Tener Karar

11/03/2022

Database Design

# Overview

I will be using MySQL database in my web application. I chose relational database because I have relations between my tables, and it will help me use structure and identify my data.

## Data Specification

Because this web application might get bigger over the time, and I could have more relationships between the tables. I have seen MYSQL is the best choice for now because everything is related. I simplify my tables into 6 tables.

User

The user table entity contains all the information about all the users (Admin & Customers) we have in the database so we're going to store entries of different users into the users’ table.

Document Structure:

CREATE TABLE users  
(  
 user\_id int auto\_increment,  
 `firstname` varchar(30),  
 `lastname` varchar(30),  
 `phone` varchar(15),  
 `email` varchar(30),  
 `address` varchar(50),  
 `city` varchar(50),  
 `zip` int,  
 `state` varchar(20),  
 `username` varchar(50),  
 `password` text,  
 `role` varchar(20) default 'normal',  
 PRIMARY KEY (`user\_id`)  
);

Appointment:

This collection holds the appointments, so we might have different appointments and then we needed the time of the appointment, the date of it, and the note. If there any.

Document Structure:

CREATE TABLE `appointment`  
(  
 `appointment\_id` int auto\_increment,  
 `time` time,  
 `date` date,  
 `note` text,  
 `user\_id` int,  
 PRIMARY KEY (`appointment\_id`),  
 FOREIGN KEY (`user\_id`) REFERENCES users (`user\_id`)  
 on DELETE cascade  
 on UPDATE cascade  
);

Appoint\_ Service

The appoint\_service table which is like a bridge for both tables service entity and appointment entity so this one connects them because this is a many-to-many relationship. customer can have one appointment with two different services and vice versa, and customer can have this service which is needed by different appointments so it's a many-to-many relationship that's why we have a bridge table holds the appointment ID and service ID.

Document Structure:

CREATE TABLE `appoints\_service`  
(  
 `appointment\_id` int,  
 `service\_id` int,  
 PRIMARY KEY (`appointment\_id`, `service\_id`),  
 FOREIGN KEY (`appointment\_id`) REFERENCES `appointment` (`appointment\_id`)  
 on DELETE cascade  
 on UPDATE cascade,  
 FOREIGN KEY (`service\_id`) REFERENCES `service` (`service\_id`)  
 on DELETE cascade on UPDATE cascade  
);

Service

The service table or entity will allow the users to add the services to the specific appointment. The user will be choosing the services from a drop-down menu. For example, user may have an appointment like he needs sauna but at the same time maybe needs to get some massage or other service. so that's like two services in one appointment so those services are going to be choosing from the drop-down menu. The only person will be able to edit, delete, or add these services is the admin.

Document Structure:

CREATE TABLE `service`  
(  
 `service\_id` int auto\_increment,  
 `name` varchar(20),  
 `description` varchar(255),  
 `price` decimal(10, 2),  
 `points` int,  
 PRIMARY KEY (`service\_id`)  
);

Point

This collection will help the customer for counting their points in the system and keep track of it. point system is like every time a user book an appointment there is going to be a reward. Every service you choose there is a specific number of points attached to it. The rewards are going to be adding up to the point table which has point accumulated, this is where we accumulate customer points.

Document Structure:

CREATE TABLE `point`  
(  
 `point\_id` int auto\_increment,  
 `point\_accumulated` int,  
 `user\_id` int,  
 PRIMARY KEY (`point\_id`),  
 FOREIGN KEY (`user\_id`) REFERENCES `users` (`user\_id`)  
 on DELETE cascade  
 on UPDATE cascade  
);

Token\_records

Token records entity tracks all the refresh tokens from the devices you are using them. This token records is the one holding our refresh tokens so we can keep our users logged in for a long period of time like all the big tech companies do.

Document Structure:

create table token\_records  
(  
 token\_record\_id bigint auto\_increment primary key,  
 user\_id int not null,  
 refresh\_token text not null,  
 foreign key (user\_id) references users (user\_id)  
 on update cascade  
 on delete cascade  
);

Text

Description automatically generated with low confidence