Université de Liège
Faculté des Sciences appliquées

Projet Partie 2 : WeND(Y)'s Party Management System

INFO0009-2 : Bases de données (organisation générale)

Auteurs :
Alyssa Di Matteo s201486
Catherine Duchemin s202046
Manon Gerard s201354

Professeur :
C. Debruyne

3e année de Bachelier Ingénieur Civil

Année académique 2022 - 2023

1 Manipulations à effectuer pour initialiser la base de données

Pour initialiser la base de données, il suffit de composer le docker. Ensuite, en allant sur *localhost*, l'utilisateur aura accès à notre site avec la base de données initialisée. Le login et mot de passe sont respectivement"group05" et "secret". A noter que les champs où sont entrés le login et le mot de passe sont insensibles à la casse.

2 Architecture du site Web

Lors de la connexion sur le site grâce à *localhost*, l'utilisateur se trouve sur la page `index.php`. Sur celle-ci, il peut se connecter. Lorsque l'utilisateur tente de se connecter, il est redirigé vers `menu.php`. Mais ce dernier n'a pas forcément accès au menu. S'il s'est trompé, un message d'erreur apparaît et il peut cliquer sur un bouton pour retourner à la page de connexion. Lorsque l'utilisateur parvient à se connecter, il a accès au menu principal, qui permet d'accéder à nos différentes pages. Celles-ci seront décrites dans la partie "Requêtes". Nous avons tenté de sécuriser le site en empêchant un utilisateur malveillant d'accéder aux différentes pages en outre passant la connexion s'il se souvient du lien. Sur chacune des pages du site, il y a la possibilité de se déconnecter, dans quel cas la session prendra fin et le cookie sera supprimé. Il y a aussi la possibilité de retourner au menu si l'on ne se trouve pas déjà dans le menu. Ce menu peut rediriger vers `affichage_contraintes.php` ainsi que rediriger vers différentes pages qui seront détaillés dans la partie "Requêtes".

3 Sélection et affichage de tuples avec contraintes

La page qui permet l'affichage et la sélection de tuples en contraignant la valeur d'un ou plusieurs champs est la page `affichage_contraintes.php`. L'utilisateur va pouvoir sélectionner la table de son choix et remplir le formulaire afin de rechercher ce qu'il souhaite. Pour les chaînes de caractères, les contraintes sont des contraintes de contenances. Un exemple de requête pour la table EMPLOYEE est : "SELECT * FROM EMPLOYEE WHERE FIRSTNAME LIKE %Da% AND LASTNAME LIKE %Gue% ". Cette requête permettra de sélectionner les tuples où "Da" est contenu dans FIRSTNAME et où "Gue" dans LASTNAME. Cependant cette requête ne fonctionnera pas si certains attributs, qui sont des chaînes de caractères dans la base de données, sont NULL car LIKE ne prend pas en compte les valeurs NULL. Dans la base de données, nous avons remarqué que COMMENT dans la table LOCATION et DESCRIPTION ainsi que PLAYLIST dans la table EVENT sont des champs, qui sont des chaînes de caractères, qui pouvaient être NULL dans la base de données, nous avons donc dû gérer ces cas.

La recherche que nous avons implémentée est indépendante des majuscules et des minuscules. Rechercher "Rue"ou "rue" donnera le même résultat.

4 Requêtes

De façon générale, nous avons essayé de tester que les champs remplis par l'utilisateur correspondaient bien à ce qui était demandé. En effet, il est assez facile pour l'utilisateur de modifier notre site web et donc d'entrer ce qu'il souhaite.
Pour cela, il y a toujours une vérification que les champs obligatoires ont bien été remplis.

En ce qui concerne les listes déroulantes, il y a une vérification que la valeur associée correspond toujours à une valeur qui était proposée. Les types de données rentrées ont aussi été vérifié.

4.1 Modification des lieux

La page de modification des lieux est `modif_location.php`. Cette page permet d'ajouter, de modifier et de supprimer des lieux. Les lieux qui ont été réservés ne pourront pas être supprimé.

4.1.1 Ajout de nouveaux lieux

La première partie de la page consiste en un formulaire permettant d'ajouter un nouveau lieu. Un lieu peut être ajouté en ajoutant une rue, une ville, un code postal et un pays. L'identifiant et le commentaire sont quant à eux facultatifs. Lorsque l'utilisateur appuie sur le bouton "ajouter", le lieu est ajouté sauf si ce dernier ne respecte pas les règles de la base de données. Si un identifiant est donné, il faut s'assurer qu'un lieu ayant le même identifiant n'existe pas. Cela se fait avec SELECT COUNT(*) FROM ‘LOCATION‘ WHERE ‘ID‘ = 7.

