WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('ERR MESSAGE' || SQLERRM);
END;
/
```

```
-- 유저 삭제 -> 찜한 가수 삭제 프로시저
CREATE OR REPLACE PROCEDURE del_user_singer_like(v_userID IN VARCHAR2)
IS
BEGIN
        DELETE FROM USER_SINGER_LIKE
        WHERE UserID = v_userID;
        COMMIT;
EXCEPTION
        WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('ERR MESSAGE' || SQLERRM);
END;
```

```
/
-- 유저 삭제 -> 유저가 생성한 다수의 플레이리스트 삭제 프로시저
CREATE OR REPLACE PROCEDURE del_playlists_of_user(v_userID IN VARCHAR2)
IS
        not_found_data EXCEPTION;

        CURSOR playlist_list(v_UserID PLAYLIST.UserID%TYPE) IS
        SELECT PlayListID
        FROM PLAYLIST
        WHERE UserID = v_UserID;

        v_PlayListID PLAYLIST.PlayListID%TYPE;
BEGIN
        FOR PlaylistList IN playlist_list(v_userID) LOOP
                del_playlist_music(PlaylistList.PlayListID);
        END LOOP;

        DELETE FROM PLAYLIST
        WHERE UserID = v_userID;
        COMMIT;
EXCEPTION
        WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('ERR MESSAGE' || SQLERRM);
END;
/
```

[결과 화면]




[ 그림 24 ]

[ 표 23 ]

**[ 플레이리스트 삭제 프로시저 ]**

```
-- 플레이리스트 삭제 프로시저
CREATE OR REPLACE PROCEDURE del_playlist(v_playlistID IN VARCHAR2)
IS
BEGIN
        del_playlist_music(v_playlistID);

        DELETE FROM PLAYLIST
        WHERE PlayListID = v_playlistID;
        COMMIT;
EXCEPTION
        WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('ERR MESSAGE' || SQLERRM);
END;
/
```

```
-- 플레이리스트 삭제 -> 음악과의 N:M 관계 삭제 프로시저
CREATE OR REPLACE PROCEDURE del_playlist_music(v_playlistID IN VARCHAR2)
IS
BEGIN
        DELETE FROM PLAYLIST_MUSIC_INT
        WHERE PlayListID = v_playlistID;
EXCEPTION
        WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('ERR MESSAGE' || SQLERRM);
END;
/
```

[결과 면화]

```
SQL> EXECUTE del_playlist('PLAYLIST1');

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> SELECT * FROM PLAYLIST WHERE PlayListID = 'PLAYLIST1';

선택된 레코드가 없습니다.

SQL> SELECT * FROM PLAYLIST_MUSIC_INT WHERE PlayListID = 'PLAYLIST1';

선택된 레코드가 없습니다.
```

[ 그림 25 ]

[ 표 24 ]

**[ 플레이리스트 내의 음악 목록을 보여주는 프로시저 ]**

```
-- 플레이리스트 내의 음악 목록을 보여주는 프로시저
CREATE OR REPLACE PROCEDURE show_playlist(v_PlayListID IN VARCHAR2)
IS
        CURSOR music_list(v_playlistID PLAYLIST.PlayListID%TYPE) IS
        SELECT b.MusicTitle
        FROM PLAYLIST_MUSIC_INT a, MUSIC b
        WHERE PlayListID = v_playlistID AND a.MusicID = b.MusicID;

        v_PlayListName PLAYLIST.PlayListName%TYPE;
        v_MusicTitle MUSIC.MusicTitle%TYPE;
BEGIN
        SELECT PlayListName
        INTO v_PlayListName
        FROM PLAYLIST
        WHERE PlayListID = v_PlayListID;

        DBMS_OUTPUT.PUT_LINE('< ' || v_PlayListName || ' 플레이리스트 목록 >');
        FOR musiclst IN music_list(v_PlayListID) LOOP
                DBMS_OUTPUT.PUT_LINE(' - ' || musiclst.MusicTitle || ' ');
        END LOOP;
EXCEPTION
        WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('ERR MESSAGE' || SQLERRM);
END;
/
```

[결과 화면]

