# 数据库设计——播放子系统

1. **需求分析**

（1）设计背景

为了进一步提升音乐播放器应用的用户体验和功能完整性，我们需要开发一个直观且功能丰富的音乐播放界面。该界面不仅要支持基本的音乐播放功能，还需要提供便捷的音乐管理和用户互动功能。同时，界面的设计应考虑到用户的隐私和数据安全，确保用户的使用安全性和数据的完整性。

（2）功能需求

1. 音乐播放

需要存储音乐的详细信息，包括歌曲名、歌手、专辑、封面路径、时长等。

支持存储和管理播放列表信息，包含用户创建的播放列表及其包含的音乐。

2. 用户交互

需要存储用户喜欢的歌曲，包括用户ID和收藏的音乐ID。

3. 音乐管理

音乐搜索与筛选：

需要支持快速查询和筛选音乐信息，包括根据歌曲名、歌手、专辑等字段进行搜索。

1. 安全性要求

支持防止暴力破解和常见的网络攻击（如SQL注入、XSS）。

实现基于角色的访问控制（RBAC），确保用户只能执行其角色权限范围内的操作。

授予数据库用户尽可能少的权限，确保应用程序用户只拥有执行必要操作的权限。

1. 日志记录

记录用户的所有重要操作，包括音乐收藏，删除，修改等。

日志需要具备审计功能，以备安全分析。

（3）性能需求

1.快速响应时间

用户注册、登录和其他操作的响应时间应尽可能短，确保良好的用户体验。目标是每个操作的响应时间不超过200毫秒。

对频繁查询的表（如 music\_list表）进行索引优化，确保查询效率。

2.可扩展性

数据库应具备良好的可扩展性，支持未来的功能扩展和数据增长。系统应能够轻松增加新的用户角色、权限和功能模块。

考虑未来可能增加的分布式数据库支持，确保系统能够横向扩展以处理更多用户和数据。

3.容错性

系统应具备良好的容错性，能够在硬件或软件故障发生时迅速恢复正常工作。需要实现数据库的高可用性和故障转移机制。

定期备份数据库，以防止数据丢失。制定完善的数据恢复策略，确保在发生故障时能够快速恢复数据。

4.数据一致性

系统应确保数据的一致性，尤其是在高并发的情况下。需要使用事务机制来保证数据操作的原子性、一致性、隔离性和持久性（ACID）。

1. 配置环境需求

1.操作系统

操作系统: Windows 11

版本: 专业版

体系结构: 64位（x64）

2.数据库

数据库管理系统（DBMS）: MySQL

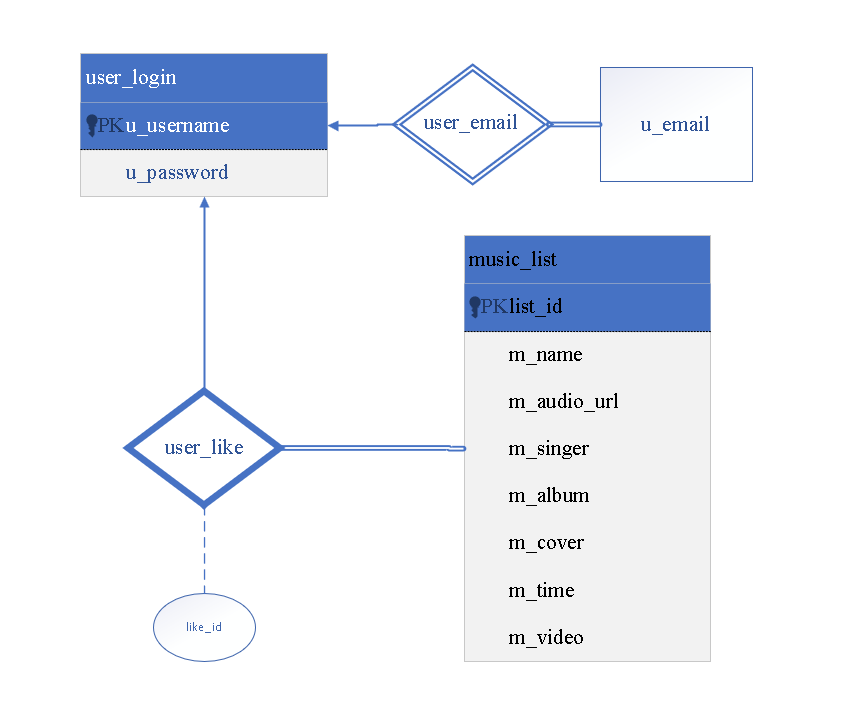
版本: MySQL 8.0 或以上

3.数据库用户权限

数据库管理员: 拥有所有权限，负责数据库的创建、管理和维护。

应用程序用户: 拥有必要的权限以执行所需的数据库操作，如读写权限。

1. **概念结构设计**



1. **逻辑结构设计**
2. 歌曲表  
    music\_list 表存储音乐播放器应用中的歌曲信息。每条记录包含一个唯一标识符，用于区分不同的歌曲，记录歌曲的名字、音频地址、歌手名字、专辑名称、封面地址、歌曲时长，以及MV地址。其中，唯一标识符是主键，并自动递增，确保每首歌曲的必要信息完整。音频地址、歌曲名字、歌手名字和歌曲时长字段不能为空。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **music\_list** | | | | | | | |
| **字段名称** | **字段内容** | **数据类型** | **是否非空** | **是否为主键** | **是否为外键** | **外键参考** | **备注** |
| **music\_id** | **歌曲唯一标识** | **INT** | **是** | **是** | **否** |  | **AUTO\_INCREMENT** |
| **m\_name** | **歌曲名字** | **VARCHAR(100)** | **是** | **否** | **否** |  |  |
