BDR – SI3 / MAM4
TP4 : Gestion d'un emploi du temps

1 Exposé du problème

On se propose de réaliser un système d'information pour gérer l'emploi du temps des étudiants des départements SI et MAM. Il s'agit de concevoir la base, de générer les différentes tables et de les remplir avec un jeu de données initial (cf les fichiers **creneaux.csv** et **edt_49.csv** dans `http://www.i3s.unice.fr/~rueher/Cours/BD/TP4`).

1.1 Fichiers de données

Le fichier **creneaux.csv** contient les caractéristiques générales des créneaux d'enseignement. Il comporte des lignes avec les champs suivants :

1. **code_module** varchar(20) : le code d'un module d'enseignement tel que 'BD', 'LFA' ou 'SYSTEME';
2. **libelle** varchar(50) : le nom détaillé du module;
3. **cursus** varchar(10): le nom d'un groupe d'étudiants pour lequel on veut pouvoir afficher un edt, comme "SI1", "MAM2".
4. **groupe** varchar(10) : le nom d'un groupe d'étudiants qui suivent ensemble un ou plusieurs enseignements, comme 'SI2', 'MA2', 'GR3' ou 'SM2'.
5. **durée** : la durée du créneau type, en nombre entier d'heures, et qui sera la même chaque fois que le créneau aura lieu.

Les fichiers **edt_x.csv**, que l'on reçoit chaque semaine pour alimenter la base, permettent de définir l'emploi du temps de la semaine créneau par créneau. Ils comportent des lignes avec les champs suivants:

1. **code_module** varchar(20) : code de module
2. **groupe** varchar(10) : groupe d'étudiants
3. **semaine** numeric(2,0) : numéro de la semaine (de 1 à 52)
4. **jour** numeric(1,0) : numéro du jour de la semaine (de 1 à 5)
5. **heured** numeric(1,0) : numéro de la tranche horaire du démarrage du créneau (de 1 à 8)
6. **salle** varchar(10) : un nom de salle

1.2 Contraintes et dépendances fonctionnelles

L'association **(code_module, groupe)** constitue un créneau type d'enseignement, se déroulant certaines semaines de l'année.
On suppose que les contraintes suivantes sont respectées:

1. Un créneau type (code_module, groupe) se déroule au plus une fois par semaine, et en une seule fois (une seule journée et sans interruption).
2. Pour une semaine donnée, un créneau type se déroule dans la même salle.
3. Chaque module est défini par son code. Il a un seul libellé.
4. un *groupe* est rattaché à un ou plusieurs cursus.
5. Chaque créneau type (code_module, groupe) a une seule durée.

Les dépendances fonctionnelles suivantes doivent être vérifiées:

1. code_module → libelle
2. code_module, groupe → duree
3. code_module, groupe, sem → jour, heured
4. code_module, groupe, sem → salle

Les dépendances 3 et 4 permettent de déduire la dépendance:

5. code_module, groupe, sem → jour, heured, salle

Remarques:

1. On suppose qu'il n'y a pas deux cours différents le même jour à la même heure dans la même salle.
2. Il existe une relation (m,n) entre un groupe et un cursus

2 Travail demandé

2.1 Conception des tables

On demande d'abord de concevoir le schéma, c'est à dire de définir les tables en 3NF. Les tables que l'on peut générer à partir des fichiers creneaux.csv et edt_49.csv ne contiennent pas de clé primaire. On va donc dériver des tables avec des clés primaires à partir des informations de ces tables et des dépendances fonctionnelles.

Une décomposition en 3NF est :
module(code_module, libelle)
creneau_type (code_module, groupe, duree)
cursus (cursus) % éventuellement on peut ajouter un libellé
groupe (groupe) % éventuellement on peut ajouter un libellé
cursus_creneau(cursus, groupe)
edt_sem (code_module, groupe, sem,jour, heured, salle)

Il faut aussi écrire les scripts SQL suivants :

1. Le script creer_schema.sql pour créer le schéma complet de la base;

```
-- Suppresion des tables

Drop table IF EXISTS  edt_sem, cursus_groupe, cursus, groupe, creneau_type,
        module, edt_init, creneau_init CASCADE;
-----------------------------------------
-- Tables de donnees
-----------------------------------------
create table creneau_init(
    code_module  varchar(20),
    libelle varchar(50),
    cursus  varchar(10),
    groupe   varchar(10),
    duree   numeric(1)
);
create table edt_init (
    module  varchar(20),
    groupe   varchar(10),
    sem     numeric(2),
    jour    numeric(1),
    heured  numeric(1),
    salle   varchar(10)
);
-----------------------------------------
-- Table pour le stockage de EDT
-----------------------------------------
create table cursus (
    nomcursus   varchar(10) primary key,
    libelle varchar(50)
);
create table groupe (
    nomcursus   varchar(10) primary key,
    libelle varchar(50)
);
create table cursus_groupe(          -- lien entre cursus et groupe
    nomcursus  varchar(10) references cursus,
    groupe   varchar(10) references groupe,
    PRIMARY KEY  (nomcursus, groupe)
```

```
);


create table module (                -- un module est defini par son code
    code_module  varchar(20) primary key,
    libelle varchar(50)
);

create table creneau_type (          -- chaque module(code_module,groupe)
                                     -- a une durée

    code_module  varchar(20) references module,
    groupe   varchar(10) references groupe,
    duree   numeric(1),
    primary key (code_module, groupe)
);

create table edt_sem (
    code_module  varchar(20) references module,
    groupe   varchar(10) references groupe,
    sem      numeric(2) CHECK ( sem >= 1 and sem <= 52),
    jour     numeric(1) CHECK ( jour  >= 1 and jour <= 5),
    heured   numeric(1) CHECK ( heured >= 1 and heured <= 8),
    salle    varchar(10),
      primary key (code_module, groupe, sem), -- code_module, groupe, sem
                                              --   -> jour, heured, duree
      unique (sem, jour, heured, salle ),      -- il n y a pas 2 cours en meme
                                              -- temps dans la même salle
      foreign key (code_module,groupe) references creneau_type
    );
```