```
SQL> EXECUTE show_playlist('PLAYLIST2')
< YB 노래 모음 플레이리스트 목록 >
- 너를 보내고
- 사랑 two
- 가을 우체국 앞에서
```

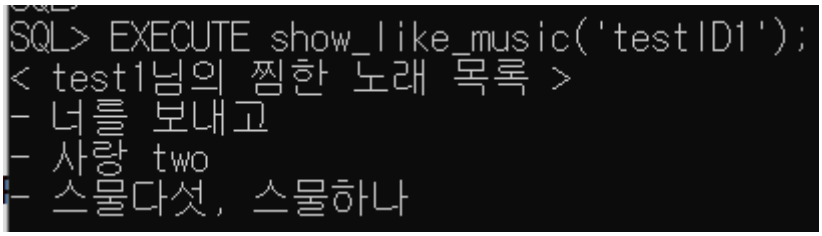
[ 그림 26 ]

[ 표 25 ]

**[ 사용자가 찜한 노래 목록 보여주는 프로시저 ]**

```
-- 사용자가 찜한 노래 목록 보여주는 프로시저
CREATE OR REPLACE PROCEDURE show_like_music(v_user IN VARCHAR2)
IS
        CURSOR music_list(v_userID USER_MUSIC_LIKE.UserID%TYPE) IS
        SELECT b.MusicTitle
        FROM USER_MUSIC_LIKE a, MUSIC b
        WHERE UserID = v_userID AND a.MusicID = b.MusicID;

        v_UserName SERVICE_USER.UserName%TYPE;
BEGIN
        SELECT UserName
        INTO v_UserName
        FROM SERVICE_USER
        WHERE UserID = v_user;

        DBMS_OUTPUT.PUT_LINE('< ' || v_UserName || '님의 찜한 노래 목록 >');
        FOR like_music IN music_list(v_user) LOOP
                DBMS_OUTPUT.PUT_LINE(' - ' || like_music.MusicTitle || ' ');
        END LOOP;
EXCEPTION
        WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('ERR MESSAGE' || SQLERRM);
END;
/
```

[결과 화면]




[ 그림 27 ]

[ 표 26 ]

**[ 사용자가 찜한 가수 목록 보여주는 프로시저 ]**

```
-- 사용자가 찜한 가수 목록 보여주는 프로시저
CREATE OR REPLACE PROCEDURE show_like_singer(v_user IN VARCHAR2)
IS
        CURSOR singer_list(v_userID USER_MUSIC_LIKE.UserID%TYPE) IS
        SELECT b.SingerName
        FROM USER_SINGER_LIKE a, SINGER b
        WHERE UserID = v_userID AND a.SingerID = b.SingerID;

        v_UserName SERVICE_USER.UserName%TYPE;
BEGIN
        SELECT UserName
        INTO v_UserName
        FROM SERVICE_USER
        WHERE UserID = v_user;

        DBMS_OUTPUT.PUT_LINE('< ' || v_UserName || '님의 찜한 가수 목록 >');
        FOR like_singer IN singer_list(v_user) LOOP
                DBMS_OUTPUT.PUT_LINE(' - ' || like_singer.SingerName || ' ');
        END LOOP;
EXCEPTION
        WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('ERR MESSAGE' || SQLERRM);
END;
/
```

[결과 화면]

```
SQL> EXECUTE show_like_singer('testID1');
< test1님의 찜한 가수 목록 >
- YB
- 자우림
```

[ 그림 28 ]

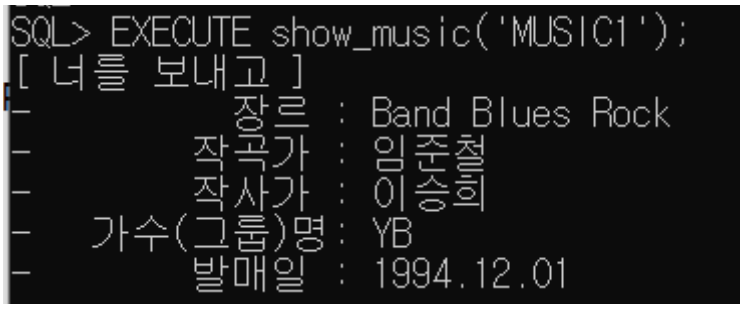
[ 표 27 ]

**[ 음악 정보를 출력하는 프로시저 ]**