| **m\_audio\_url** | **歌曲音频地址** | **VARCHAR(255)** | **是** | **否** | **否** |  |  |
| **m\_singer** | **歌手名字** | **VARCHAR(100)** | **是** | **否** | **否** |  |  |
| **m\_album** | **专辑名称** | **VARCHAR(100)** | **否** | **否** | **否** |  |  |
| **m\_cover** | **封面地址** | **VARCHAR(255)** | **否** | **否** | **否** |  |  |
| **m\_time** | **歌曲时长** | **VARCHAR(100)** | **是** | **否** | **否** |  |  |
| **m\_video** | **MV地址** | **VARCHAR(255)** | **否** | **否** | **否** |  |  |

表1 歌曲表

1. 用户喜欢歌曲表

user\_like 表存储用户喜欢的歌曲信息。每条记录包含用户的唯一标识符和用户喜欢的歌曲标识符。用户的唯一标识符和歌曲标识符共同组成联合主键，以确保每个用户对每首歌的喜欢记录是唯一的。这两个字段都是非空字段，用户的唯一标识符引用 user\_login 表，当用户被删除时，相应的喜欢记录也会被删除；歌曲标识符引用 music\_list 表，当歌曲被删除时，相应的喜欢记录也会被删除。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **user\_like** | | | | | | | |
| **字段名称** | **字段内容** | **数据类型** | **是否非空** | **是否为主键** | **是否为外键** | **外键参考** | **备注** |
| **u\_username** | **用户的唯一标识符** | **VARCHAR(50)** | **是** | **是** | **是** | **user\_login.**  **u\_username** | **级联删除** |
| **like\_id** | **用户喜欢的歌曲** | **VARCHAR(50)** | **是** | **是** | **是** | **music\_list.**  **music\_id** | **级联删除** |

表2 用户喜欢歌曲表

1. **日志记录设计**

（1）日志表设计

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **music\_logs** | | | | | | | |
| **字段名称** | **字段内容** | **数据类型** | **是否非空** | **是否为主键** | **是否为外键** | **外键参考** | **备注** |
| **log\_id** | **日志的唯一标识符** | **INT** | **是** | **是** | **否** |  | **AUTO\_INCREMENT** |
| **u\_username** | **用户的唯一标识符** | **VARCHAR(50)** | **否** | **否** | **是** | **user\_login.**  **u\_username** |  |
| **l\_action** | **日志类型** | **VARCHAR(50)** | **是** | **否** | **否** |  |  |
| **l\_description** | **日志描述** | **VARCHAR(255)** | **否** | **否** | **否** |  |  |
| **l\_time** | **日志创建时间** | **VARCHAR(50)** | **是** | **否** | **否** |  |  |

