

Hérault Events

Chakib ELHOUITI : 21813619

Massili KEZZOUL : 21815514

Groupe P



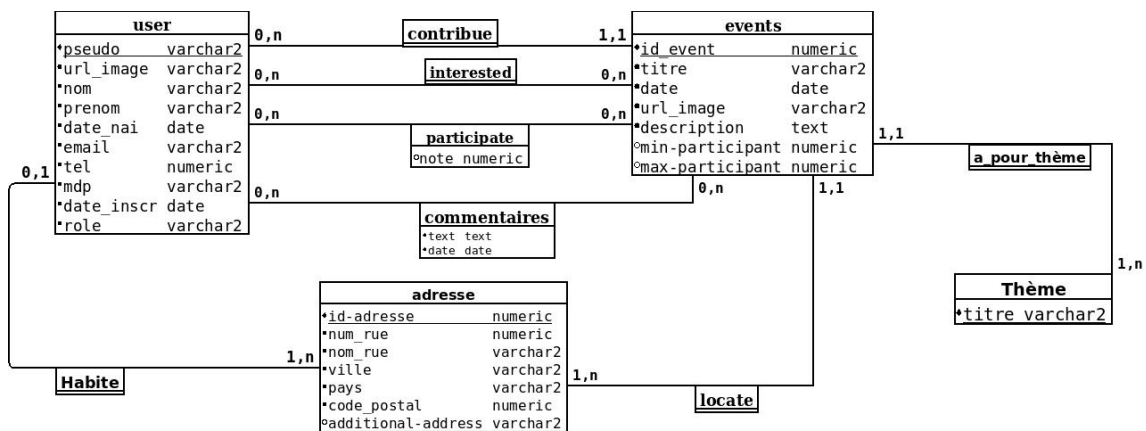
1 Introduction

Dans le cadre d'un projet commun entre deux UE de la faculté des sciences de l'université de Montpellier, nous avons réalisé une base de données associée à une application web permettant la publication d'événements culturels ou sportifs dans un département donné. Nous avons choisi le département de l'Hérault. L'application peut d'ailleurs être visitée à cette adresse 'http ://webpeda.etu.umontpellier.fr/e20180011096' à condition d'être sur le réseau de l'université de Montpellier.

2 Modél entité-association

2.1 Schéma E/A

Tout d'abord nous avons modélisé notre base de données de la manière suivante :



Modélisation version 1.2/ 17 décembre 2019

Schéma 1 – Modél E/A

2.2 Explication du Schéma

Les tables

Adresse : Cette table permet de stocker l'ensemble des adresses qui seront utilisés par la table *User* et *Events*. On a choisi de ne pas stocker les coordonnées GPS des adresses car ils peuvent être calculés (par une API par exemple). On a décidé de le faire de cette manière aussi parcequ'il est préférable de ne pas demander des coordonnées GPS à un utilisateur lambda de l'application web.

User : La table *User* stock les informations personnelles des utilisateurs du site web. Un utilisateur est identifié par un pseudo qu'il renseigne à son inscription. Ce dernier doit bien évidemment être unique.

Il existe trois types d'utilisateur :

- Visiteur ('visitor')
- Contributeur ('contributor')
- Administrateur ('admin')

Le type de chaque utilisateur est stocké dans l'attribut *role_user* qui est une énumération (de type ENUM). L'attribut *email*, et *tel* doivent être unique.

Events : Cette table se charge de stocker les informations relatifs à un événement. Un événement est défini par un identificateur numérique qui générer automatiquement à l'insertion d'un tuple (par AUTO_INCREMENT) et contient forcément un titre et une date (attribut de type DATETIME). Il peut aussi contenir un lien vers une image (local ou distante), une description (de type TEXT) et un nombre minimum et maximum de participant.

Theme : Chaque événement est organisé autour d'un thème donné, donc on a aussi modélisé une table thème afin de stocker tout les thèmes.

Les association

Participte : cette association permet de stocker les participation des utilisateurs à des événements. C'est à dire un utilisateur peut participer à plusieurs événements comme ne pas participer à aucun événement et un événement peut avoir aucun ou plusieurs participants. C'est pareil pour les autres associations commentaires et interested

contribuer : permet de stocker un utilisateur qui sera un contributeur dans un ou plusieurs événements. C'est pareil pour les autres associations père/fils.

2.3 Modèle logique de données

Ensuite nous avons réaliser le modèle relationnel :

adresse (id_adresse, num_rue, nom_rue, ville, pays, code_postal, additional_adresse)
user(pseudo, nom, prenom, date_nai, email, tel, mdp, date_inscr, role, *id_adresse*)
events(id_event, titre, date, url_image, description, min-participant, max-participant, *pseudo_contributeur*, *id_adresse*, *theme*)
thème(titre)
particiapte(*pseudo*, *id_event*, note)
interested(*pseudo*, *id_event*)
commentaires(*pseudo*, *id_event*, text, date).

2.4 Traduction du modèle relationnel en Langage sql (Définition des données)

Enfin on a écrit le script sql permettant de créer les tables du modèle relationnel (MYSQL).

3 Les procédures

throw_err : C'est une procédure qui jette une erreur et affiche un message donné en paramètres, elle est utilisé dans les triggers et les procédures pour les arrêtees avec une erreur.

events_users_par_annee C'est une Procédure, elle prend en paramètres une année et une chaîne de caractères(events ou utilisateurs),si le deuxième paramètres est events, la procédure affiche les événements de l'année donnée en paramètres, si non elle affiche les utilisateurs inscrit pendant l'année donnée en arguments.

events_par_commentaires : C'est une Procédure, elle prend en paramètres une chaîne de caractères (asc ou desc pour l'ordre), elle ordonne les événements par nombre de commentaires soit par ordre croissant ou décroissant selon la chaîne donnée en paramètres.

4 Les fonctions

note_event : C'est une fonction, elle prend en paramètres un id_event(un événement) et calcul sa note moyenne.

nb_participe : C'est une fonction, elle prend en paramètres un id_event(un événement) et calcul le nombre de participants à cet événement.

nb_interesses : C'est une fonction, elle prend en paramètres un id_event(un événement) et calcul le nombre d'intéressés à cet événement.

classement_event : C'est une fonction, elle prend en paramètres un id_event(un événement) et retourne le classement de l'événements par note moyenne, c'est à dire son classement parmi tous les événements, s'il y'a des événements de même note, la fonction retourne son classement par note et par son id(du plus petit au plus grand).

5 Les triggers

Sur toutes les fonctionnalités demandées, nous les avons toutes implementées sauf :

- La visualisation de tout les événements en mode cartographique, néanmoins nous avons pu implementées dans la page d'un seul événement donné, sa position dans une carte (avec OpenLayers).

Par contre, nous avons implementé la possibilité pour un utilisateur, une fois inscrit, de modifier ses informations personnels, ajouter une photo pour son profil et aussi la possibilité de supprimer son compte.

On a aussi ajouté la possibilité pour un utilisateur de s'intéressé à un événement avant d'y participer.

6 Conclusion

6.1 Les problèmes rencontrés

On a rencontré quelques problèmes concernant la syntaxe mysql, qui était un peu difficile pour créer des triggers et des procédures, aussi

6.2 Les compétences acquises

Pour conclure, à l'issue de ce projet nous avons réussi à réaliser un site web fonctionnel et prêt à l'utilisation.

Ce projet nous aura permis d'approfondir nos connaissances en développement web et de compléter nos acquis sur les outils de base du web, tel que :HTML, CSS, PHP et JAVASCRIPT.