```
-- 음악 정보를 출력하는 프로시저
CREATE OR REPLACE PROCEDURE show_music(v_musicID IN VARCHAR2)
IS
	CURSOR genre_list(v_MusicID MUSIC_GENRE_INT.MusicID%TYPE) IS
	SELECT GenreName
	FROM MUSIC_GENRE_INT
	WHERE MusicID = v_MusicID;

	CURSOR composer_list(v_MusicID MUSIC_GENRE_INT.MusicID%TYPE) IS
	SELECT b.ComposerName
	FROM MUSIC_COMPOSER_INT a, COMPOSER b
	WHERE a.ComposerID = b.ComposerID AND a.MusicID = v_MusicID;

	CURSOR lylicist_list(v_MusicID MUSIC_GENRE_INT.MusicID%TYPE) IS
	SELECT b.LylicistName
	FROM MUSIC_LYLICIST_INT a, LYLICIST b
	WHERE a.LylicistID = b.LylicistID AND a.MusicID = v_MusicID;

	v_music MUSIC%ROWTYPE;
	v_SingerName SINGER.SingerName%TYPE;
BEGIN
	SELECT MusicTitle, SingerID, MusicReleaseDate
	INTO v_music.MusicTitle, v_music.SingerID, v_music.MusicReleaseDate
	FROM MUSIC
	WHERE MusicID = v_musicID;

	SELECT SingerName
	INTO v_SingerName
	FROM SINGER
	WHERE SingerID = v_music.SingerID;

	DBMS_OUTPUT.PUT_LINE('[ ' || v_music.MusicTitle || ' ]');
	DBMS_OUTPUT.PUT('- ' || lpad('장르 : ', 16));
	FOR GenreList IN genre_list(v_musicID) LOOP
		DBMS_OUTPUT.PUT(GenreList.GenreName || ' ');
	END LOOP;
	DBMS_OUTPUT.NEW_LINE;
	DBMS_OUTPUT.PUT('- ' || lpad('작곡가 : ', 16));
	FOR ComposerList IN composer_list(v_musicID) LOOP
```

```
                DBMS_OUTPUT.PUT(ComposerList.ComposerName || ' ');
        END LOOP;
        DBMS_OUTPUT.NEW_LINE;
        DBMS_OUTPUT.PUT('- ' || lpad('작사가 : ', 16));
        FOR LylicistList IN lylicist_list(v_musicID) LOOP
                DBMS_OUTPUT.PUT(LylicistList.LylicistName || ' ');
        END LOOP;
        DBMS_OUTPUT.NEW_LINE;
        DBMS_OUTPUT.PUT_LINE('- ' || lpad('가수(그룹)명: ', 16) || v_SingerName);
        DBMS_OUTPUT.PUT_LINE('-    '    ||    lpad('발매일    :    ',    16)    ||
v_music.MusicReleaseDate);
EXCEPTION
        WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('ERR MESSAGE' || SQLERRM);
END;
/
```

[결과 화면]



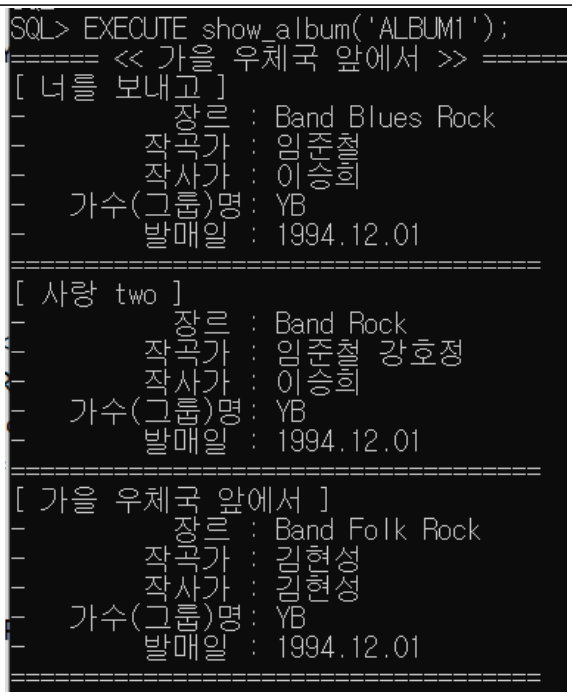

[ 그림 29 ]

[ 표 28 ]

**[ 앨범 정보를 출력하는 프로시저 (음악 정보 출력 프로시저 사용) ]**