---

2. Le script initialiser.sql pour alimenter la base avec les données stables indépendantes des semaines

---

```
/*
init_schema_edt.sql

*/
-- tables de donnees

\copy creneau_init from creneaux.csv delimiter ';' csv

-- ------------------------------------
-- Tables indépendantes de la semaine
-- ------------------------------------

insert into module
 select distinct C.code_module, C.libelle from creneau_init C;

insert into groupe
    select distinct C.groupe from creneau_init C;

insert into creneau_type
    select distinct C.code_module, C.groupe, C.duree from creneau_init C;

insert into cursus
    select distinct C.cursus from creneau_init C;

insert into cursus_groupe
    select distinct C.cursus, C.groupe from creneau_init C;
```

---

3. Le script remplir_49.sql qui permet d'alimenter la base avec l'emploi du temps de la semaine 49

```
-- Tables de données

\copy edt_init from edt_49.csv delimiter ';' csv

-- Table Semaine

insert into edt_sem   select * from edt_init;
```

Pour plus de souplesse, on pourra créer un script général:

```
\i create_schema_edt.sql
\i init_schema_edt.sql
\i fill_semaine_edt.sql
\i invited.sql
\i affichage.sql
\i test.sql
```

.

L'utilisation des références devra garantir que les contraintes liées aux dépendances fonctionnelles sont respectées.

2.2 Récupération du contenu d'un fichier csv:

\copy *table* from *ficher* delimiter ';' csv (si le delimiteur est un point virgule)

2.3 Création d'une table pour l'affichage d'un EDT

Cette table doit contenir les items qu'il faut afficher dans l'emploi du temps d'une semaine donnée d'un cursus donné. Pour un même jour, il peut y avoir plusieurs lignes s'il y a des cours en parallèle.

Pour déterminer, journée par journée, sur quelle ligne placer chacun des modules se déroulant ce jour-là, on pourra utiliser la table SQL suivante :

`affichage (module, groupe, jour, heure_deb,duree, salle, ligne)`

qu'on remplira par programme. Comme vous ne désirez pas faire apparaître en clair votre mot de passe dans le code java, la connexion se fera sous couvert de l'utilisateur `invited` dont le mot de passe est `invited`. Vous devez créeer cet utilisateur et lui donner uniquement les droits dont il a besoin.

```
DROP database IF EXISTS   invited;
revoke all on edt_sem, creneau_type,
     cursus, cursus_groupe, module, affichage FROM invited;
DROP user IF EXISTS invited;
create  user invited;
create database invited with owner invited;
alter user invited with password'invited';
alter user  invited set search_path to public ;

/*
-------------------------------------------------
Table pour affichage de EDT d'une semaine
-------------------------------------------------
*/
drop table IF EXISTS  affichage;

create table affichage (
    code_module  varchar(20) references module,
    groupe  varchar(10),
    jour    numeric(1) ,
    heured  numeric(2) ,
    duree  numeric(2),
    ligne   numeric(2),
    salle   varchar(10),
   unique (code_module, groupe),
    primary key (jour,ligne, heured)
);
/*
-------------------------------------------------
```

```
Attribution des droits (necessaires pour l'acces
via le programme Java)
-------------------------------------------------

*/
grant select, insert, delete, update on edt_sem, creneau_type,
      cursus, cursus_groupe, module, affichage to invited;

/*
revoke all on edt_sem, creneau_type,
      cursus, cursus_groupe, module, affichage FROM invited;
*/
```

Pour placer dans cette table les créneaux à afficher, y compris les créneaux vides (avec un champ module égal à NULL), on remplira la colonne "ligne" à l'aide d'un programme java qui devra effectuer les opérations suivantes:

1. Sélectionner un par un les créneaux à placer (avec une requête SQL dans la table générale des créneaux) en fonction de la semaine et du cursus choisi .

2. Pour chaque créneau à placer, déterminer (là encore avec une requête SQL) s'il existe un créneau vide susceptible de l'accueillir. S'il n'en n'existe pas, introduire une nouvelle "ligne " d'affichage pour le jour concerné, en ajoutant simplement à la table affichage un créneau vide de 8h démarrant en 1 pour le jour concerné, avec une nouvelle ligne.
   Pour "placer" le créneau dans l'emploi du temps, dans le créneau vide d'accueil sélectionné, il suffit alors d'assigner la ligne correspondante à ce créneau vide, et de remplacer le créneau vide par 0,1 ou 2 nouveaux créneaux vides, selon la configuration.
   Il faut aussi mettre des modules NULL dans les "trous".

2.4 Remplissage via JDBC de la table d'affichage

On vous demande d'utiliser jdbc pour vous connecter a votre base de données pour remplir la table affichage avec les enseignements de la semaine 49 pour les étudiants de votre cursus.

Vous aurez besoin d'un driver postgres. Ces drivers sont disponibles à l'adresse `https://jdbc.postgresql.org/download.html`.

La procédure de compilation et de test du programme Java est:

```
javac EdtRemplissage.java
java -cp postgresql-xxx.jar:. EdtRemplissage
```

où 'xxx' est la version de votre driver.

Voir fichier : EdtRemplissage.java
