

UNIVERSITÉ DE MONTPELLIER
L3 INFORMATIQUE

Bases de données

RAPPORT DE PROJET S5
PROJET BASE DE DONNÉES — HLIN511

Étudiants :

Sylvain COURROYE (20120447)
Denis BEAUGET (21608519)

Année : 2019 – 2020

Groupe : A – CMI

Chapitre 1

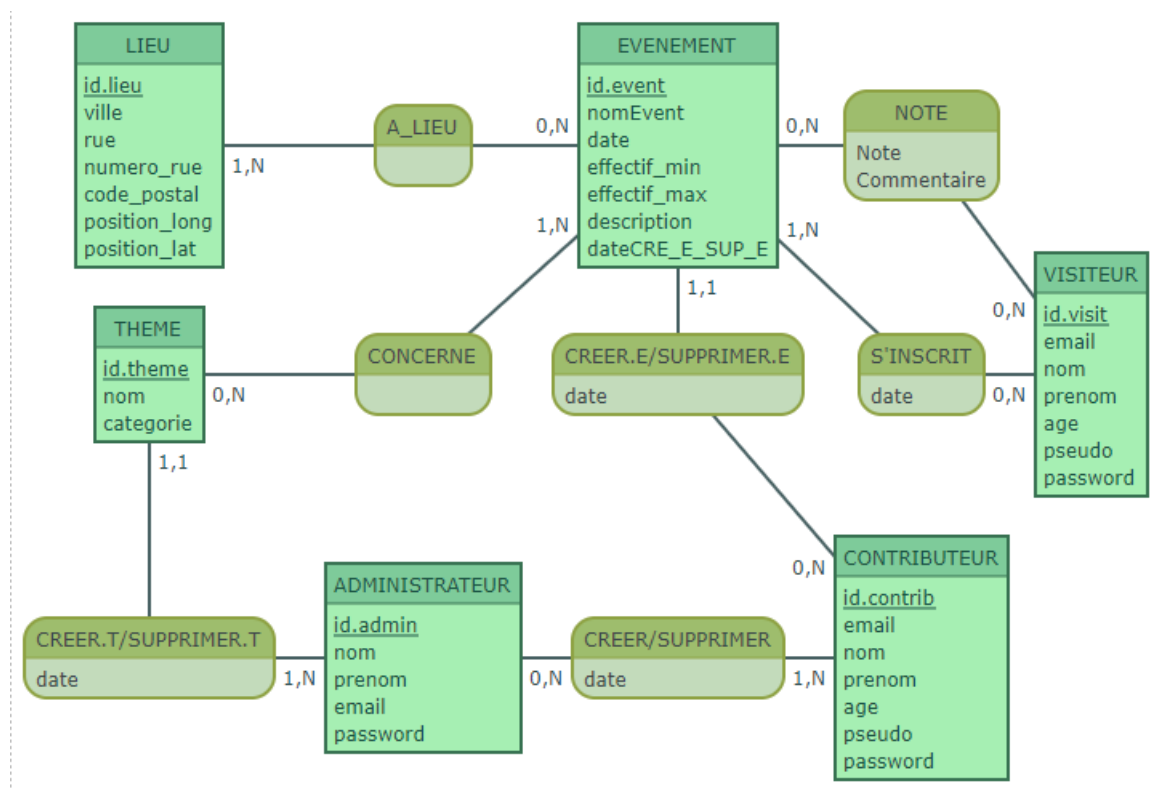
Explication du projet

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

Le but pour nous était de comprendre les différents outils du monde de la base de données, les différents logiciels 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



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 données il fallait limiter les erreurs pour éviter des changements qui engendreraient une cascade de modification dans le futur.

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

2.2 Choix des entités

Nous avons commencé par identifier les différentes 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 étaient sensiblement différents et nous aurions eu du mal à différencier les différentes personnes alors nous avons choisi une solution plus simple, une entité par personne, visiteur, contributeur et administrateur permettant un modèle plus visuel et plus simple avec des rôles bien distincts. Nous avons également créé une entité thème, elle permet d'avoir un suivi de tous les thèmes et de les classer par catégories.

