Modélisation et définition de données II

1 Modélisation d'un emploi du temps

On désire gérer les emplois du temps des différents personnels (*enseignants, enseignants-chercheurs et chercheurs*) de l'UFR, pour savoir à tout moment s'il est possible de les joindre, et où. Pour cela, on considère que, en dehors des périodes où ils peuvent être joints dans leur bureau, les personnels peuvent être en réunion, quel que soit leur statut. Une réunion est désignée par une date précise, une tranche horaire et une salle de réunion. On veut connaître les autres personnes participant à la réunion.

Chaque personne est désignée par son nom, son prénom, le bureau où on peut la joindre.

Les *enseignants* peuvent, de plus, être en cours. Un cours est identifié par la matière enseignée à laquelle est affectée toujours la même salle. Il est désigné par une période de début et de fin (ex. de février à mai), un jour de la semaine, une tranche horaire et une salle de cours. Plusieurs enseignants peuvent enseigner la même matière dans l'année, à des jours et créneaux horaire différents. Un enseignant peut enseigner plusieurs fois la même matière dans l'année, à des périodes différentes.

Les *chercheurs* peuvent être à certaines périodes de l'année en mission en dehors de l'UFR. Une mission est désignée par une date de début et de fin, un lieu de mission avec le numéro de téléphone correspondant. Les chercheurs appartiennent à un laboratoire dont on peut joindre le secrétariat en cas d'urgence.

Les *enseignants-chercheurs* sont à la fois enseignants et chercheurs, avec un pourcentage plus ou moins grand d'enseignement (par rapport à la recherche) à effectuer. Ils peuvent donc être soit en réunion, soit en mission, soit en cours.

Établir le schéma Entité-Association de cette application. Ne pas oublier de préciser les cardinalités des associations et les identificateurs des entités. N'oubliez pas de respecter les règles de bonnes conduites.

Réfléchissez aux contraintes que vous pouvez imposer sur cette base de données.

Solution:

```
CREATE SCHEMA td6 ;

SET search_path TO td6 ;


CREATE SEQUENCE  td6.seq_personnel_id ;

CREATE TABLE personnel (personnel_id INTEGER PRIMARY KEY DEFAULT nextval('
    td6.seq_personnel_id'::regclass),
                        nom CHARACTER VARYING(30)  NOT NULL,
                        prenom CHARACTER VARYING(30) NOT NULL,
                        bureau INTEGER ,
                        UNIQUE(nom, prenom)) ;

CREATE SEQUENCE td6.laboratoire_id ;

CREATE TABLE td6.laboratoire(laboratoire_id INTEGER PRIMARY KEY DEFAULT
    nextval('td6.laboratoire_id'::regclass),
        nom CHARACTER VARYING(30) NOT NULL,
        secretariat CHARACTER(10) CHECK(secretariat SIMILAR TO '0[0-9]{9}')
     ,
        UNIQUE(nom) ) ;


CREATE TABLE td6.chercheur (laboratoire_id INTEGER REFERENCES td6.
    laboratoire) INHERITS(td6.personnel) ;
```

```
CREATE TABLE td6.enseignant() INHERITS(td6.personnel) ;

CREATE TABLE td6.enseignant_chercheur(pourcentage INTEGER CHECK(0 <
    pourcentage AND pourcentage<100)) INHERITS (td6.chercheur,td6.enseignant
    ) ;

CREATE SEQUENCE td6.seq_salle_id ;

CREATE TABLE td6.salle(salle_id INTEGER PRIMARY KEY DEFAULT nextval('td6.
    seq_salle_id'::regclass),
                       batiment character varying(30),
                       numero integer,
                       UNIQUE(batiment,numero));

CREATE SEQUENCE td6.seq_mission_id ;

CREATE TABLE td6.mission (mission_id INTEGER PRIMARY KEY DEFAULT nextval('
    td6.seq_mission_id'::regclass),
                          lieu CHARACTER VARYING (100) NOT NULL,
                          startdate TIMESTAMP ,
                          enddate TIMESTAMP CHECK (startdate < enddate),
                          personnel_id INTEGER REFERENCES td6.chercheur,
                          EXCLUDE USING gist (personnel_id WITH =, tsrange(
    startdate, enddate) WITH &&)) ;

ALTER TABLE td6.chercheur ADD CONSTRAINT pk_chercheur PRIMARY KEY (
    personnel_id) ;

ALTER TABLE td6.enseignant ADD CONSTRAINT pk_enseignant PRIMARY KEY (
    personnel_id) ;

ALTER TABLE td6.enseignant_chercheur ADD CONSTRAINT pk_enseignant_chercheur
     PRIMARY KEY (personnel_id) ;

CREATE SEQUENCE td6.seq_reunion_id ;

CREATE TABLE td6.reunion (reunion_id INTEGER PRIMARY KEY DEFAULT nextval('
    td6.seq_reunion_id'::regclass),
                          salle_id INTEGER REFERENCES td6.salle,
                          startdate TIMESTAMP ,
                          enddate TIMESTAMP CHECK (startdate < enddate),
                          EXCLUDE USING gist (salle_id WITH =, tsrange(
    startdate, enddate) WITH &&)) ;

CREATE TABLE td6.personnel_reunion (reunion_id INTEGER REFERENCES td6.
    reunion,
                                    personnel_id INTEGER REFERENCES td6.
    personnel) ;

CREATE SEQUENCE td6.seq_matiere_id ;

CREATE TABLE td6.matiere (matiere_id INTEGER PRIMARY KEY DEFAULT nextval('
    td6.seq_matiere_id'::regclass),
                          titre CHARACTER VARYING(30)) ;

CREATE SEQUENCE td6.seq_cours_id ;


CREATE TABLE td6.cours(cours_id INTEGER PRIMARY KEY DEFAULT nextval('td6.
    seq_cours_id'::regclass),
                       matiere_id INTEGER REFERENCES td6.matiere,
                       semestre INTEGER CHECK(semestre IN (1,2)),
                       jour CHARACTER VARYING(8) CHECK(jour IN ('lundi','
```

```
    mardi','mercredi','jeudi','vendredi','samedi','dimanche')),
                        heure_debut time,
                        heure_fin time CHECK (heure_debut < heure_fin),
                        personnel_id INTEGER REFERENCES td6.enseignant,
                        salle_id INTEGER REFERENCES td6.salle,
                        EXCLUDE USING gist(salle_id WITH =, jour WITH =,
    tsrange(date '2000-01-01' + heure_debut, date '2000-01-01' + heure_fin)
    WITH &&)
                        ) ;

ALTER TABLE td6.cours ADD COLUMN annee INTEGER CHECK (annee >=1970) ;

CREATE TABLE td6.calendrier(annee INTEGER CHECK (annee >= 1970),
                            semestre INTEGER CHECK (semestre IN (1,2)),
                            CONSTRAINT pk_calendrier PRIMARY KEY(annee,
    semestre),
                            date_debut date,
                            date_fin date CHECK(date_debut< date_fin),
                            CONSTRAINT compat CHECK (EXTRACT(YEAR FROM
    date_debut)=annee)) ;

ALTER TABLE td6.cours ADD CONSTRAINT excl_enseignant
      EXCLUDE USING gist(personnel_id WITH =,
                         jour WITH =,
                         tsrange(date '2000-01-01' + heure_debut, date '
    2000-01-01' + heure_fin) WITH &&) ;
```

