Database Theory

Assignment 3

Author: Nitin Pasikanti &
Sundarakrishnan Ganesh
Supervisor: Ilir Jusufi
Semester: Autumn 2020
Course code: 2DV513

Table of Contents

Tasks

1 Idea

Everyone loves to have a little fun and as a student it is common to attend parties and events happening around the city, it could be family event or just socializing to let off a little steam. The project idea involves designing a web application that displays the users a list of all socializing events along with the name of the city it is taking place. The goal here is to solve the problem of organizing events and inviting people to new events.

To do this a database is required which will store the details of the events like the event name, event date, event location (city), audience. The application lets new new users to register and their login info is stored in the database. Registered users can login to the system while the login credentials are verified with the database. Another feature is, registered users can create new events which will be shown on the homepage of the application. Finally, after logging in users can respond to an event by selecting the "Going?" button under the "Status" column in the homepage.

After logging in when a user responds(Going) to an event, the user id is stored against the event id in the database and the number of people attending the event gets updated in the homepage under the "Attendees" column. This application is a good fit for users of all age groups and it has a user-friendly web interface.

1.1 Features

1. Users can view all the events on the homepage
2. New users can register
3. Users can login
4. User can create an event
   i) Enter event name
   ii) Enter event date
   iii) Enter audience type for the event
   iv) Enter event location
5. User can respond to an event after being logged in by clicking "Going?" button on the homepage
6. Users can view the number of people attending an event

1.2 Technical Environment

The technologies used for building the web application are as follows:

- MySQL (DBMS)
- Node.js (Programming language)
- Expressjs (Web application framework)

2 Logical Model

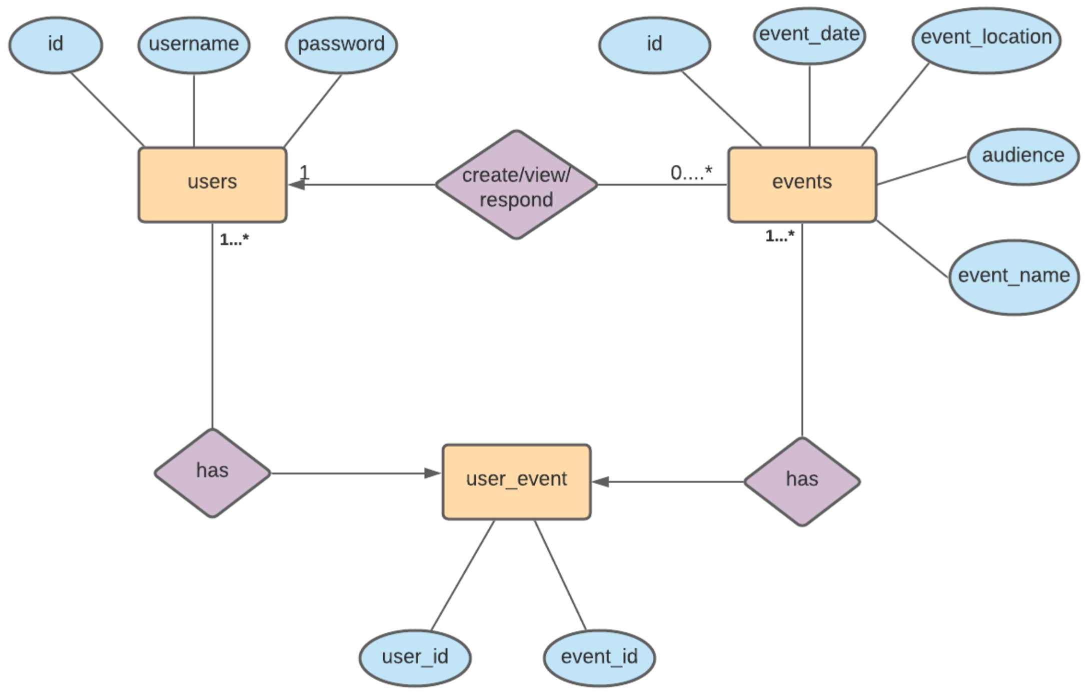

Figure 1

2.1 Entities and Attributes

The major entities of this database are ‘users’ and ‘events’ since the end product will be used by the users for booking and attending events. Therefore, we have created three tables: users, events and user_event.

The attributes of the ‘users’ table are username and password. The user would need to enter a username and password during system login and if it is a new user then these attributes would be initialized with values which would later be used for login. The attributes of the ‘events’ table are event_name, event_location, event_date, audience. These values get saved into their respective attributes when the user creates a new eventn from the web application. Both, the users and events table have an id attribute

The attributes of ‘user_event’ table are user_id obtained from id attribute in ‘users’ table and event_id obtained from id attribute form ‘events’ table. We use these to crossmatch with the user attending a particular event as it creates a link between link the events and every user attending an event. When the user clicks the “Going?” button beside an event on the homepage then the user’s id is associated with that event’s id.

