Manipulation d'une base de données avec extension relationnelle PostgreSQL
Corrigé

1 Héritage

.) Créez une relation Communes Hierarchieavec comme attributs : numero (entier), nom (texte), population (entier), populationActiveHomme (entier), populationActiveFemme (entier).

```
CREATE TABLE communesHierarchie     (numero                 integer,
                                    nom                     text,
                                    population              integer,
                                    populationActiveHomme   integer,
                                    populationActiveFemme   integer );
```

.) Créez une relation PrefecturesHierarchie avec comme attribut supplémentaire de communes nomPrefet (texte) en utilisant l'héritage de PostgreSQL.

```
CREATE TABLE prefecturesHierarchie (nomPrefet text)
        INHERITS (communesHierarchie);
```

.) Insérez les tuples dans les relations CommunesHierarchie et prefecturesHierarchie à partir des relations communes et prefectures.

2 Manipulations

2.1 Tuples

.) Donnez toutes les agglomérations disponibles (communes et préfectures) → 63 Tuples

```
select * from communesHierarchie;
```

```
 numero |        nom          | population | populationactivehomme | populationactivefemme
----------+---------------------+------------+-----------------------+-----------------------
        1 | AUCAMVILLE  |       3807  |                1009   |                847
        2 | AUREVILLE       |        456  |                 138   |                 93
...
(63 rows)
```

.) Donnez toutes les communes non-préfectures → 62 tuples

```
select * from only communesHierarchie;
```

numero	nom	population	populationactivehomme	populationactivefemme
1	AUCAMVILLE	3807	1009	847
2	AUREVILLE	456	138	93
...				
56	LA SALVETAT-SAINT-GILLES	4285	1178	880
57	SEILH	818	217	194
59	TOURNEFEUILLE	16680	4384	3547
60	L'UNION	11759	2897	2304
61	VIEILLE-TOULOUSE	868	234	176
62	VIGOULET-AUZIL	932	247	180
63	VILLENEUVE-TOLOSANE	7566	1999	1522

(62 rows)

Important : il n'y a pas le 58 (Toulouse)

2.2 Fonctions

.) Créez une fonction differenceMinimum qui renvoie la différence entre le minimum des populations actives des hommes et le minimum des populations actives des femmes pour les communes ayant une population supérieure à une valeur donnée en paramètre.

```
create function differenceMinimum (int4) returns int4 as
        'select min (populationActiveHomme) - min (populationActiveFemme)
                        from communes
                where population > $1'
        language 'sql';
```

.) Appliquez cette fonction pour une population de 9500 habitants

```
select differenceMinimum (9500);
```

Valeur attendue : 506

.) Démontrez que le résultat est juste :

```
select  min (populationActiveHomme)
from    communes
where   population >9500;

select  min (populationActiveFemme)
from    communes
where   population >9500;
```

2.3 Domaine tuple

.) Créez une relation Regions avec comme attribut : codeRegion (entier), nomRegion (texte) et un attribut, infoPrefecture ayant un tuple de la relation PrefecturesHierarchie comme domaine.

```
create table regions (      codeRegion      integer,
                            nomRegion       text,
                            infoPrefecture  prefecturesHierarchie);
```

.) Ecrire une fonction, donnePrefecture qui renvoie le tuple de la relation Prefectures ayant pour valeur de l'attribut nom la chaine de caractère passée en paramètre.

```
create function donnePrefecture (text)  returns prefectures as
        'select *
        from  prefecturesHierarchie
        where nom = $1'
        language 'sql';
```

.) Insérez le tuple suivant dans la relation Regions
→ codeRegion : 14,
nomRegion : Midi-Pyrénées

```
insert into regions (codeRegion, nomRegion) values (14, 'Midi-Pyrenees');
```

.) Utilisez la fonction pour insérer la préfecture associée . Attention : Les noms de préfecture sont en majuscule.

```
Select  *
from    regions;

update regions
set infoPrefecture = donnePrefecture ('TOULOUSE')
where codeRegion = 14;

select  *
from    regions;
```

.) Donnez à partir d'un accès à la relation Regions (uniquement), la population active homme de la préfecture de la région 14 → Valeur attendue : 84988

```
create function extraitPrefecture (integer) returns prefecturesHierarchie as
        'Select infoPrefecture
        From   Regions
        Where  codeRegion = $1'
        language 'sql';

select  populationActiveHomme(extraitPrefecture(14))
from    regions;
```

.) Est ce que la modification de la population active des hommes dans la relation PrefecturesHierarchie modifie le résultat de la requête précédente ? Qu'en concluez vous ?

```
UpdateprefecturesHierarchie
```

```
set       populationactivehomme = 2
where  numero = 58;

select   populationActiveHomme
from    prefecturesHierarchie
where  numero = 58;

select   populationActiveHomme(extraitPrefecture(14))
from    regions;
```

Pas de modification, il n'y a donc pas de partage de tuples mais seulement une recopie du/des tuples.

```
update prefecturesHierarchie
set       populationactivehomme = 84988
where  numero = 58;
```

3 Règles

Créez une règle, complement, qui garantit que toute insertion faite dans la relation communes sera répercutée dans communesHierarchie.
Insérez une commune avec les caractéristiques (200, 'communeTest', 1000, 300, 300) et montrez qu'une commune identique a été insérée dans communesHierarchie.

```
-- create rule complement as on insert
-- to communes
-- do also insert into communesHierarchie
--  values (new.numero, new.nom, new.population,new.populationActiveHomme,
            new.populationActiveFemme);

create rule complement as on insert
to communes
do also insert into communesHierarchie select * from communes where numero = new.numero;

select count(*) as communesHierarchieAvant from communesHierarchie;

insert into communes values (200, 'communeTest', 1000, 300, 300);

select count(*) as communesHierarchieApres from communesHierarchie;

select * from communesHierarchie where numero = 200;

delete from communes where numero = 200;
delete from communesHierarchie where numero = 200;
```

4 Nettoyage

```
drop function differenceMinimum (integer);

drop function extraitPrefecture (integer);

drop function donnePrefecture (text);

drop table regions;

drop table visite;

drop rule complement on communes;

drop table prefecturesHierarchie;

drop table communesHierarchie;
```