2 Proposer des requêtes pour répondre aux questions suivantes

1. Combien y a-t-il de clients à Moscou ?

Solution:

```
SELECT COUNT(customer_id)
FROM sakila.customer NATURAL JOIN sakila.address NATURAL JOIN
 sakila.city
WHERE city='Moscow';
```

2. Pour chaque pays indiquez le nombre de clients.

Solution:

```
SELECT country,COUNT(customer_id)
FROM sakila.customer NATURAL JOIN sakila.address NATURAL JOIN
 sakila.city NATURAL JOIN sakila.country
GROUP BY country_id;
```

3. Pour chaque `customer_id`, quelle est la catégorie de film qu'il a le plus loués ?

Solution:

```
WITH NbRented AS (SELECT customer_id, category_id, COUNT(film_id)
as
```

```
N FROM rental NATURAL JOIN inventory NATURAL JOIN film_category
GROUP BY customer_id, category_id)

SELECT customer_id, category_id FROM NbRented as R WHERE N = (
SELECT
MAX(N) FROM NbRented WHERE customer_id = R.customer_id);
```

4. Quel est le nombre moyen de clients par pays ?

Solution:

```
WITH CustomerPerCountry AS
(SELECT country,COUNT(customer_id) AS N
FROM sakila.customer NATURAL JOIN sakila.address NATURAL JOIN
sakila.city NATURAL JOIN sakila.country
GROUP BY country_id)

SELECT AVG(N) FROM CustomerPerCountry;
```

5. Lister les pays pour lesquels toutes les villes ont au moins un client.

Solution:

```
WITH CityWithNoCustomer AS
(SELECT city_id,country_id FROM sakila.city as V
WHERE NOT EXISTS
(SELECT * FROM customer NATURAL JOIN address WHERE city_id=V.
city_id))

SELECT country_id FROM sakila.country
WHERE country_id NOT IN (SELECT country_id FROM CityWithNoCustomer
);
```

6. Déterminer la liste des films disponibles dans toutes les pays.

Solution:

L'agrégation est pratique dans ce cas. On compte le nombre de pays où le film est disponible et on vérifie que c'est le même nombre que le nombre de pays.

```
SELECT film_id FROM inventory NATURAL JOIN store
NATURAL JOIN address
NATURAL JOIN city
GROUP BY film_id
HAVING  COUNT(DISTINCT country_id) =
(SELECT COUNT(country_id) FROM sakila.country);
```

7. Créer un rapport qui montre le titre, la note, la catégorie, le prix, la longueur des films qui durent plus de trois heures triés par taux de location décroissants.

3 Contraintes

On utilisera les schémas Sakila et world. Exprimer en SQL les contraintes suivantes :

1. Une capitale doit appartenir au pays dont elle est la capitale.

2. La somme des pourcentages des langues parlées dans un pays ne doit pas excéder 100.
3. La somme des populations des villes d'un pays doit être inférieur ou égal à sa population totale.
4. Un même DVD ne peut pas être loué deux fois en même temps.
5. Les locations de DVD sont toujours effectuées par un employé qui travaille dans le magasin où se trouve le DVD.
6. Les clients ont été enregistrés dans la base de données (champ `create_date`) avant leur première location.