Université de Montpellier
L3 informatique

Bases de données

Rapport de projet S5
Projet Base de données — HLIN511

Étudiants :
Sylvain Courroye
Denis Beauget (21608519)

Année : 2019 – 2020
Groupe : A – CMI

Chapitre 1

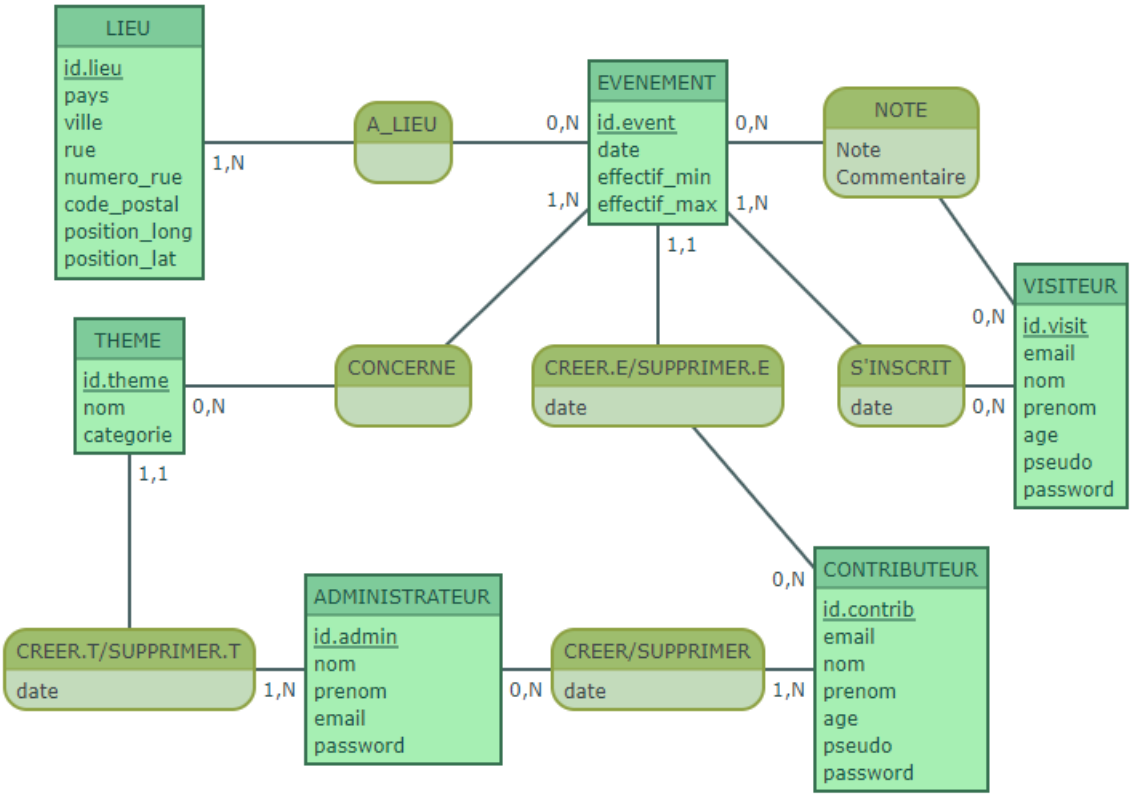
Explication du projet

Dans le cadre de l'UE HLIN511 Systèmes d'informations et base de données 2 nous devions créer et mettre en place une base de données permettant la publication d'évenements culturels ou sportifs.

Le but pour nous était de comprendre les différents outils du monde de la base de données, les différents logiciel et langages mais aussi mettre en pratique les éléments appris en cours en respectant un certain nombre de contraintes.

Chapitre 2

Conception du modèle entité-association



[1]

2.1 Choix du logiciel et réflexion

La première étape du projet était de comprendre la problématique et les points clés du projet pour faire les bons choix au niveau du modèle. Le modèle entité-association étant le point de départ de toute la base de donnés il fallait limiter les erreurs pour éviter des changements qui engendrerait une cascade de modification dans le futur.

Nous avons décider de choisir l'outil libre et gratuit Mocodo online. Celui-ci permet de créer des modèles entité-association en ligne et sans la contrainte graphique (il se charge lui même d'organiser les différents éléments du shéma)

2.2 Choix des entités

Nous avons commencé par identifiés les différents entités du projet, les visiteurs, les administrateurs et les contributeurs. Notre première idée était de créer une entité personne qui aurait possédé un attribut de rôle permettant d'identifier son rôle et de mettre en place une hiérarchie d'héritage. Malheureusement, nous nous sommes rendu compte que les rôles étaients sensiblement différents et nous aurions eu du mal à différencier les différentes personnes alors nous avons choisis une solution plus simple, une entité par personne, visiteur, contributeur et administrateur permettant un modèle plus visuel et plus simple avec des rôles bien distinct. Nous avons également créer une entité thème, elle permet d'avoir un suivit de tout les thèmes et de les classés par catégorie.