表3 日志表

1. **权限分配**

（1）管理员用户：

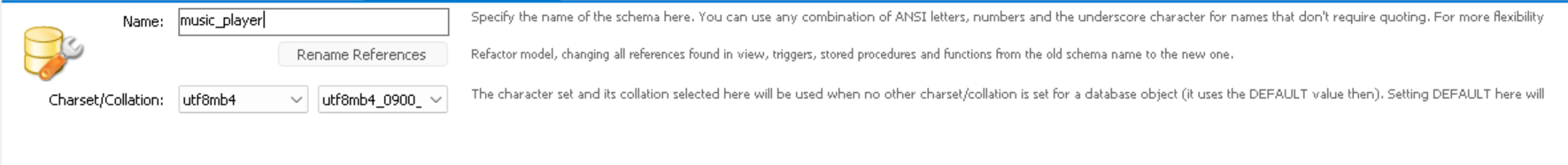
仅拥有日志表的所有权限，无权查看、修改应用程序用户的信息。

1. 应用程序用户：

仅限于应用程序运行所需的最小权限集，即歌曲表和用户喜欢歌曲表。

1. **数据库实现**
2. 创建数据库

使用utf8mb4字符集, 数据库排序规则使用utf8mb4\_general\_ci。



1. 创建表

DROP TABLE IF EXISTS music\_logs;-- music\_logs，先进行删除

DROP TABLE IF EXISTS user\_like;-- user\_like，先进行删除

DROP TABLE IF EXISTS music\_list; -- music\_list，先进行删除

-- 创建音乐列表

CREATE TABLE music\_list (

music\_id INT AUTO\_INCREMENT, -- 列表的唯一标识符，自增，主键

m\_name VARCHAR(100) NOT NULL, -- 歌曲名字，非空，长度限制为100字符

m\_audio\_url VARCHAR(255) NOT NULL, -- 歌曲音频地址，非空，长度限制为255字符

m\_singer VARCHAR(100) NOT NULL, -- 歌手名字，非空，长度限制为100字符

m\_album VARCHAR(100), -- 专辑名称，长度限制为100字符，可以为空

m\_cover VARCHAR(255), -- 封面地址，长度限制为255字符，可以为空

m\_time TIME NOT NULL, -- 歌曲时长，非空，使用 TIME 格式

m\_video VARCHAR(255), -- MV地址，长度限制为255字符，可以为空

PRIMARY KEY (music\_id)

);

-- 创建用户喜欢列表

CREATE TABLE user\_like (

u\_username VARCHAR(50) NOT NULL, -- 当前元组所对应的用户的标识符，非空，外键

like\_id INT NOT NULL, -- 当前元组所对应的音乐列表的标识符，非空，外键

PRIMARY KEY (u\_username, like\_id), -- 联合主键，由用户标识符和音乐列表标识符组成

FOREIGN KEY (u\_username) REFERENCES user\_login(u\_username) ON DELETE CASCADE, -- 外键约束，引用 user\_login 表的 u\_username，支持级联删除

FOREIGN KEY (like\_id) REFERENCES music\_list(music\_id) ON DELETE CASCADE -- 外键约束，引用 music\_list 表的 music\_id，支持级联删除

);