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

2.3 Choix des associations

La seconde problématique principale était de pouvoir faire apparaître dans le modèle les fonctionnalités importantes et essentielles au bon fonctionnement de la base de données tout en restant assez compréhensible pour pouvoir être intégrée dans un site web dans le futur (Projet HLIN510)

Nous avons décidé dans un premier temps de mettre en avant toutes 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 événements auxquels ils ont participé. 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  CREATE TRIGGER ATTENTION_THEME
6  BEFORE DELETE ON THEME
7  FOR EACH ROW
8  BEGIN
9      DECLARE nb_events NUMERIC(3);
10     DECLARE idtheme INT;
11     SET idtheme = OLD.idtheme;
12     SELECT COUNT(*) INTO nb_events
13     FROM EVENEMENT
14     WHERE EVENEMENT.idtheme = idtheme;
15     IF nb_events <> 0 THEN
16         signal sqlstate '45000' set message_text =
17         'Impossible de supprimer ce theme,
18         il est utilisé dans un ou plusieurs événements existant';
19     END IF;
20     END; |
21 DELIMITER ;
```

Ce premier trigger concerne les tables theme et evenement et répond à une problématique créer par nos choix de conception. Nous avons 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 choisi 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 événements. Ce trigger permet de vérifier lors d'une suppression d'un thème si celui-ci n'est pas déjà utilisé dans un événement de la table événement et ainsi empêcher la suppression du thème correspondant.

```
1  /* Trigger de suivi dans la table A_LIEU en cas d'insert d'un événement */
2
3  DROP TRIGGER IF EXISTS SUIVI_THEME;
4  DELIMITER |
5  CREATE TRIGGER SUIVI_THEME
6  AFTER INSERT ON EVENEMENT
7  FOR EACH ROW
8  BEGIN
9  INSERT INTO CONCERNE VALUES(NEW.idevent,NEW.idtheme);
10 END; |
11 DELIMITER ;
12
13
14
15 /*Trigger de suivi dans la table CONCERNE en cas d'insert d'un événement */
16
17 DROP TRIGGER IF EXISTS SUIVI_LIEU;
18 DELIMITER |
19 CREATE TRIGGER SUIVI_LIEU
20 AFTER INSERT ON EVENEMENT
21 FOR EACH ROW
22 BEGIN
23 INSERT INTO A_LIEU VALUES(NEW.idlieu,NEW.idevent);
24 END; |
25 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 événement. Ils permettent une "automatisation" de la base de données, l'ajout d'un événement étant un élément central dans notre modèle celui-ci doit déclencher une mise à jours de certaine table, ici en cas d'ajout d'un nouvel événement 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 événement.

3.2 Procédure et fonction

```
1  /*Procédure de comptage d'apparition d'un theme*/
2
3  DROP PROCEDURE IF EXISTS NOMBRE_UTILISATION_THEME;
4  DELIMITER |
5  CREATE PROCEDURE NOMBRE_UTILISATION_THEME(OUT NombreTotal INT,
6  IN idtheme INTEGER)
7  BEGIN
8  SELECT COUNT(*) INTO NombreTotal
9  FROM EVENEMENT WHERE EVENEMENT.idtheme = idtheme
10 GROUP BY EVENEMENT.idtheme;
11 END; |
12 DELIMITER ;
13
14 /* Procédure de moyenne d'effectif pour les événements */
15
16 DROP PROCEDURE IF EXISTS MOYENNE_EFFECTIF_EVENT;
17 DELIMITER |
18 CREATE PROCEDURE MOYENNE_EFFECTIF_EVENT(OUT MoyenneEffectif INT)
19 BEGIN
20     DECLARE SumEffectifMax NUMERIC(10);
21     DECLARE SumEffectifMin NUMERIC(10);
22     DECLARE Nb_events NUMERIC(3);
23
24     SELECT SUM(effectif_min) INTO SumEffectifMin FROM EVENEMENT;
25     SELECT SUM(effectif_max) INTO SumEffectifMax FROM EVENEMENT;
26     SELECT COUNT(*) INTO Nb_events FROM EVENEMENT;
27     SET MoyenneEffectif = (SumEffectifMin + SumEffectifMax) / Nb_events;
28
29     END; |
30 DELIMITER ;
31
32
33 /*Procédure de comptage du lieu le plus utilisé */
34
35 DROP PROCEDURE IF EXISTS MAX_APPARITION_VILLE;
36 DELIMITER |
37 CREATE PROCEDURE MAX_APPARITION_Ville(OUT NomVille VARCHAR(50),
38 OUT MaxApparitionVille VARCHAR(50))
39 BEGIN
40
41     SELECT ville,COUNT(*) AS NbUtilisation
42     INTO NomVille,MaxApparitionVille FROM LIEU GROUP BY ville
43     HAVING COUNT(*) >= ALL(SELECT COUNT(*) FROM LIEU GROUP BY ville);
```



```
44  
45     END ; |  
46     DELIMITER ;
```

La première procédure est une "procédure administrateur" elle permet à l'administrateur de compter le nombre de fois qu'un thème apparaît (est utilisé) dans la table événement, 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 utilisé 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. La deuxième procédure permet de calculer la moyenne des effectifs min/max sélectionner par les contributeurs, cela permet de se rendre compte du nombre moyen de personnes attendues aux événements et de, par exemple, comparer cela au nombre de visiteurs du site. Enfin, la dernière procédure permet de voir quel ville apparaît le plus dans la liste des événements, cela permet de se rendre compte de la ville la plus "active" ou il y a le plus d'événement créer.