La requête d'ajout dépend des champs entrés par l'utilisateur. Ainsi, les champs laissés vide ne se retrouveront pas dans la requête. La requête générale est la suivante :
INSERT INTO ‘LOCATION‘ (‘ID‘, ‘STREET‘, ‘CITY‘, ‘POSTAL_CODE‘, ‘COUNTRY‘, ‘COMMENT‘) VALUES (7, "Allée de la Découverte", "Liège", 4000, "Belgique", "Institut Montefiore")

4.1.2 Sélection d'un lieu existant

Afin de pouvoir sélectionner le lieu à modifier, une liste déroulante contenant tous les lieux a été créée. Les lieux existants ont été trouvées en utilisant la requête suivante :
SELECT * FROM ‘LOCATION‘

4.1.3 Mise à jour des informations

Une fois le lieu choisi, ses informations peuvent être modifiées, tout en veillant à la validité des entrées de l'utilisateur.
Si l'utilisateur souhaite modifier l'identifiant du lieu, il faut tout d'abord s'assurer que ce dernier n'est pas déjà utilisé par un autre lieu. La requête pour cela est :
SELECT COUNT(*) FROM ‘LOCATION‘ WHERE ‘ID‘ = 8

Une fois de plus, tous les champs ne se trouveront pas dans la requête finale en fonction de ceux remplis. La requête complète de mise à jour du lieu 4 est
UPDATE ‘LOCATION‘ SET ‘ID‘ = 8, ‘STREET‘ = "Allée de la Découverte", ‘CITY‘ = "Liège", ‘POSTAL_CODE‘ = 4000, ‘COUNTRY‘ = "Belgique", ‘COMMENT‘ = "Institut Montefiore" WHERE ‘ID‘ = 4

4.1.4 Suppression du lieu

Une fois le lieu choisi, il est aussi possible de le supprimer. Cependant, il ne faut pas supprimer un lieu s'il est réservé. Ainsi, il faut vérifier si le lieu choisi peut être supprimé

grâce à la requête SELECT ‘LOCATION‘ FROM ‘EVENT‘ WHERE ‘LOCATION‘ = 4

Une fois cette vérification faite, la suppression a lieu ou non. Si elle a lieu, la requête utilisée est DELETE FROM ‘LOCATION‘ WHERE ‘ID‘ = 4

L’identifiant de la localisation est aussi utilisé dans la table des événements. Lors de sa modification, il ne faut pas oublier de le mettre à jour dans la table des événements. Il faut aussi empêcher de supprimer un lieu s’il est réservé. Ainsi, dans l’initialisation de la base de donnée, nous avons mis ALTER TABLE ‘EVENT‘ ADD CONSTRAINT ‘EVENT_fk_6‘ FOREIGN KEY (‘LOCATION‘) REFERENCES ‘LOCATION‘ (‘ID‘) ON DELETE RESTRICT ON UPDATE CASCADE ;

4.2 Modification des CD

La page de modification des CD et des chansons est `modif_CD.php`. Pour cette page, le but est de pouvoir sélectionner un CD afin de pouvoir ajouter, modifier ou supprimer une chanson qui se trouve sur le CD choisi. Si une chanson est supprimée du CD, elle est aussi supprimée de toutes les listes de lectures dans lesquelles elle apparaît.

Pour cette page, nous avons rencontré un problème avec les clés étrangères de CONTAINS. Dans l’initialisation de la base de données, nous avons seulement ajouté une clé étrangère : ALTER TABLE ‘CONTAINS‘ ADD CONSTRAINT ‘CONTAINS_fk_1‘ FOREIGN KEY (‘PLAYLIST‘) REFERENCES ‘PLAYLIST‘ (‘NAME‘).