-- 创建日志表

CREATE TABLE music\_logs (

log\_id INT AUTO\_INCREMENT PRIMARY KEY, -- 日志的唯一标识符，自增

u\_username VARCHAR(50), -- 用户标识符

l\_action VARCHAR(50) NOT NULL, -- 操作类型

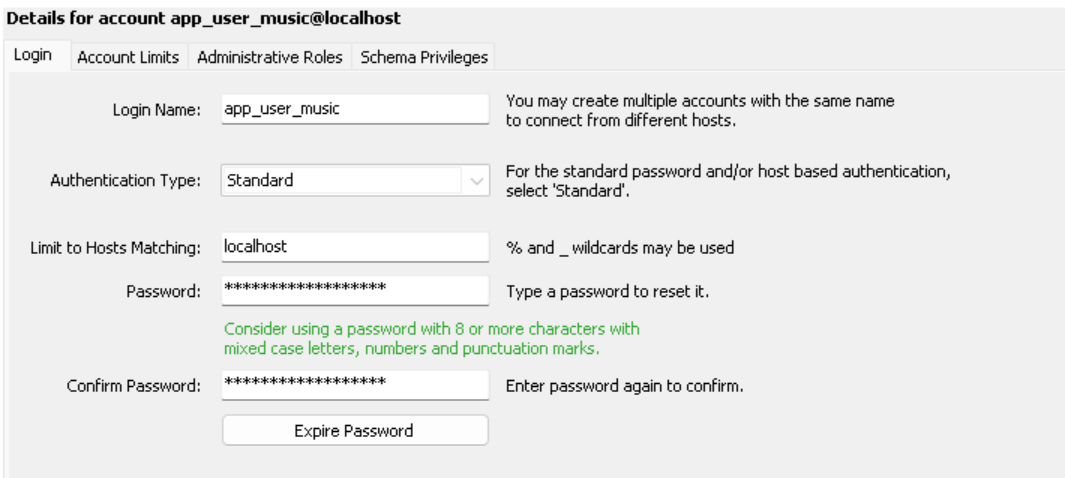
l\_description VARCHAR(255), -- 操作描述

l\_time DATETIME DEFAULT CURRENT\_TIMESTAMP, -- 操作时间，默认当前时间

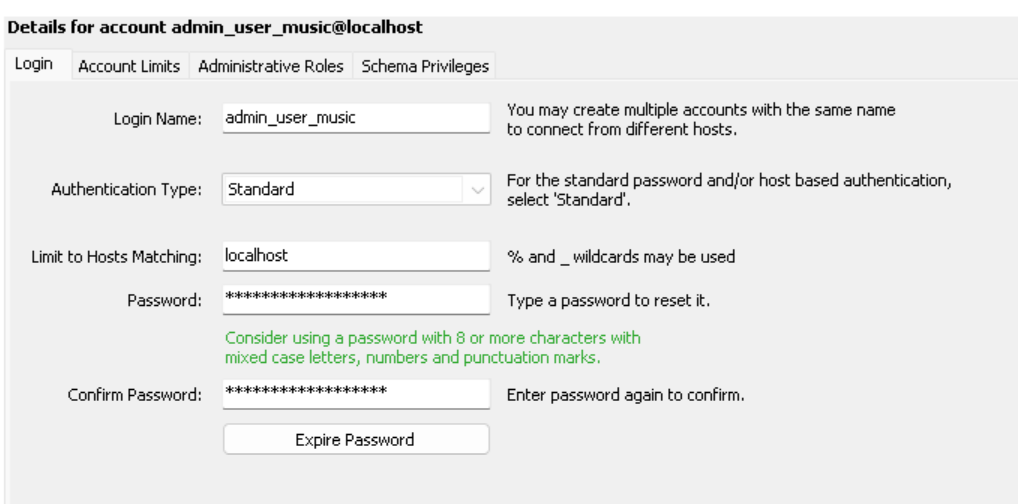
FOREIGN KEY (u\_username) REFERENCES user\_login(u\_username)

);

1. 创建用户
2. 应用程序用户



1. 管理用户



1. 权限授予
2. 切换到root用户。
3. 为管理用户授予music\_logs上的所有权限。

GRANT ALL PRIVILEGES ON music\_player.music\_logs TO 'admin\_user\_music'@'localhost' WITH GRANT OPTION;

1. 为应用程序用户授予music\_list表和 user\_like表上的所有权限。

GRANT ALL PRIVILEGES ON music\_player.music\_list TO 'app\_user\_music'@'localhost';

GRANT ALL PRIVILEGES ON music\_player.user\_like TO 'app\_user\_music'@'localhost';

1. 维护日志

-- 删除可能存在的旧触发器

DROP TRIGGER IF EXISTS trg\_user\_like\_insert;

DROP TRIGGER IF EXISTS trg\_user\_like\_delete;

DROP TRIGGER IF EXISTS trg\_music\_list\_insert;

DROP TRIGGER IF EXISTS trg\_music\_list\_update;

DROP TRIGGER IF EXISTS trg\_music\_list\_delete;

DELIMITER //

-- 创建触发器：在插入到 user\_like 表时记录日志

CREATE TRIGGER trg\_user\_like\_insert

AFTER INSERT ON user\_like

FOR EACH ROW

BEGIN

INSERT INTO music\_logs (u\_username, l\_action, l\_description)

VALUES (NEW.u\_username, 'INSERT', CONCAT('User liked song with id ', NEW.like\_id));

END;

//

-- 创建触发器：在删除 user\_like 表时记录日志

CREATE TRIGGER trg\_user\_like\_delete

AFTER DELETE ON user\_like

FOR EACH ROW

BEGIN

INSERT INTO music\_logs (u\_username, l\_action, l\_description)

