Medium-like Blogging Platform: Database Design

This document describes a detailed relational database design for a blogging platform supporting users, posts, media uploads, followers, comments, likes, and premium subscriptions.

Core Tables

These tables form the core entities of the platform: Users, Posts, Categories, Post_Categories, Comments, Tags, Post_Tags.

Users

```
CREATE TABLE Users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    display_name VARCHAR(100),
    bio TEXT,
    profile_picture_url VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Posts

```
CREATE TABLE Posts (
    post_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    title VARCHAR(255) NOT NULL,
    slug VARCHAR(255) NOT NULL UNIQUE,
    content TEXT NOT NULL,
    excerpt TEXT,
    status ENUM('draft','published','archived') DEFAULT 'draft',
    featured_image_url VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```

Categories

```
CREATE TABLE Categories (
    category_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL UNIQUE,
    slug VARCHAR(100) NOT NULL UNIQUE,
    description TEXT
);
```

Additional Tables

To fulfill the complete requirements, the following tables are added:

Follows

```
CREATE TABLE Follows (
    follower_id INT NOT NULL,
    followed_id INT NOT NULL,
    followed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (follower_id, followed_id),
    FOREIGN KEY (follower_id) REFERENCES Users(user_id) ON DELETE CASCADE,
    FOREIGN KEY (followed_id) REFERENCES Users(user_id) ON DELETE CASCADE
);
```

Post_Likes

```
CREATE TABLE Post_Likes (
    like_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    post_id INT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE (user_id, post_id),
    FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE,
    FOREIGN KEY (post_id) REFERENCES Posts(post_id) ON DELETE CASCADE
);
```

Media

```
CREATE TABLE Media (
    media_id INT PRIMARY KEY AUTO_INCREMENT,
    post_id INT NOT NULL,
    user_id INT NOT NULL,
    media_type ENUM('image','video','other') NOT NULL,
    file_url VARCHAR(255) NOT NULL,
    caption TEXT,
    uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (post_id) REFERENCES Posts(post_id) ON DELETE CASCADE,
    FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE
);
```

Subscription_Plans

```
CREATE TABLE Subscription_Plans (
    plan_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    price DECIMAL(10,2) NOT NULL,
    billing_cycle ENUM('monthly','yearly') DEFAULT 'monthly',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

User_Subscriptions

```
CREATE TABLE User_Subscriptions (
    subscription_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    plan_id INT NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE,
    status ENUM('active','canceled','expired') DEFAULT 'active',
    FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE,
    FOREIGN KEY (plan_id) REFERENCES Subscription_Plans(plan_id) ON DELETE CASCADE
);
```

Transactions

```
CREATE TABLE Transactions (
    transaction_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    plan_id INT NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    payment_method VARCHAR(50),
    payment_status ENUM('pending','completed','failed') DEFAULT 'pending',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE,
    FOREIGN KEY (plan_id) REFERENCES Subscription_Plans(plan_id)
);
```

Relationships Summary

Users ↔ Posts (1:N) Users ↔ Comments (1:N) Users ↔ Likes (1:N) Users ↔ Follows (M:N self-referencing) Posts ↔ Categories (M:N) Posts ↔ Tags (M:N) Posts ↔ Media (1:N) Users ↔ Subscriptions (1:N) Subscriptions ↔ Transactions (1:N)