2.2 Relations

The relationship between ‘users’ and ‘events’ entity is ‘create/view/respond’, it has a one to many relation because a single user can create multiple events, view a list of all events and respond to an event if he is interested.

The relation between ‘user_event’ and ‘users’ is one-many and one user has one user_event but each user_event consists of various users because one user can create only one user_ event but one user_event consists of all users that are attending the events.

Similarly, an event has one user_event but each user_event consists of various events because user_events consists of all the events that a user is attending.

3 Design in SQL

The design is translated to collections in SQL and data is shown below:

We translated entities and relationships by naming the tables according to entity name and the colums according to their respective attributes. We obtained 3 tables after normalization and the primary keys are shown as underlined. Refer the table 1 shown below

Database schema : fikatime	
Entity (table name)	Attributes (columns)
1. events	events(id, event_name, event_date, event_location, audience)
2. user_event	user_event(user_id, event_id)
3. users	users(id, username, password)

Table 1

The schema of each of the tables are presented below in figure 2, figure 3 and figure 4

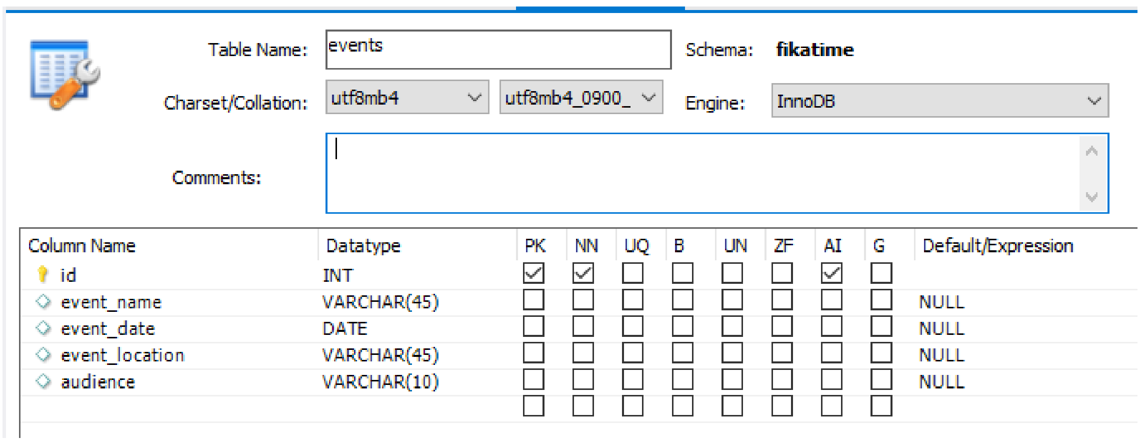

Table Name: events

Schema: **fikatime**

Charset/Collation: utf8mb4 | utf8mb4_0900_

Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	✓	✓					✓		
event_name	VARCHAR(45)									NULL
event_date	DATE									NULL
event_location	VARCHAR(45)									NULL
audience	VARCHAR(10)									NULL

Figure 2

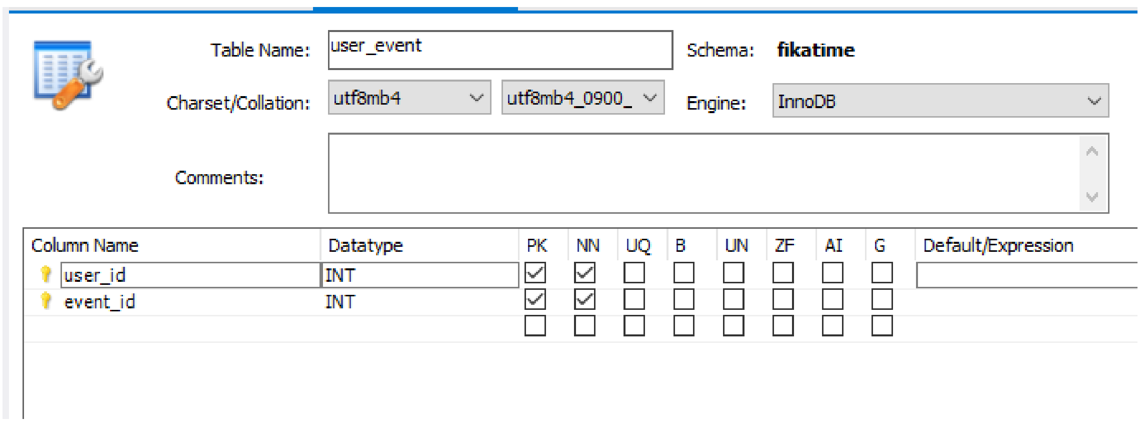

Table Name: user_event

Schema: **fikatime**

Charset/Collation: utf8mb4 | utf8mb4_0900_

Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
user_id	INT	✓	✓							
event_id	INT	✓	✓							