VALUES (OLD.u\_username, 'DELETE', CONCAT('User unliked song with id ', OLD.like\_id));

END;

//

-- 创建触发器：在插入到 music\_list 表时记录日志

CREATE TRIGGER trg\_music\_list\_insert

AFTER INSERT ON music\_list

FOR EACH ROW

BEGIN

INSERT INTO music\_logs (u\_username, l\_action, l\_description)

VALUES (NULL, 'INSERT', CONCAT('Added song with id ', NEW.music\_id, ' and name ', NEW.m\_name));

END;

//

-- 创建触发器：在更新 music\_list 表时记录日志

CREATE TRIGGER trg\_music\_list\_update

AFTER UPDATE ON music\_list

FOR EACH ROW

BEGIN

INSERT INTO music\_logs (u\_username, l\_action, l\_description)

VALUES (NULL, 'UPDATE', CONCAT('Updated song with id ', NEW.music\_id, ' from ', OLD.m\_name, ' to ', NEW.m\_name));

END;

//

-- 创建触发器：在删除 music\_list 表时记录日志

CREATE TRIGGER trg\_music\_list\_delete

AFTER DELETE ON music\_list

FOR EACH ROW

BEGIN

INSERT INTO music\_logs (u\_username, l\_action, l\_description)

VALUES (NULL, 'DELETE', CONCAT('Deleted song with id ', OLD.music\_id, ' and name ', OLD.m\_name));

END;

//

DELIMITER ;

所有代码详见附件。

1. **数据库测试**
2. 基本结构测试

1.检查表结构

确保所有表和列存在，并且数据类型和约束正确。

-- 检查 music\_list 表的结构

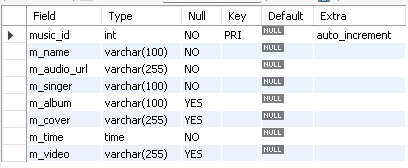
DESCRIBE music\_list;

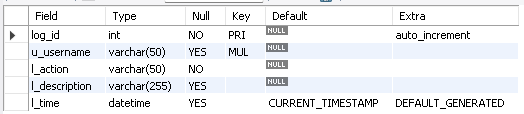
-- 检查 music\_logs 表的结构

DESCRIBE music\_logs;

-- 检查 user\_like 表的结构

DESCRIBE user\_like;



、



1. 数据操作测试
2. 插入歌单

INSERT INTO music\_list (m\_name, m\_audio\_url, m\_singer, m\_album, m\_cover, m\_time, m\_video) VALUES

('打上花火', './audios/打上花火.mp3', '米津玄師', '打上花火 (初回限定盤)', './music-videos/打上花火.jpg', '04:49', './music-videos/打上花火.mp4'),

('前前前世', './audios/前前前世.mp3', 'RADWIMPS', '君の名は', './music-videos/前前前世.jpg', '04:52', './music-videos/前前前世.mp4'),

('Rain', './audios/言叶之庭.mp3', '秦基博 (はたもとひろ)', '言ノ葉', './music-videos/言叶之庭.jpg', '04:31', './music-videos/言叶之庭.mp4'),

('Bad Apple', './audios/Bad Apple.mp3', 'nomico (のみこ)', 'Lovelight', './music-videos/Bad Apple.jpg', '03:39', './music-videos/Bad Apple.mp4'),

('COLORs', './audios/Colors.mp3', '泽野弘之·秦基博', 'WIT STUDIO', './music-videos/Colors.jpg', '03:06', './music-videos/Colors.mp4'),

('LOSER', './audios/LOSER.mp3', '米津玄師', 'LOSER / ナンバーナイン', './music-videos/LOSER.jpg', '04:02', './music-videos/LOSER.mp4'),

('One Last Kiss', './audios/One Last Kiss.mp3', '宇多田光', 'One Last Kiss', './music-videos/One Last Kiss.jpg', '04:12', './music-videos/One Last Kiss.mp4'),

('YOKU', './audios/YOKU.mp3', 'Eve', '廻人', './music-videos/YOKU.jpg', '02:46', './music-videos/YOKU.mp4'),

('グランドエスケープ', './audios/グランドエスケープ.mp3', '三浦透子 / RADWIMPS', '天気の子', './music-videos/グランドエスケープ.jpg', '05:38', './music-videos/グランドエスケープ.mp4'),