Une entité événement, élément central de la base de donnée contenant les informations nécéssaire (effectif,date,id) et finalement une entité lieu que nous avons décider de mettre à part de l'entité événément. Cela nous permet d'avoir plus d'information sur un lieu (pays,rue,position...) et surtout de pouvoir associer un lieu à plusieurs évenements tout en gardant une trace dans une table annexe (A LIEU) de chaque association lieu-evenement.

2.3 Choix des associations

La seconde problèmatique principal était de pouvoir faire apparaitre dans le modèle les fonctionallités importantes et essentiel au bon fonctionnement de la base de données tout en restant assez compréhensible pour pouvoir être intégrer dans un site web dans le futur (Projet HLIN510)

Nous avons décider dans un premier temps de mettre en avant toute les relations de création/suppression et leur suivi à travers un attribut date. Ainsi on comprend aisément qu'un administrateur

1. Mocodo : `http://mocodo.wingi.net/`

peut créer ou supprimer des contributeurs ou des thèmes ou qu'un contributeur peut créer ou supprimer des événements. Cela permet d'identifier visuellement les rôles de chacun sans surcharger d'information le modèle.

Il nous fallait ensuite des relations de "suivi" permettant de relier et de comprendre comment les entités fonctionnent entre elles. Par exemple, savoir quel thème était associé à quel événement ou quel visiteur s'était inscrit à un événement ou encore comme expliquer dans la section précédente, savoir quel lieu était associé à quel événement.

Et enfin l'association "note" elle permet d'avoir un suivi des notes des visiteurs suivant les évenements auquels ils ont participés. Nous avons préféré une association plutôt qu'une entité pour rester dans notre choix "d'association de suivi" plutôt qu'une entité qui représente des éléments concrets du modèle.

Chapitre 3

Trigger et fonction

3.1 Trigger

```
1  /*Trigger de vérification pour la suppression d'un thème*/
2
3  DROP TRIGGER IF EXISTS ATTENTION_THEME
4  DELIMITER $$
5  DROP TRIGGER IF EXISTS ATTENTION_THEME
6  DELIMITER |
7  CREATE TRIGGER ATTENTION_THEME
8  BEFORE DELETE ON THEME
9  FOR EACH ROW
10 BEGIN
11     DECLARE
12     nb_events NUMERIC(3);
13     SELECT COUNT(*) INTO nb_events
14     FROM THEME,EVENEMENT
15     WHERE THEME.idtheme = EVENEMENT.idtheme
16     GROUP BY THEME.idtheme;
17     IF nb_events <> 0 THEN
18         signal sqlstate '45000' set message_text =
19         'Impossible de supprimer ce theme,
20         il est utilisé dans un ou plusieurs évenements existant';
21     END IF;
22     END; |
23 DELIMITER ;
```

Ce premier trigger concerne les tables theme et evenement et répond à une problématique créer par nos choix de conception. Nous avions décidé de séparer la table theme et evenement pour pouvoir trier les theme par catégorie et également attribuer plusieurs événements à un même theme.

Mais un problème venait naturellement, si un administrateur voulait supprimer un thème d'une catégorie particulière celui-ci pouvait, le cas échéant, être déjà choisit par un contributeur pour l'un de ses événements et donc la suppression de ce thème entrainerai un effet de bord sur la table des évenements. Ce trigger permet de vérifier lors d'une suppression d'un thème si celui-ci n'est pas déjà utiliser dans un évenement de la table évenement et ainsi empêcher la suppression du thème correspondant.

```
/* Trigger de suivi dans la table A_LIEU en cas d'insert d'un évenement */

DROP TRIGGER IF EXISTS SUIVI_THEME;
DELIMITER |
CREATE TRIGGER SUIVI_THEME
AFTER INSERT ON EVENEMENT
FOR EACH ROW
BEGIN
INSERT INTO CONCERNE VALUES(NEW.idevent,NEW.idtheme);
END; |
DELIMITER ;



/*Trigger de suivi dans la table CONCERNE en cas d'insert d'un évenement */

DROP TRIGGER IF EXISTS SUIVI_LIEU;
DELIMITER |
CREATE TRIGGER SUIVI_LIEU
AFTER INSERT ON EVENEMENT
FOR EACH ROW
BEGIN
INSERT INTO A_LIEU VALUES(NEW.idlieu,NEW.idevent);
END; |
DELIMITER ;
```

Ces 2 triggers sont assez simpliste et on comme unique fonction de mettre à jour la base de données en cas de nouvelle insertion dans la table évenement. Ils permmettent une "automatisation" de la base de données, l'ajout d'un évenement étant un élement central dans notre modèle celui-ci doit déclencher une mise à jours de certaine table, ici en cas d'ajout d'un nouvel évenement les tables de suivi CONCERNE associant un evenement à un theme et A_LIEU associant un lieu a un evenement sont mises à jours pour prendre en compte de nouveau évenement.

