# 数据库设计

一、需求分析

（1）设计背景

为了提升音乐播放器应用的安全性和用户体验，我们需要开发一个可靠且安全的用户登录界面。该登录系统不仅要支持基础的用户注册和登录功能，还需满足用户对隐私和数据安全的高要求。

（2）功能需求

1.用户注册：

用户可以注册新账户，填写基本信息如用户名、密码、邮箱等。

系统需要验证用户输入的有效性（不能含有空格等特殊字符，用户名唯一）。

2.用户登录

用户使用用户名和密码进行登录。

系统需要验证用户输入的有效性和正确性。

系统需要保证用户的账号信息的安全性。

3.密码管理

用户可以重置密码，需要输入原密码进行安全认证。

系统需要保证密码的高度安全性。

4.安全性要求

密码需要使用加密算法存储（如 bcrypt）。

支持防止暴力破解和常见的网络攻击（如SQL注入、XSS）。

5.日志记录

记录用户登录、注销、密码更改等重要操作日志。

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 users 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

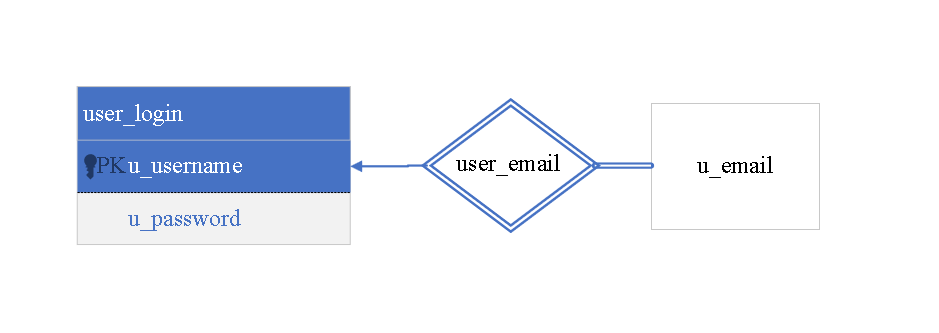
版本: MySQL 8.0 或以上

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

（1）用户账号信息表。对用户的账号和密码进行描述，主要内容包括用户账号，用户密码。主要用于用户的注册，登录，注销，以及修改。各字段的名称和数据类型及主键设置情况见表1。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **user\_login** | | | | | | | |
| **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** | **用户名** | **VARCHAR(50)** | **是** | **是** | **否** |  |  |
| **u\_password** | **密码字段** | **VARCHAR(255)** | **是** | **否** | **否** |  |  |

表1 用户账号信息表

（1）用户邮箱信息表。对用户的邮箱进行描述，主要内容包括用户邮箱。主要用于用户的注册，以及密码的找回。各字段的名称和数据类型及主键设置情况见表2。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **user\_email** | | | | | | | |
| **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** | **否** | **user\_login.**  **u\_username** | **级联删除** |
| **u\_email** | **用户的邮箱** | **VARCHAR(50)** | **否** | **否** | **否** |  |  |

表2用户邮箱信息表

1. 日志记录设计
2. 日志表设计

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **user\_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)** | **是** | **是** | **否** | **user\_login.**  **u\_username** |  |
| **l\_type** | **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** |  |  |

1. 权限分配

（1）管理员用户：

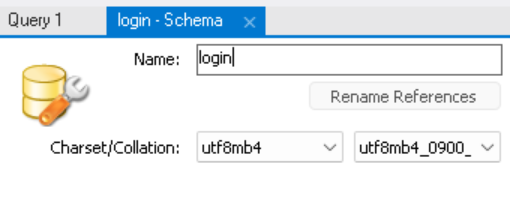
拥有对所有数据库对象（包括表、视图、存储过程等）的完全访问权限。

1. 应用程序用户：

仅限于应用程序运行所需的最小权限集。

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. 创建表格

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

DROP TABLE IF EXISTS user\_email;-- 如果存在user\_email，先进行删除

DROP TABLE IF EXISTS user\_login; -- 如果存在user\_login，先进行删除

-- 创建用户账户信息表

CREATE TABLE user\_login (

u\_username VARCHAR(50) NOT NULL PRIMARY KEY, -- 用户名，长度限制为50字符，要求唯一且不为空

u\_password VARCHAR(255) NOT NULL -- 密码字段，长度为255字符，存储加密后的密码

);

-- 创建用户邮箱信息表

CREATE TABLE user\_email (

u\_username VARCHAR(50) NOT NULL PRIMARY KEY, -- 用户的唯一标识符，作为外键引用 user\_login 表的 u\_username

u\_email VARCHAR(50), -- 用户的邮箱，长度限制为50字符，可以为空

FOREIGN KEY (u\_username) REFERENCES user\_login(u\_username) ON DELETE CASCADE-- 为了维护数据库的一致性，采用级联删除

);