```
-- 앨범 정보를 출력하는 프로시저 (음악 정보 출력 프로시저 사용)
CREATE OR REPLACE PROCEDURE show_album(v_albumID IN VARCHAR2)
IS
        v_albumName ALBUM.AlbumTitle%TYPE;
BEGIN
        SELECT AlbumTitle
        INTO v_albumName
        FROM ALBUM
        WHERE AlbumID = v_albumID;

        DBMS_OUTPUT.PUT_LINE('======  <<  ' || v_albumName || ' >>
======');
        FOR music_list IN (SELECT MusicID FROM MUSIC WHERE AlbumID =
v_albumID) LOOP
                show_music(music_list.MusicID);

DBMS_OUTPUT.PUT_LINE('==================================
=');
        END LOOP;
EXCEPTION
        WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('ERR MESSAGE' || SQLERRM);
END;
/
```

[결과 화면]


SQL> EXECUTE show_album('ALBUM1');
====== << 가을 우체국 앞에서 >> ======
[ 너를 보내고 ]
- 장르 : Band Blues Rock
- 작곡가 : 임준철
- 작사가 : 이승희
- 가수(그룹)명 : YB
- 발매일 : 1994.12.01
==========
[ 사랑 two ]
- 장르 : Band Rock
- 작곡가 : 임준철 강호정
- 작사가 : 이승희
- 가수(그룹)명 : YB
- 발매일 : 1994.12.01
==========
[ 가을 우체국 앞에서 ]
- 장르 : Band Folk Rock
- 작곡가 : 김현성
- 작사가 : 김현성
- 가수(그룹)명 : YB
- 발매일 : 1994.12.01
==========

[ 그림 30 ]

[ 표 29 ]

**[ 가수(그룹) 정보를 출력하는 프로시저 ]**

```
-- 가수(그룹) 정보를 출력하는 프로시저
CREATE OR REPLACE PROCEDURE show_singer(v_singerID IN VARCHAR2)
IS
	CURSOR member_list(v_SingerID SINGER.SingerID%TYPE) IS
	SELECT b.MemberName
	FROM SINGER_MEMBER_INT a, MEMBER b
	WHERE a.SingerID = v_SingerID AND a.MemberID = b.MemberID;

	v_singer SINGER%ROWTYPE;
BEGIN

	SELECT SingerName, SingerDebutDate
	INTO v_singer.SingerName, v_singer.SingerDebutDate
	FROM SINGER
	WHERE SingerID = v_singerID;

	DBMS_OUTPUT.PUT_LINE('[ ' || v_singer.SingerName|| ' ]');
	DBMS_OUTPUT.PUT('- ' || lpad('구성원 : ', 10));
	FOR MemberList IN member_list(v_singerID) LOOP
		DBMS_OUTPUT.PUT(MemberList.MemberName || ' ');
	END LOOP;
	DBMS_OUTPUT.NEW_LINE;
	DBMS_OUTPUT.PUT_LINE('- ' || lpad('데뷔일 : ', 10) ||
v_singer.SingerDebutDate);
EXCEPTION
	WHEN OTHERS THEN
		DBMS_OUTPUT.PUT_LINE('ERR MESSAGE' || SQLERRM);
END;
/
```

[결과 화면]