3.2 Procédure et fonction

```
/*Procédure de comptage d'apparition d'un theme*/

DROP PROCEDURE IF EXISTS NOMBRE_UTILISATION_THEME;
DELIMITER |
CREATE PROCEDURE NOMBRE_UTILISATION_THEME(OUT NombreTotal INT,
IN idtheme INTEGER)
BEGIN
SELECT COUNT(*) INTO NombreTotal
FROM EVENEMENT WHERE EVENEMENT.idtheme = idtheme
GROUP BY EVENEMENT.idtheme;
END; |
DELIMITER ;
```

Cette première procédure est une "procédure administrateur" elle permet à l'administrateur de compter le nombre de fois qu'un theme apparait (est utilisé) dans la table évenement, ainsi on peut se rendre compte de l'utilisation d'un thème et si il serait plus judicieux de le supprimer si il n'est pas utlisé ou de modifier son intitulé si il est trop ciblé, par exemple le thème "Photo noir et blanc orienté paysage 98°" est probablement un thème peu sélectionné et donc cette procédure permet de s'en rendre compte plus facilement.

Chapitre 4

Les Tests

Pour illustrer le fonctionnement de notre base de données nous avons effectuer une série d'insertion dans la base pour simuler le fonctionnement de celle-ci.

On commence par vider les informations de la base en respectant l'ordre de suppression mis en place à travers les foreign key :

```
1  DELETE FROM A_LIEU;
2  DELETE FROM CONCERNE;
3  DELETE FROM INSCRIT;
4  DELETE FROM CREER_SUPPRIMER_CONTRIB;
5  DELETE FROM EVENEMENT;
6  DELETE FROM LIEU;
7  DELETE FROM THEME;
8  DELETE FROM CONTRIBUTEUR;
9  DELETE FROM ADMINISTRATEUR;
10 DELETE FROM VISITEUR;
```

Ensuite, toujours en respectant l'ordre des foreign key et leurs contraintes on ajoute les éléments nécéssaire à la création d'un événement.

```
INSERT INTO ADMINISTRATEUR VALUES('1','Cardano',
'Vincent','vincent.cardano@gmail.com','cardano34');

INSERT INTO VISITEUR VALUES('1',
'michel.dumas@laposte.net','Dumas','Michel','68','Michou','cuisinella');
INSERT INTO CONTRIBUTEUR VALUES('1','1',
'michel.dumas@laposte.net','Dumas','Michel','68','Michou','cuisinella');
INSERT INTO CREER_SUPPRIMER_CONTRIB VALUES('1','1',now());
INSERT INTO CONTRIBUTEUR VALUES('2','1',
'roger.delcours@wanadoo.fr','Delcours','Roger','43','Roro','delro21');
INSERT INTO CREER_SUPPRIMER_CONTRIB VALUES('2','1',now());


INSERT INTO LIEU VALUES('1','France','Montpellier','Gambetta','5',
34000,234,-213);
INSERT INTO LIEU VALUES('2','France','Paris','Place Montmartre','21',
78000,454,228);
INSERT INTO LIEU VALUES('3','France','Grenoble',' Rue Millet','11',
38000,800,134);
INSERT INTO LIEU VALUES('4','France','Toulouse','Rue Clément','19',
31000,234,900);

INSERT INTO THEME VALUES('1','Shooting photo','Photo','1',now());
INSERT INTO THEME VALUES('2','Exposition peinture','Art','1',now());
INSERT INTO THEME VALUES('3','Maraude','Associatif','1',now());
```

Ces différentes insertions nous permettent maintenant de créer des évenements. Ces insertions d'évenement vont déclencher nos triggers de suivie et inscrire dans les tables correspondantes les informations de ce nouvel évenement.

```
INSERT INTO EVENEMENT VALUES('1','2019-12-21',400,900,'1','1','2',now());
INSERT INTO EVENEMENT VALUES('2','2019-12-28',2,40,'1','1','3',now());
INSERT INTO EVENEMENT VALUES('3','2019-12-24',0,20,'1','3','4',now());
```

Pour comparer, les triggers permettent d'éviter de devoir rajouter des insertions de suivi "manuelle" et peu explicite comme cette ligne par exemple :

```
INSERT INTO INSCRIT VALUES('1','1',now());
```

Et finalement nous avons illustrer l'utilisation de la procédure du nombre d'utilisation d'un thème à travers un appel de celle-ci sur une variable : En suivant les différentes insertions, le résultat est 2, cet à dire que le theme avec l'id 1 (Correspondant à 'Shooting photo') est utilisé dans 2 évenements distinct.

```
1 SET @nb=0;
2 CALL NOMBRE_UTILISATION_THEME(@nb,'1');
3 SELECT @nb as NombreTotal;
```