Chapitre 4

Les Tests

Pour illustrer le fonctionnement de notre base de données nous avons effectué une série d'insertions 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 nécessaires à la création d'un événement.

```

1  INSERT INTO ADMINISTRATEUR VALUES('1','Cardano',
2  'Vincent','vincent.cardano@gmail.com','cardano34');
3
4  INSERT INTO VISITEUR VALUES('1',
5  'michel.dumas@laposte.net','Dumas','Michel','68','Michou','cuisinella');
6
7  INSERT INTO CONTRIBUTEUR VALUES('1','1',
8  'michel.dumas@laposte.net','Dumas','Michel','68','Michou','cuisinella');
9  INSERT INTO CREER_SUPPRIMER_CONTRIB VALUES('1','1',now());
10
11 INSERT INTO CONTRIBUTEUR VALUES('2','1',
12 'roger.delcours@wanadoo.fr','Delcours','Roger','43','Roro','delro21');
13 INSERT INTO CREER_SUPPRIMER_CONTRIB VALUES('2','1',now());
14
15
16 INSERT INTO LIEU VALUES('1','Montpellier','Gambetta','5',34000,
17 3.876716,43.610769);
18
19 INSERT INTO LIEU VALUES('2','Paris','Place Montmartre','21',78000,
20 2.333333,48.866667);
21
22 INSERT INTO LIEU VALUES('3','Grenoble','Rue Millet','11',38000,
23 5.724524,45.188529);
24
25 INSERT INTO LIEU VALUES('4','Toulouse','Rue Clément','19',31000,
26 1.433333,43.600000);
27
28 INSERT INTO THEME VALUES('1','Exposition photo','Photo','1',now());
29 INSERT INTO THEME VALUES('2','Exposition peinture','Art','1',now());
30 INSERT INTO THEME VALUES('3','Maraude','Associatif','1',now());

```

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

```

1  INSERT INTO EVENEMENT VALUES('1','PhotArt','2019-12-21',400,900,'1','1','2',
2  "Exposition photo regroupant vos plus beaux clichés de l'année 2019",
3  now());
4  INSERT INTO EVENEMENT VALUES('2','MarryMe','2019-12-21',10,100,'1','1','2',
5  "Les plus belles photos de notre équipe de bénévole sur le thème du mariage",
6  now());
7  INSERT INTO EVENEMENT VALUES('3','Galerie Artis','2019-12-28',2,40,'1','2',

```

```

8      '3',
9      "Recueil de tableau de jeune étudiant en art de l'université de Grenoble",
10     now());
11     INSERT INTO EVENEMENT VALUES('4','Alerte Maraude','2019-12-24',0,20,'1','3',
12     '4',
13     "Nous allons nous déplacer dans le centre-ville pour
14     donner aux personnes dans le besoin des paniers repas",now());

```

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

```

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

```

Et finalement nous avons illustré l'utilisation des procédures de nombre d'utilisations d'un thème, la moyenne des effectifs des events à travers un appel de celle-ci sur une variable et enfin la ville qui apparait le plus dans les events avec le nombre correspondant : en suivant les différentes insertions, le résultat est 2 pour le nombre d'utilisation du theme avec l'id 1, la moyenne des effectifs est de 368 et la ville qui apparait le plus est Montpellier, 2 fois .

```

1      SET @nb=0;
2      CALL NOMBRE_UTILISATION_THEME(@nb,'1');
3      SELECT @nb as NombreTotal;
4
5      CALL MOYENNE_EFFECTIF_EVENT(@p0);
6      SELECT @p0 AS `MoyenneEffectif`;
7
8      CALL MAX_APPARITION_Ville(@p0, @p1);
9      SELECT @p0 AS `NomVille`, @p1 AS `MaxApparitionVille`;
10
11     /* La suppression du theme avec l'id 1,par exemple,
12     déclenche le trigger lié au suppression d'un theme et empêche
13     la suppression de celui-ci */
14     /* Décommentez la ligne suivante pour avoir le message du trigger */
15
16     /* DELETE FROM THEME WHERE idtheme = 1; */

```