Figure 3

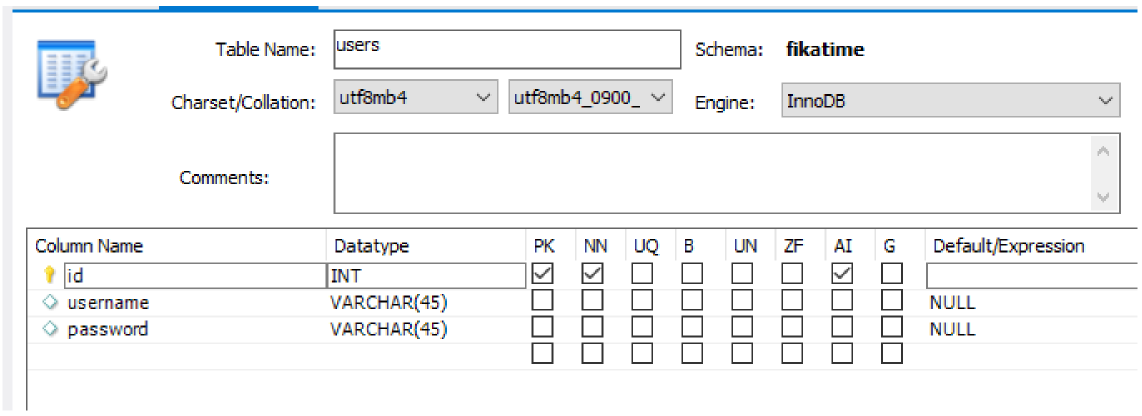

Table Name: users

Schema: **fikatime**

Charset/Collation: utf8mb4 | utf8mb4_0900_

Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	✓	✓					✓		
username	VARCHAR(45)									NULL
password	VARCHAR(45)									NULL

Figure 4

Creating the ‘user_event’ table avoids redundancy as we do not repeat the data inside a single table and therefore, this model fulfils the normalized form. This table has one to many relation with both the ‘users’ and ‘events’ table. The attributes in this table point to the unique id’s of the ‘users’ and ‘events’ tables respectively.

4 SQL Queries

The important queries implemented in the application are discussed below:

1. The first multirelational query was employed in implementing the attendance preference of the event.
   i) Firstlt, the id of a user is obtained first,
   query: SELECT id FROM user WHERE username='currentLoggedInUser';
   ii) The respective event id is assigned as values to each attendance button. Thus, when the user clicks the button, the event id is displayed. The event id is acquired from querying the events table in the database,
   query: "SELECT * FROM events".
   iii) Thus a new table is created (if it weren't before) and the respective user_id and event_id are inserted into it,
   query: INSERT INTO user_event (user_id, event_id) VALUES ('user_id_value', 'event_id_value')

2. This query is used to find the history of all the users attending all the events and simultaneously shows the users that are not attending any events.
   Query: SELECT * FROM user_event LEFT JOIN events ON user_event.event_id = events.id;

3. This query saves the values of the event created by the users into their respective event attributes.
   Query: INSERT INTO events (event_name, event_date,event_location, audience) VALUES ('event_name_value', 'event_date_value', 'event_location_value', 'audience.value')

4. Then the View 'v1' was implemented to join the user id and event id.
   query: CREATE VIEW v1 FROM SELECT * FROM events LEFT JOIN user_event ON events.id = user_event.user_id;

5. Two other views 'v2' and 'home' were created.

6. The 'v2' view is used to count the number of people who clicked the "Going" button. This is incremented and updated in this view,
   query: CREATE VIEW v2 SELECT COUNT(event_id) AS `attendees` FROM v1 GROUP BY event_id"

7. The 'home' view is a multirelation as it renders the important information to be presented on the homepage from the other two views.
   Query: CREATE VIEW home SELECT * FROM events LEFT OUTER JOIN events.id = v2.eid;

We used the views to render the important information on the homepage of the web application as we felt it would be the most efficient way to fetch data as well as ensure resuse and maintainability in the future.

5 Implementation

The source code consists of a app.js file which implements the server side functionality of the web application. The client side user interface is provided by the pug files inside the views directory. The routes directory contains javascript files which implement the functionality of each of features of the application i.e login, register, attendanceStatus, event, homepage.

The database.js file contains MySQL credentials the host, user, database which is used to establish connection with MySQL and renders data onto the homepage of the web application. These credentials need to be matched with the schema name of the database in the MySQL server. Before running the application, the dump file must be imported into MySQL Server.

Traversing through the application:

The application currently has the following routes.
  * / (which is the homepage where all the events are listed.)
  * /login (which is the login page with a quite simple authentication mechanic.)
  * /register (which is the register page where you can register a user into the application.)
  * /createevent (which is the page where you can create events to be listed in the homepage.)

The backup.sql dump file is attached in the submission.

6 Supplemental Video

A presentation link to the web application is given below:

https://youtu.be/Pd2lQHGNa-Q