-- 创建登录系统日志表

CREATE TABLE user\_logs (

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

u\_username VARCHAR(50), -- user\_login 表的 u\_username，用于记录操作日志的用户

l\_type VARCHAR(50) NOT NULL, -- 日志类型，描述日志的类别或操作类型（例如登录、注销、密码更改等）

l\_description VARCHAR(255), -- 日志描述，记录详细的操作信息或事件描述

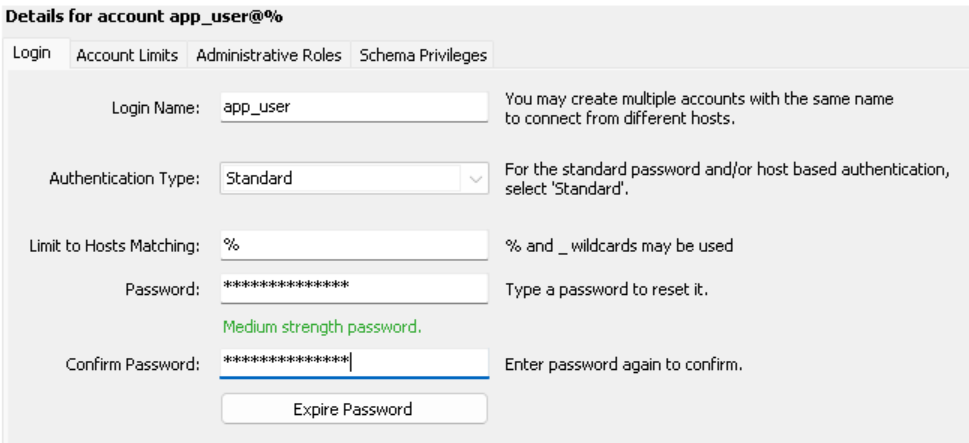
l\_time DATETIME DEFAULT CURRENT\_TIMESTAMP, -- 日志创建时间，使用当前时间戳作为默认值

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

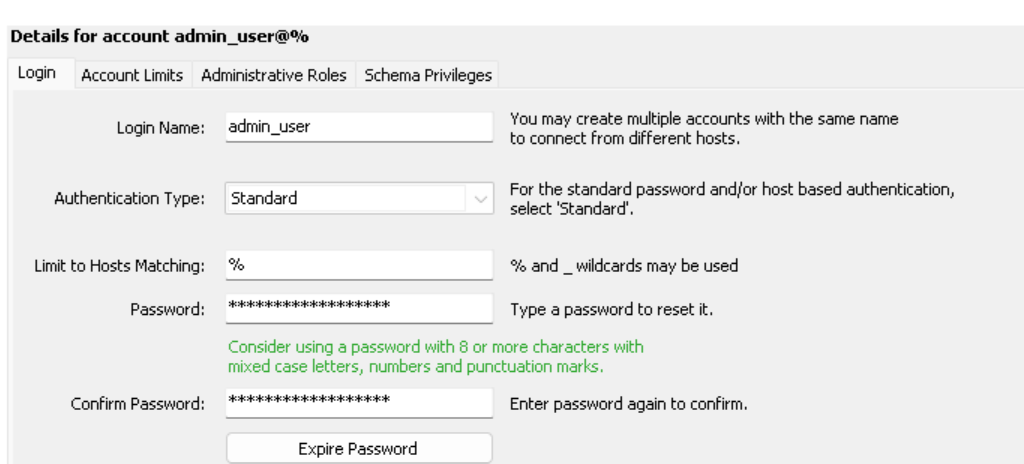
);

1. 创建用户

1.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login上的所有权限。

GRANT ALL PRIVILEGES ON login.\* TO 'admin\_user'@'%' WITH GRANT OPTION;

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

GRANT ALL PRIVILEGES ON login.user\_login TO 'app\_user'@'%';

GRANT ALL PRIVILEGES ON login.user\_email TO 'app\_user'@'%';

1. 维护日志

-- 删除已存在的触发器

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

DROP TRIGGER IF EXISTS trg\_user\_login\_update;

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

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

DROP TRIGGER IF EXISTS trg\_user\_email\_update;

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

-- 更改终止符为 //

DELIMITER //

-- 创建 user\_login 表的 INSERT 触发器

CREATE TRIGGER trg\_user\_login\_insert

AFTER INSERT ON user\_login

FOR EACH ROW

BEGIN

INSERT INTO user\_logs (u\_username, l\_type, l\_description)

VALUES (NEW.u\_username, 'INSERT', 'User created');

END;

//

-- 创建 user\_login 表的 UPDATE 触发器

CREATE TRIGGER trg\_user\_login\_update

AFTER UPDATE ON user\_login

FOR EACH ROW

BEGIN

INSERT INTO user\_logs (u\_username, l\_type, l\_description)

VALUES (NEW.u\_username, 'UPDATE', CONCAT('User updated. Fields: ', OLD.u\_username, ' -> ', NEW.u\_username, ', ', OLD.u\_password, ' -> ', NEW.u\_password));

END;

//

-- 创建 user\_login 表的 DELETE 触发器

CREATE TRIGGER trg\_user\_login\_delete

AFTER DELETE ON user\_login

FOR EACH ROW

BEGIN

INSERT INTO user\_logs (u\_username, l\_type, l\_description)

VALUES (OLD.u\_username, 'DELETE', 'User deleted');

END;

//

-- 创建 user\_email 表的 INSERT 触发器

CREATE TRIGGER trg\_user\_email\_insert

AFTER INSERT ON user\_email

FOR EACH ROW

BEGIN

INSERT INTO user\_logs (u\_username, l\_type, l\_description)

VALUES (NEW.u\_username, 'INSERT', 'Email added');

END;

//

-- 创建 user\_email 表的 UPDATE 触发器

CREATE TRIGGER trg\_user\_email\_update

AFTER UPDATE ON user\_email

FOR EACH ROW

BEGIN

INSERT INTO user\_logs (u\_username, l\_type, l\_description)

VALUES (NEW.u\_username, 'UPDATE', CONCAT('Email updated. Fields: ', OLD.u\_email, ' -> ', NEW.u\_email));

END;

//

-- 创建 user\_email 表的 DELETE 触发器

CREATE TRIGGER trg\_user\_email\_delete

AFTER DELETE ON user\_email

FOR EACH ROW

BEGIN

INSERT INTO user\_logs (u\_username, l\_type, l\_description)

VALUES (OLD.u\_username, 'DELETE', 'Email deleted');

END;

//

-- 恢复终止符为 ;

DELIMITER ;