Database

Relational, PostgresSQL database

Database reasoning

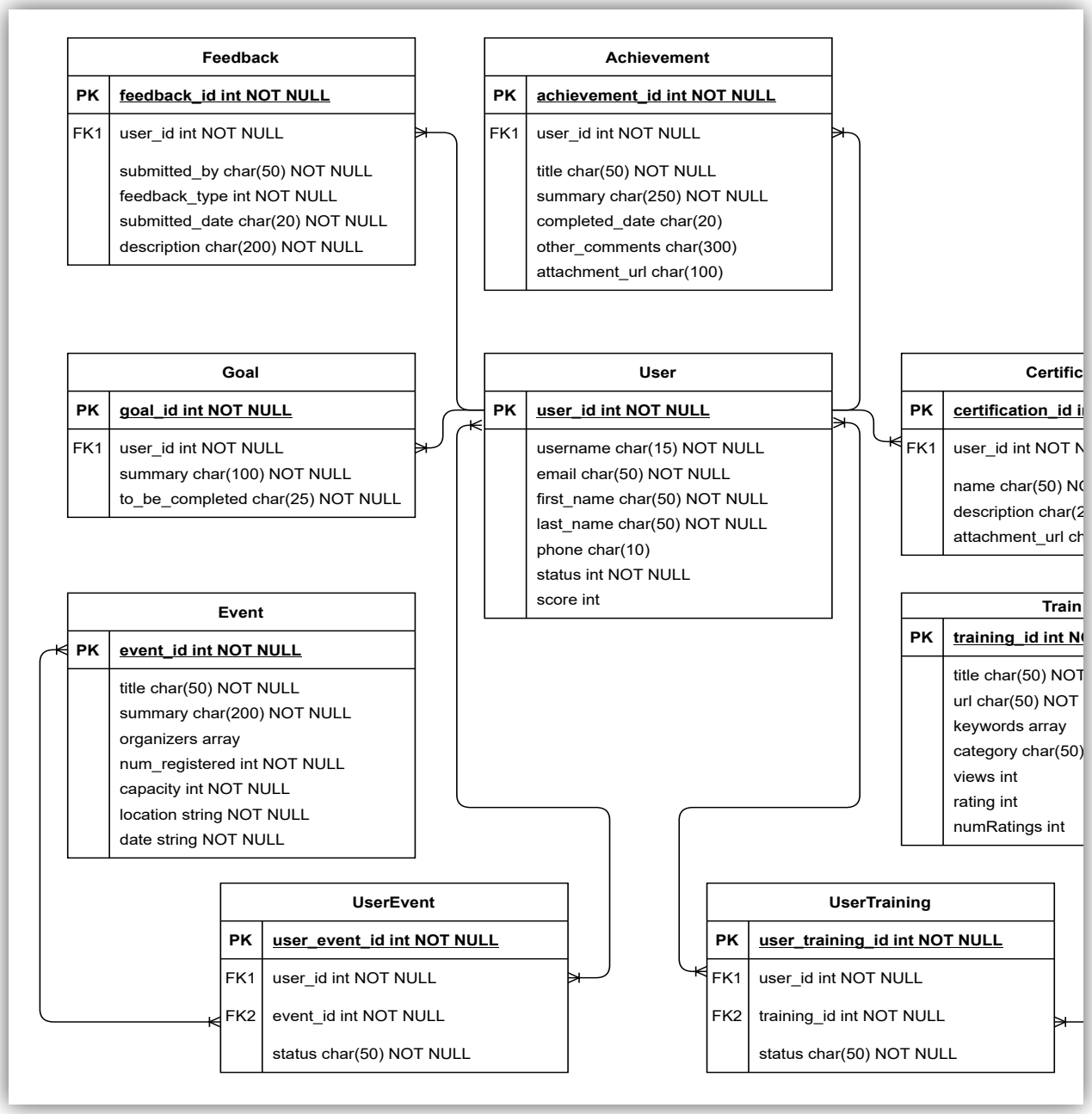
A relational database was a simple choice for the Employee Development Dashboard. The data is very structured with not much long-standing need to expand. All of the data is based off a single entity – a User object – and therefore is relational in nature. There are a few objects within the database that will need to be JSON structures, and since PostgresSQL has native support for this functionality, it was an easy choice for the database in this application.

Entities

- **User**
  - The main object of the application that holds metadata of a user. Every other entity is based off this entity as we are tracking information specifically for a user
- **Goal**
  - Has a one-to-many relationship with a User object. Users can register many goals, but a single goal cannot be on multiple users. Goals showcase something that a user registers as something they would like to accomplish in the future.
- **Achievement**
  - Has a one-to-many relationship with a User object. Users can register many achievements, but a single achievement cannot be on multiple users. Achievements mark something that the user has accomplished.
- **Feedback**
  - Has a one-to-many relationship with a User object. A feedback object can be only be associated with one user and a user can have many feedback items. Feedback objects are registered when another employee submits a specific feedback item for a single employee.
- **Certification**
  - Has a one-to-many relationship with a User object. Users can register many certifications, but a single certification cannot be on multiple users. Certifications are closely tied to achievements, but a little different in that they are specific accomplishments on a professional level, where achievements can be anything professionally.
- **Event**
  - This entity is registered by a user admin as something users can attend and get points for/attend (in later iteration). They will be showcased on their own page and can be attended by users
- **Training**

    - This entity showcases videos that users can watch and learn from. Many users can watch the same video.
- **UserEvent**
    - This table allow users to have a many-to-many relationship with events. Because users can attend many events and events can have many users, this table is needed to show this relationship.
- **UserTraining**
    - This table allow users to have a many-to-many relationship with training. Because users can watch many training videos and videos can have many users this table is needed to show this relationship.

Design


Feedback
PK
feedback_id int NOT NULL
FK1
user_id int NOT NULL
submitted_by char(50) NOT NULL
feedback_type int NOT NULL
submitted_date char(20) NOT NULL
description char(200) NOT NULL
Achievement
PK
achievement_id int NOT NULL
FK1
user_id int NOT NULL
title char(50) NOT NULL
summary char(250) NOT NULL
completed_date char(20)
other_comments char(300)
attachment_url char(100)
Goal
PK
goal_id int NOT NULL
FK1
user_id int NOT NULL
summary char(100) NOT NULL
to_be_completed char(25) NOT NULL
User
PK
user_id int NOT NULL
username char(15) NOT NULL
email char(50) NOT NULL
first_name char(50) NOT NULL
last_name char(50) NOT NULL
phone char(10)
status int NOT NULL
score int
Certific
PK
certification_id i
FK1
user_id int NOT N
name char(50) NO
description char(2
attachment_url ch
Event
PK
event_id int NOT NULL
title char(50) NOT NULL
summary char(200) NOT NULL
organizers array
num_registered int NOT NULL
capacity int NOT NULL
location string NOT NULL
date string NOT NULL
Train
PK
training_id int N
title char(50) NOT
url char(50) NOT
keywords array
category char(50)
views int
rating int
numRatings int
UserEvent
PK
user_event_id int NOT NULL
FK1
user_id int NOT NULL
FK2
event_id int NOT NULL
status char(50) NOT NULL
UserTraining
PK
user_training_id int NOT NULL
FK1
user_id int NOT NULL
FK2
training_id int NOT NULL
status char(50) NOT NULL