Nous savons que ce n’est pas la seule car CD_NUMBER et TRACK_NUMBER sont des clés étrangères liées aux clés primaires CD_NUMBER et TRACK_NUMBER de SONG. Cependant, en essayant d’ajouter : ALTER TABLE ‘CONTAINS‘ ADD CONSTRAINT ‘CONTAINS_fk_2‘ FOREIGN KEY (‘CD_NUMBER‘, ‘TRACK_NUMBER‘) REFERENCES ‘SONG‘ (‘CD_NUMBER‘, ‘TRACK_NUMBER‘) ON UPDATE CASCADE ON UPDATE DELETE, nous avons rencontrer un problème à cause du ON UPDATE que nous n’avons pas réussi à régler. Nous n’avons donc pas introduit cette clé étrangère dans l’initialisation de la base de données.

4.2.1 Choix du CD

La première étape sur cette page est le choix du CD dans lequel nous voulons ajouter, modifier ou supprimer des chansons. Pour cela, une liste déroulante contenant tous les CD de la base de donnée a été créée. Cette liste déroulante a été créée avec l’élément HTML `<select>` qui permet de fournir une liste d’options. L’utilisateur pourra donc choisir son CD en cliquant sur le bouton "choisir". Tous les CD ont été trouvés en utilisant la requête suivante : SELECT * FROM ‘LOCATION‘

Une fois le CD choisi, une chanson peut être ajoutée et une chanson peut être choisie soit pour la modifier ou soit pour la supprimer.

4.2.2 Ajout d’une chanson sur le CD

L’utilisateur peut ajouter sa chanson via un formulaire où il est obligé de remplir tous les champs proposés, c’est-à-dire tous les attributs de la table chanson sauf "CD_NUMBER" car il a déjà une valeur, qui est le numéro du CD que l’utilisateur a choisi. Pour rendre les

champs des formulaires obligatoires, le mot clé `<required>` est utilisé. La requête générale qui permet l'ajout est : INSERT INTO SONG(CD_NUMBER, TRACK_NUMBER, TITLE, ARTIST, DURATION, GENRE) VALUES(1, 12, 'Hello', 'Adele', 296, POP).

Pour la durée, le type dans le formulaire est "number". L'utilisateur doit rentrer la durée de la chanson en secondes. Avant d'exécuter la requête, la durée en secondes va être changée afin d'avoir un format hh :mm :ss. Pour le genre, nous avons choisi de faire une liste déroulante avec les genres qui se trouvent déjà dans la base de données, nous avons utilisé la requête : SELECT DISTINCT 'GENRE' FROM 'SONG'. Le mot clé DISTINCT a été utilisé afin d'éviter les doublons dans la liste déroulante. L'utilisateur aura le choix entre tous les genres déjà présent. Nous avons choisi cette façon de faire pour éviter que l'utilisateur rentre n'importe quel mot ou rentre un genre qui n'existe pas. Cette entrée sera vérifiée lors de l'ajout d'une chanson grâce à SELECT COUNT(*) FROM 'GENRE' WHERE 'NAME' = POP

Étant donné que l'attribut "track_number" d'une chanson est la clé primaire de la relation "SONG", il est unique. Il ne peut donc pas y avoir plusieurs chansons qui ont le même "track_number" dans le même CD. Afin de respecter cette condition, avant d'ajouter la chanson, nous vérifions que le "track_number" entré n'existe pas déjà dans ce CD. S'il existe déjà, la chanson ne peut pas être ajoutée, s'il n'existe pas la chanson est ajoutée. La requête générale utilisée est : SELECT * FROM SONG WHERE 'TRACK_NUMBER' = 1 AND 'CD_NUMBER'= 6. Si elle renvoie vrai, cela veut dire que la chanson avec le track_number 1 existe déjà dans le CD numéro 6.

4.2.3 Choix d'une chanson dans le CD

L'utilisateur peut choisir une chanson du CD via une liste déroulante qui a été créee de la même façon que la liste déroulante pour le CD. Cependant, ici les chansons proposées sont uniquement celles disponibles sur le CD qui a été choisi. La requête utilisée est : SELECT * FROM 'SONG' WHERE CD_NUMBER = :CD_NUMBER. Par exemple, si l'utilisateur a choisi le CD numéro 3, la requête est "SELECT * FROM 'SONG' WHERE CD_NUMBER = 3".