```
SQL> EXECUTE show_singer('SINGER1');
[ YB ]
-  구성원 : 윤도현 박태희 김진원 허준 Scott
-  데뷔일 : 1994.12.01
```

[ 그림 31 ]

[ 표 30 ]

**[ 해당 가수의 인기도를 측정하기 위한 함수 (찜 당한 횟수를 통해) ]**

```
-- 해당 가수의 인기도를 측정하기 위한 함수 (찜 당한 횟수를 통해)
CREATE OR REPLACE FUNCTION popularity_singer
(v_singerID IN VARCHAR2) RETURN NUMBER
IS
        cnt NUMBER := 0;
BEGIN
        SELECT COUNT(*)
        INTO cnt
        FROM USER_SINGER_LIKE
        WHERE SingerID = v_singerID;
RETURN cnt;
EXCEPTION
        WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('ERR MESSAGE' || SQLERRM);
END;
/
```

[결과 화면]

```
SQL> VAR popularity NUMBER;
SQL> EXECUTE :popularity := popularity_singer('SINGER2');

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> print popularity;

POPULARITY
----------
         2
```

[ 그림 32 ]

[ 표 31 ]

**[ 해당 노래의 인기도를 측정하기 위한 함수 (찜 당한 횟수를 통해) ]**

```
-- 해당 노래의 인기도를 측정하기 위한 함수 (찜 당한 횟수를 통해)
CREATE OR REPLACE FUNCTION popularity_music
(v_musicID IN VARCHAR2) RETURN NUMBER
IS
        cnt NUMBER := 0;
BEGIN
        SELECT COUNT(*)
        INTO cnt
        FROM USER_MUSIC_LIKE
        WHERE MusicID = v_musicID;
RETURN cnt;
EXCEPTION
        WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('ERR MESSAGE' || SQLERRM);
END;
/
```

[결과 화면]

```
SQL> VAR popularity NUMBER;
SQL> EXECUTE :popularity := popularity_music('MUSIC1');

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> print popularity;

POPULARITY
----------
         1
```

[ 그림 33 ]

[ 표 32 ]

## 5. 평가 및 결론

### 5.1 평가

이 프로젝트에서는 먼저 제시한 데이터베이스 설계안을 바탕으로 데이터 모델링을 구성하였습니다. 그 후 데이터 모델링을 기반으로 논리적 설계를 진행하고 그 후 이를 기반으로 물리적 설계를 진행하였습니다. 다음 설계를 바탕으로 구현된 데이터베이스는 사용자 정보, 음악 정보, 앨범 정보, 플레이리스트 정보, 가수 정보 등을 각 테이블 간의 관계와 앞에서 생성한 프로시저, 함수, 트리거 등을 통하여 효율적으로 관리할 수 있도록 구성되었습니다.

구현된 데이터베이스는 ORACLE을 통해 구성되었으며, 이를 통해 해당 서비스의 사용자들은 다양한 음악을 쉽고 편리하고 빠르게 스트리밍할 수 있습니다.

### 5.2 결론

이 프로젝트를 통해 데이터베이스 모델링을 경험하고 ERWIN을 사용하여 논리적, 물리적 설계를 구성하는 경험을 통해 ERWIN 프로그램의 숙련도를 향상시킬 수 있었습니다. 또한, 해당 물리적 설계를 통해 생성한 데이터베이스를 ORACLE을 통해 데이터를 삽입, 삭제를 하여 관리하고 프로시저, 함수, 트리거를 추가하여 해당 데이터베이스의 효율성과 편리성을 높이도록 하는 경험을 통해 ORACLE의 숙련도를 향상시킬 수 있었습니다.

이를 통해 해당 서비스의 사용자들에게 다양한 음악을 쉽고 편리하고 빠르게 스트리밍할 수 있는 서비스를 제공할 수 있게 되었습니다. 이를 통해 데이터베이스 프로그래밍에 대한 이해도를 높일 수 있었으며, 향후 데이터베이스 관련 프로젝트를 진행할 때에도 보다 능숙하게 작업할 수 있을 것입니다.

## 6. 참고문헌

[1] 이수현 외 1인, 디지털 음악 시장의 승자, 스트리밍 서비스, 한국저작권위원회, 2023