Contraintes

1 Contraintes SQL

Contraintes CHECK. Cette contrainte permet de vérifier que les colonnes d'une ligne donnée vérifient une certaine condition. Par exemple, on pourrait vouloir vérifier que la valeur de la colonne `percentage` de la table `countrylanguage` est inférieure à 100 ou que la valeur de la colonne `return_date` de la table `rental` est soit `NULL` ou supérieure à `rental_date`. Ces contraintes se définissent lors de la création de la table. Par example :

```
CREATE TABLE solde(produit TEXT,
                   reduc_pourcentage INT,
                   CHECK reduc_pourcentage<=100 AND reduc_pourcentage%10=0);
```

Contraintes DEFAULT / NOT NULL / UNIQUE. Ces contraintes s'applique à une colonne en particulier. La contrainte `DEFAULT` spécifie une valeur par défaut pour remplir une colonne lorsque sa valeur n'est pas spécifiée, `NOT NULL` que la valeur de la colonne ne peut pas être `NULL` et `UNIQUE` que deux lignes différentes de la table ne peuvent pas avoir la même valeur sur les colonnes indiquées. Par exemple :

```
CREATE TABLE membre(nom VARCHAR(50) NOT NULL,
                    prenom VARCHAR(50) NOT NULL,
                    date_inscription DATE DEFAULT NOW(),
                    UNIQUE(nom, prenom)  );
```

Contraintes PRIMARY/FOREING KEY. Formellement, la contrainte `PRIMARY KEY` est équivalente à `UNIQUE` et `NOT NULL`. C'est un moyen d'identifier de façon unique chaque ligne. Chaque table est censée avoir une clé primaire. La contrainte `FOREIGN KEY` spécifie que la valeur d'une (ou de plusieurs) colonne contient des valeurs présentes dans une autre table. Par exemple, la colonne `capital` de `country` contient une valeur qui doit apparaître dans la colonne `id` de `city`. Par exemple :

```
CREATE TABLE membre(nom VARCHAR(50) NOT NULL, prenom VARCHAR(50) NOT NULL,
                    id INT, PRIMARY KEY id);
CREATE TABLE amis(id1 INT, id2 INT,
                  FOREING KEY (id1) REFERENCES membre (id),
                  FOREING KEY (id2) REFERENCES membre (id));
```

Contraintes EXCLUDE. Les contraintes `NOT NULL, DEFAULT, KEY, CHECK` portent uniquement sur une ligne tandis que la contrainte `UNIQUE` permet uniquement de tester des égalités entre lignes. On veut parfois vérifier une condition plus riche sur deux lignes. Par exemple, si notre table contient des réservations d'une salle entre `start_date` et `end_date`, on ne veut pas que deux réservations se chevauchent. On écrira :

```
CREATE TABLE reservation(
start_date DATE, end_date DATE,
EXCLUDE USING gist (daterange(start_date, end_date) WITH &&));
```

2 Modéliser les données d'une association

La Rustinette est une association qui permet à ses adhérents d'accéder à un local et à des outils pour réparer son vélo. Elle fabrique aussi des vélos et les vend pour se financer. Elle souhaite informatiser son système de gestion des adhérents et de ses comptes et elle a pensé à vous pour l'aider. Dessinez un modèle Entité-Association avec les cardinalités et les contraintes ainsi qu'une implémentation SQL réalisant ce modèle. L'association vous fournit un texte décrivant son fonctionnement pour vous aider :

Une personne souhaitant adhérer doit obligatoirement donner son nom, son prénom, sa date de naissance, son code postal et au moins une des informations suivantes pour pouvoir être jointe : un téléphone, un mail ou une adresse postale. Le prix de l'adhésion est de 15€ minimum mais l'adhérent peut choisir un prix plus élevé pour aider l'association. Une adhésion est valable un an. Chaque membre se voit associer un numéro d'adhérent unique (par ordre d'inscription) qui reste le même en cas de réadhésion. L'association propose aussi une adhésion familiale à 35€ minimum qui permet à toutes les personnes d'un même foyer d'accéder à l'atelier. Ces personnes doivent tout de même avoir chacune un numéro d'adhérent.

L'association vend des vélos qu'elle a fabriqués et dont le prix est fixé. Seuls les adhérents ayant une adhésion à jour peuvent acheter un vélo. Chaque vélo monté par l'association se voit associé un numéro unique. On a besoin de connaître sa couleur et la taille de ses roues (24", 26" ou 28").

Les pièces mises à disposition par l'association sont à prix libres et une boîte permet de récolter les dons des adhérents. Chaque soir, les bénévoles relèvent l'argent de cette boîte, recomptent la caisse et mettent une partie de l'argent de la caisse en banque. Le montant des dons, le total de la caisse (qui peut différer du total des recettes s'il y a eu un problème) et le total de l'argent mis à la banque sont consignés avec la date. L'association accepte les chèques et les espèces. Pour faire les comptes facilement, il faut avoir accès au mode de règlement de toute les transactions effectuées.

3 TP

Implémentez le schéma de base de données suivant pour gérer une plateforme d'échange de messages sous forme de conversation, en prenant soin de respecter les contraintes qui sont indiquées et les clés primaires/étrangères. On suppose qu'on a une table `membres` qui contient obligatoirement les informations suivantes sur ses membres : un login qui identifie les membres de façon unique (clé primaire), un mot de passe de plus de 8 caractères (fonction `length`), une date d'inscription, son status parmi : 'membre', 'modérateur', 'administrateur'. Le membre peut éventuellement avoir donné un email (qui doit contenir un '@') et son site personnel mais ce n'est pas obligatoire.

Un table `conversation`. Chaque conversation est identifié par un numéro unique et a un sujet. Une table `conversation_membres` qui contient des listes de logins et d'identifiant de conversation. On a aussi une table `message` qui contient un identifiant unique, l'identifiant de conversation dont il fait partie, le corps du message, la date d'envoi et le login de l'émetteur. Enfin une table `lu` qui contient une liste de logins, d'identifiants de message et de dates, la date étant la date de lecture du message par l'utilisateur (`NULL` par défaut si non-lu).

Comment forcer l'émetteur à faire partie de la conversation ?

4 DM

1. Créez une table `repertoire` avec les colonnes `id` de type `INT` qui est une clé primaire, `nom` de type `VARCHAR(50)`, `portable` de type `VARCHAR(10)`. Ajoutez une contrainte pour que le numéro de portable commence par 0, soit de longueur 10 exactement et ne contiennent que des chiffres. On peut tester si une chaîne de caractère `s` ne contient que des chiffres avec `s~'^[0-9]*'`.
2. Créez une table `amis` avec les colonnes `id1` et `id2` de type `INT` qui sont toutes deux des clés étrangères en référence à la colonne `id` de `repertoire`. Ajoutez une contrainte pour que `id1` et `id2` aient des valeurs distinctes.
3. Créez une table `historique_appels` avec une colonne `id` qui est une clé étrangère en référence à la colonne `id` de `repertoire`, une colonne `debut_appel` et une colonne `fin_appel` de type `timestamp` (date + heures + minutes + secondes). Ajoutez une contrainte pour interdire le fait que deux appels différents s'effectuent en même temps.