4.2.4 Modification d'une chanson sur le CD

L'utilisateur peut modifier le track_number, le titre, l'artiste, la durée et le genre d'une chanson. Pour permettre à l'utilisateur de modifier un attribut ou des attributs de chanson, un formulaire a été créé. L'utilisateur va pouvoir remplir les champs qu'il souhaite modifier, il n'est pas obligé de remplir tous les champs proposés. Pour le genre, nous avons aussi créé une liste déroulante avec les genres possibles comme lors de l'ajout de chanson. L'utilisateur aura donc la possibilité de modifier le genre de la chanson par un des genres proposé dans la liste déroulante. Le genre sera une fois de plus vérifiée afin d'éviter des entrées d'utilisateurs malveillants. Pour la durée, comme lors de l'ajout, le temps devra être rentré en secondes par l'utilisateur. Par exemple, une requête complète de modification de la chanson est : "UPDATE 'SONG' SET 'CD_NUMBER' = 3, 'TRACK_NUMBER' = 7, 'TITLE' = 'Shape Of You', 'ARTIST' = 'Ed Sheeran', 'DURATION'= 263 , 'GENRE' ='Pop' WHERE 'TRACK_NUMBER' = 9 AND 'CD_NUMBER'= 3". Cette requête modifie le track_number, le titre, l'artiste, la durée et le genre de la chanson qui a le track_number = 9 et qui se trouve sur le CD numéro 3.

Pour la modification, nous devons aussi faire attention au track_number, comme pour l'ajout. Afin de ne pas entrer un track_number qui existe déjà dans le CD, la même requête utilisée dans le cas de l'ajout est utlisé, par exemple : " SELECT * FROM SONG WHERE 'TRACK_NUMBER' = 1 AND 'CD_NUMBER'= 6". Si elle renvoie vrai, cela veut dire que la chanson avec le track_number 1 existe déjà dans le CD numéro 6 et que la chanson ne peut pas être modifiée. Pour pouvoir la modifié, il faudra que l'utilisateur entre un track_number qui n'existe pas encore dans le CD.

4.2.5 Suppression d'une chanson

L'utilisateur peut aussi supprimer la chanson choisie. Pour cela, il suffit d'appuyer sur le bouton "supprimer". Ce bouton va permettre d'exécuter la requête de suppression, la chanson sera donc supprimée du CD. Par exemple la requête, "DELETE FROM 'SONG' WHERE 'TRACK_NUMBER' = 1 AND 'CD_NUMBER'= 6" va permettre de supprimer du CD numéro 6 la chanson qui a le track_number 1.

La chanson doit aussi être supprimée des listes de lectures dans lesquelles elle se trouve. Pour cela, nous faisons une nouvelle requête qui va permettre de la supprimer. Par exemple, "DELETE FROM CONTAINS WHERE 'TRACK_NUMBER' = 1 AND 'CD_NUMBER'= 6" va permettre de supprimer de toutes les listes de lectures la chanson avec le track_number 6 qui se trouve sur le CD numéro 1.

4.3 Tableau de bord des événements

La requête s'exécute directement sans besoin de l'intervention de l'utilisateur.
Tout d'abord l'information sur la date du jour est assignée à la variable $today et affichée sur la page. Ensuite la requête consiste simplement à extraire les informations demandées de la table EVENT (DATE, NOM, RENTAL_FEE, THEME et PLAYLIST), puis à comparer ligne par ligne la date de chaque événement avec la variable $today :

- Si la date de l'événement est antérieure à $today alors PASSE est indiqué dans la colonne STATUT
- Si la date de l'événement est strictement égale à $today alors PRESENT est indiqué dans la colonne STATUT
- Si la date de l'événement est postérieure à $today alors FUTUR est indiqué dans la colonne STATUT

Pour ce qui concerne la colonne COÛT TOTAL, une variable $fee de 1500€ est ajoutée au rental_fee associé à l'événement si celui-ci n'est pas nul. Si rental_fee est nul alors c'est simplement la variable $fee qui est affichée.

4.4 Disponibilité des CD

Sur cette page, l'utilisateur commence par choisir sur quel attribut et dans quel ordre il souhaite que le tableau s'affiche à l'aide d'une liste déroulante.

La requête envoyée à la base de données doit réunir des informations extraites de 3 tables différentes : EVENT, CONTAINS et CD, CONTAINS étant la table qui permet de faire la jointure entre les tables EVENT et CD. Pour ce faire, la requête effectue une jointure naturelle entre ces 3 tables grâce à l'attribut CD_NUMBER qui leur est commun.

