DB-Project-Part 1

ER diagram：(先用着，但建议换个新的)

图示, 示意图

AI 生成的内容可能不正确。

Relational schema (keys & constraints) with database system：

Primary Key, *Foreign Key*

Users: user\_id, username, email, password\_hash, created\_at;

Boards: board\_id, *user\_id(Users(user\_id))*, name, description, created\_at;

Pins: pin\_id, *user\_id(Users(user\_id))*, *board\_id(Boards(board\_id))*,

*original\_pin\_id(Pins(pin\_id))*, tags, source\_url, created\_at;

Pictures: *pin\_id(Pins(pin\_id))*, image\_blob, original\_url, uploaded\_url;

Friendships: friendship\_id, *requester\_id(Users(user\_id))*, *requested\_id(Users(user\_id))*, created\_at, updated\_at;

FollowStreams: stream\_id, *user\_id(Users(user\_id))*, name, created\_at;

FollowStreamsBoards: *stream\_id(FollowStreams(stream\_id)), board\_id(Boards(board\_id))*;

Like: like\_id, *user\_id(Users(user\_id)), pin\_id(Pins(pin\_id)),* created\_at;

Comments: comment\_id, *user\_id(Users(user\_id)), pin\_id(Pins(pin\_id))*, comment\_text, created\_at;

*-- Users Table*

CREATE TABLE Users (

user\_id SERIAL PRIMARY KEY,

username VARCHAR(50) UNIQUE NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

*-- Boards Table*

CREATE TABLE Boards (

board\_id SERIAL PRIMARY KEY,

user\_id INT NOT NULL,

name VARCHAR(100) NOT NULL,

description TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id) ON DELETE CASCADE

);

*-- Pins Table*

CREATE TABLE Pins (

pin\_id SERIAL PRIMARY KEY,

user\_id INT NOT NULL,

board\_id INT NOT NULL,

original\_pin\_id INT,

tags TEXT,

source\_url VARCHAR(255),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id) ON DELETE CASCADE,

FOREIGN KEY (board\_id) REFERENCES Boards(board\_id) ON DELETE CASCADE,

FOREIGN KEY (original\_pin\_id) REFERENCES Pins(pin\_id) ON DELETE CASCADE

);

*-- Pictures Table*

CREATE TABLE Pictures (

pin\_id INT PRIMARY KEY,

image\_blob BYTEA NOT NULL,

original\_url VARCHAR(255),

uploaded\_url VARCHAR(255),

FOREIGN KEY (pin\_id) REFERENCES Pins(pin\_id) ON DELETE CASCADE

);

*-- Friendships Table*

CREATE TABLE Friendships (

friendship\_id SERIAL PRIMARY KEY,

requester\_id INT NOT NULL,

requested\_id INT NOT NULL,

status VARCHAR(10) NOT NULL CHECK (status IN ('pending', 'accepted', 'declined')),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UNIQUE (requester\_id, requested\_id),

FOREIGN KEY (requester\_id) REFERENCES Users(user\_id) ON DELETE CASCADE,

FOREIGN KEY (requested\_id) REFERENCES Users(user\_id) ON DELETE CASCADE

);

*-- FollowStreams Table*

CREATE TABLE FollowStreams (

stream\_id SERIAL PRIMARY KEY,

user\_id INT NOT NULL,

name VARCHAR(100) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id) ON DELETE CASCADE

);

*-- FollowStreamBoards Table*

CREATE TABLE FollowStreamBoards (

stream\_id INT NOT NULL,

board\_id INT NOT NULL,

PRIMARY KEY (stream\_id, board\_id),

FOREIGN KEY (stream\_id) REFERENCES FollowStreams(stream\_id) ON DELETE CASCADE,

FOREIGN KEY (board\_id) REFERENCES Boards(board\_id) ON DELETE CASCADE

);

*-- Likes Table*

CREATE TABLE Likes (

like\_id SERIAL PRIMARY KEY,

user\_id INT NOT NULL,

pin\_id INT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UNIQUE (user\_id, pin\_id),

FOREIGN KEY (user\_id) REFERENCES Users(user\_id) ON DELETE CASCADE,

FOREIGN KEY (pin\_id) REFERENCES Pins(pin\_id) ON DELETE CASCADE

);

*-- Comments Table*

CREATE TABLE Comments (

comment\_id SERIAL PRIMARY KEY,

user\_id INT NOT NULL,

pin\_id INT NOT NULL,

comment\_text TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id) ON DELETE CASCADE,

FOREIGN KEY (pin\_id) REFERENCES Pins(pin\_id) ON DELETE CASCADE

);

Pins and Repins: Repins reference the original pin\_id. Deleting the original will delete all repins followed by it. Pin’s tags stored as a comma-separated string in Pins for simplicity, enabling keyword search via LIKE.

Image Storage: Each image is stored once per original pin. Repins reference the same image blob. Follow streams are private, but boards/pins are public.