('エウテルペ', './audios/エウテルペ.mp3', 'EGOIST', 'Alter Ego', './music-videos/エウテルペ.jpg', '03:47', './music-videos/罪恶王冠.mp4'),

('Infected', './audios/Infected.mp3', 'Starset', 'Infected', './music-videos/Infected.jpg', '03:37', './music-videos/Infected.mp4'),

('Unbecoming', './audios/Unbecoming.mp3', 'Starset', 'Unbecoming', './music-videos/Unbecoming.jpg', '03:48', './music-videos/Unbecoming.mp4'),

('Monster', './audios/Monster.mp3', 'Starset', 'Monster', './music-videos/Monster.jpg', '03:03', './music-videos/Monster.mp4'),

('Telescope', './audios/Telescope.mp3', 'Starset', 'Telescope', './music-videos/Telescope.jpg', '03:48', './music-videos/Telescope.mp4'),

('The Everlasting Guilty Crown', './audios/The Everlasting Guilty Crown.mp3', 'EGOIST', 'The Everlasting Guilty Crown', './music-videos/The Everlasting Guilty Crown.jpg', '01:33', './music-videos/The Everlasting Guilty Crown.mp4'),

('不问天', './audios/不问天.mp3', 'papaw泡泡', '不问天', './music-videos/不问天.jpg', '04:13', './music-videos/不问天.mp4'),

('时光盲盒', './audios/时光盲盒.mp3', 'ChiliChill', '时光盲盒', './music-videos/时光盲盒.jpg', '04:46', './music-videos/时光盲盒.mp4'),

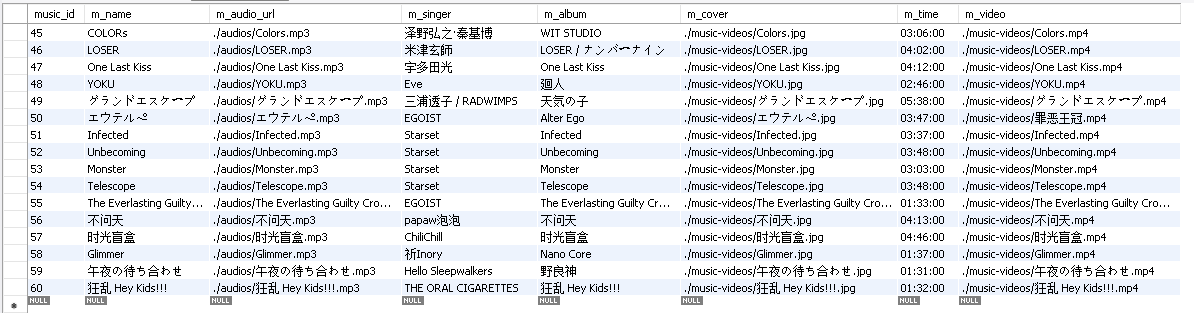
('Glimmer', './audios/Glimmer.mp3', '祈Inory', 'Nano Core', './music-videos/Glimmer.jpg', '01:37', './music-videos/Glimmer.mp4'),

('午夜の待ち合わせ', './audios/午夜の待ち合わせ.mp3', 'Hello Sleepwalkers', '野良神', './music-videos/午夜の待ち合わせ.jpg', '01:31', './music-videos/午夜の待ち合わせ.mp4'),

('狂乱 Hey Kids!!!', './audios/狂乱 Hey Kids!!!.mp3', 'THE ORAL CIGARETTES', '狂乱 Hey Kids!!!', './music-videos/狂乱 Hey Kids!!!.jpg', '01:32', './music-videos/狂乱 Hey Kids!!!.mp4');

查看表中信息：

SELECT \* FROM music\_player.music\_list;



1. 登录账号，将歌曲加入我喜欢歌单：

登录过程省略，账号为RyanQT7

点击喜欢按钮，将歌曲加入歌单：  
 



检查表中数据：

SELECT \* FROM music\_player.user\_like;



查看日志：



取消喜欢：



查看表中数据：  
 

查看日志：  
 

1. 权限测试

查看管理用户权限：

SHOW GRANTS FOR 'admin\_user\_music'@'localhost'



查看应用用户权限：

SHOW GRANTS FOR 'app\_user\_music'@'localhost'  
 