4.5 Modification des événements

La page de modification des événements est `modif_event.php`. Cette page permet de modifier des événements à venir. Remarquons que la date de l'événement est disponible, mais pas l'heure. Ainsi, les événements sont considérés comme à venir s'ils ont lieu après la date d'aujourd'hui. Le fuseau horaire utilisé est UTC comme indiqué par SET time_-zone = "+00 :00". Il a donc fallu trouver la date du lendemain, date à partir de laquelle les événements ont le droit d'être modifié. Cela s'est fait grâce à la requête SELECT DATE_ADD(CURDATE(), INTERVAL 1 DAY) AS date_demain

4.5.1 Sélection d'un événement à venir

Afin de pouvoir sélectionner l'événement à modifier, une liste déroulante contenant tous les événements futurs a été créé. Ils ont été trouvés en utilisant la requête suivante :
SELECT * FROM ‘EVENT‘ WHERE ‘DATE‘ > CURDATE()

4.5.2 Mise à jour des informations

Une fois l'événement à changer choisi, ses informations peuvent être modifiées, tout s'assurant de préserver la cohérence. Pour cela, certains champs ont dû être restreints à des valeurs spécifiques. Ces champs et leurs requêtes associés sont :

- la date est limitée à la valeur minimale de la date de demain trouvé précédemment
- le client : SELECT ‘CLIENT_NUMBER‘ FROM ‘CLIENT‘
- le manager : SELECT ‘ID‘ FROM ‘MANAGER‘
- le planificateur d'événement : SELECT ‘ID‘ FROM ‘EVENTPLANNER‘
- le DJ : SELECT ‘ID‘ FROM ‘DJ‘
- le thème : SELECT ‘NAME‘ FROM ‘THEME‘
- le lieu : SELECT ‘ID‘ FROM ‘LOCATION‘
- la playlist : SELECT ‘NAME‘ FROM ‘PLAYLIST‘

Tous ces champs seront revérifiés individuellement avant de faire la modification. En cas d'erreur, un message est affiché pour indiquer l'erreur à l'utilisateur l'erreur. Les requêtes de vérifications dans les cas où les champs ont été remplis sont les suivantes :

- la date : $date_demain <= "2023-06-30"
- le client : SELECT COUNT(*) FROM ‘CLIENT‘ WHERE ‘CLIENT_NUMBER‘ = 9
- le manager : SELECT COUNT(*) FROM ‘MANAGER‘ WHERE ‘ID‘ = 2
- le planificateur d'événement : SELECT COUNT(*) FROM ‘EVENTPLANNER‘ WHERE ‘ID‘ = 5
- le DJ : SELECT COUNT(*) FROM ‘DJ‘ WHERE ‘ID‘ = 9
- le thème : SELECT COUNT(*) FROM ‘THEME‘ WHERE ‘NAME‘ = "Mean Girls"
- le lieu : SELECT COUNT(*) FROM ‘LOCATION‘ WHERE ‘ID‘ = 2
- la playlist : SELECT COUNT(*) FROM ‘PLAYLIST‘ WHERE ‘NAME‘ = "Vampire Mood"

Si l'utilisateur souhaite modifier l'identifiant du lieu, il faut tout d'abord s'assurer que ce dernier n'est pas déjà utilisé par un autre lieu. En même temps, il y a une vérification que l'événement est bien à venir. La requête pour cela est :
SELECT ‘ID‘ FROM ‘EVENT‘ WHERE ‘ID‘ = 3 AND ‘DATE‘ > CURDATE()

Il convient d'avoir accès à la valeur de certains champs pour pouvoir faire d'autres vérifications. Si ceux-ci ont été changés, leur nouvelle valeur est retenue. Si au contraire, ils n'ont pas été changés, il faut aller rechercher leur valeur dans la base de données grâce aux requêtes suivantes :

- la date : SELECT ‘DATE‘ FROM ‘EVENT‘ WHERE ‘ID‘ = 3
- le manager : SELECT ‘MANAGER‘ FROM ‘EVENT‘ WHERE ‘ID‘ = 3
- le planificateur d'événement : SELECT ‘EVENT_PLANNER‘ FROM ‘EVENT‘ WHERE ‘ID‘ = 3
- le DJ : SELECT ‘DJ‘ FROM ‘EVENT‘ WHERE ‘ID‘ = 3
- la playlist : SELECT ‘PLAYLIST‘ FROM ‘EVENT‘ WHERE ‘ID‘ = 3

L'entreprise "WeND(Y)'s" avait réclamé qu'un responsable ne peut affecter que des employés qu'il supervise. Ainsi, si le manager est changé ou soit le dj, soit le planificateur est modifié, il faut s'assurer que la contrainte est toujours respectée.

- S'il y a un nouveau manager ou un nouveau planificateur d'événement : SELECT COUNT(*) FROM ‘SUPERVISION‘ WHERE ‘SUPERVISOR_ID‘ = 2 AND ‘EMPLOYEE_ID‘ = 5
- S'il y a un nouveau manager ou un nouveau DJ : SELECT COUNT(*) FROM ‘SUPERVISION‘ WHERE ‘SUPERVISOR_ID‘ = 2 AND ‘EMPLOYEE_ID‘ = 9

Il faut aussi s'assurer qu'un dj et planificateur d'événement ne travail pas plus d'une fois par jour.

- S'il y a une nouvelle date ou un nouveau planificateur d'événement : SELECT COUNT(*) FROM ‘EVENT‘ WHERE ‘DATE‘ = "2023-06-30" AND ‘EVENT_PLANNER‘ = 5
- S'il y a une nouvelle date ou un nouveau DJ : SELECT COUNT(*) FROM ‘EVENT‘ WHERE ‘DATE‘ = "2023-06-30" AND ‘DJ‘ = 9

Enfin, il faut vérifier que les CD sont disponibles. Cela a lieu si la date ou la playlist est changée. Pour vérifier cela, il faut chercher le nombre de CD disponibles à la date de l'événement, sans prendre en compte l'événement. L'événement est pris en compte pour déterminer le nombre de CD nécessaires. Une fois la requête retournant ces valeurs faites, il suffit de vérifier qu'il y a assez de cd disponibles par rapport au besoin. La requête est SELECT CD_NUMBER, nb_available, nb_needed FROM (SELECT * FROM (SELECT CD_NUMBER, COPIES - COUNT(*) AS nb_available FROM (SELECT PLAYLIST, CD_NUMBER, COPIES FROM CONTAINS NATURAL JOIN CD GROUP BY CD_NUMBER, PLAYLIST, COPIES) AS T1 NATURAL JOIN (SELECT PLAYLIST FROM EVENT WHERE ID != 3 AND DATE = "2023-06-30") AS T2 NATURAL JOIN (SELECT CD_NUMBER, COPIES FROM CD) AS T3 GROUP BY CD_NUMBER, COPIES) AS T4 UNION (SELECT CD_NUMBER, COPIES AS nb_available FROM CD WHERE CD_NUMBER NOT IN (SELECT CD_NUMBER FROM (SELECT PLAYLIST, CD_NUMBER, COPIES FROM CONTAINS NATURAL JOIN CD GROUP BY CD_NUMBER, PLAYLIST, COPIES) AS T11 NATURAL JOIN (SELECT PLAYLIST FROM EVENT WHERE ID != 3 AND DATE = "2023-06-30")

AS T12 NATURAL JOIN (SELECT CD_NUMBER, COPIES FROM CD) AS T13 GROUP BY CD_NUMBER, COPIES))) AS T14 NATURAL JOIN (SELECT CD_-NUMBER, COUNT(DISTINCT CD_NUMBER) AS nb_needed FROM CONTAINS WHERE PLAYLIST = "Vampire Mood" GROUP BY CD_NUMBER) AS T24

Une fois toutes ces vérifications faites, la modification peut avoir lieu. Une fois de plus, tous les champs ne se trouveront pas dans la requête finale en fonction de ceux remplis. La requête complète de mise à jour de l'événement 3 est
UPDATE ‘EVENT‘ SET ‘ID‘ = 12, ‘NAME‘ = "Proclamation", ‘DATE‘ = "2023-06-30", ‘DESCRIPTION‘ = "Proclamation de la facsa", ‘CLIENT‘ = 9, ‘MANAGER‘ = 2, ‘EVENT_PLANNER‘ = 5, ‘DJ‘ = 9, ‘THEME‘ = "Mean Girls", ‘TYPE‘ = "Birthday", ‘LOCATION‘ = 2, ‘RENTAL_FEE‘ = 150, ‘PLAYLIST‘ = "Vampire Mood" WHERE ‘ID‘ = 3

Pour assurer la cohérence de la base de donnée, des clés étrangères ont été introduites à la création des tables. Lors des modifications, nous testions tout de même ces relations afin de pouvoir afficher un message approprié à l'utilisateur.
ALTER TABLE ‘EVENT‘ ADD CONSTRAINT ‘EVENT_fk_1‘ FOREIGN KEY (‘CLIENT‘) REFERENCES ‘CLIENT‘ (‘CLIENT_NUMBER‘) ;
ALTER TABLE ‘EVENT‘ ADD CONSTRAINT ‘EVENT_fk_2‘ FOREIGN KEY (‘MA-NAGER‘) REFERENCES ‘MANAGER‘ (‘ID‘) ;
ALTER TABLE ‘EVENT‘ ADD CONSTRAINT ‘EVENT_fk_3‘ FOREIGN KEY (‘EVENT_-PLANNER‘) REFERENCES ‘EVENTPLANNER‘ (‘ID‘) ;
ALTER TABLE ‘EVENT‘ ADD CONSTRAINT ‘EVENT_fk_4‘ FOREIGN KEY (‘DJ‘) REFERENCES ‘DJ‘ (‘ID‘) ;
ALTER TABLE ‘EVENT‘ ADD CONSTRAINT ‘EVENT_fk_5‘ FOREIGN KEY (‘THEME‘) REFERENCES ‘THEME‘ (‘NAME‘) ;
ALTER TABLE ‘EVENT‘ ADD CONSTRAINT ‘EVENT_fk_6‘ FOREIGN KEY (‘LO-CATION‘) REFERENCES ‘LOCATION‘ (‘ID‘) ON DELETE
RESTRICT ON UPDATE CASCADE ;
ALTER TABLE ‘EVENT‘ ADD CONSTRAINT ‘EVENT_fk_7‘ FOREIGN KEY (‘PLAY-LIST‘) REFERENCES ‘PLAYLIST‘ (‘NAME‘) ;

Pour s'assurer de n'avoir qu'un DJ par jour et qu'un planificateur par jour, des contraintes ont aussi été ajoutés dans le tableau des événements. Celles-ci sont CONSTRAINT CHECK_UN_DJ_PAR_JOUR UNIQUE (‘DATE‘, ‘DJ‘) et CONSTRAINT CHECK_-UN_EVENT_PLANNER_PAR_JOUR UNIQUE (‘DATE‘, ‘EVENT_PLANNER‘)

4.6 Tableau de bord des CD

La requête s'exécute directement dans besoin de l'intervention de l'utilisateur.
La requête envoyée à la base de données est exécutée en deux temps :

— La première requête est une requête globale sur le contenu des tables CONTAINS, SONG et CD car on a besoin de ces 3 tables pour retrouver toutes les informations demandées. Une jointure naturelle est effectuée entre la table CONTAINS et une requête imbriquée afin de pouvoir compter le nombre de fois que les chansons d'un CD sont incluses dans des playlists. La requête imbriquée effectue un JOIN entre les tables CD et SONG afin de pouvoir calculer les durées totales, moyennes, minimun et

maximum des chansons sur chacun des CD. Le tout permet de réunir les informations demandées à l'exception des genres des chansons.

— La seconde requête est effectuée à l'intérieur de la boucle WHILE qui permet l'affichage des informations à l'écran ligne par ligne (une ligne = les informations sur un CD). Pour chaque CD, on extrait 3 tableaux de 2 colonnes : la première colonne est identique pour chaque tableau car elle correspond au numéro de CD. La seconde colonne contient les genres associés aux chansons du CD soit dans SPECIALIZES.GENRE, soit dans SPECIALIZES.SUBGENRE (avec un JOIN entre les tables SONG et SPECIALIZES), soit dans SONG.GENRE. On réalise ensuite une UNION pour obtenir un seul tableau de 2 colonnes, et on extrait les informations de la 2eme colonne avec GROUP_CONCAT pour obtenir une liste de tous les genres des chansons du CD.

5 Distribution des rôles et des tâches

— Initialisation de la base de données : Alyssa + Manon

— Interface Web de connexion et menu : Manon

— Sélection et affichage des tables avec contraintes : Alyssa

— Modification des lieux : Manon

— Modification des CD : Alyssa

— Tableau de bord des événements : Catherine

— Disponibilité des CD : Catherine

— Modification des événements : Manon

— Tableau de bord des CD : Catherine

— Rapport : Alyssa, Catherine et Manon

Pour réaliser ce projet, nous avons chacune contribuée aux différentes parties en s'aidant l'une l'autre.