Extension d'une base de données relationnelle : PostgreSQL
Corrigé

1 Héritage

1.1 Schéma

.) Créez une relation COMMUNES_HIERARCHIE avec comme attributs : NUM_COMMUNE (entier), NOM (texte), POPULATION (entier), POPULATION_ACTIVE_HOMME (entier), POPULATION_ACTIVE_FEMME (entier).

```
CREATE TABLE COMMUNES_HIERARCHIE
                    (NUM_COMMUNE                INTEGER,
                    NOM                         TEXT,
                    POPULATION                  INTEGER,
                    POPULATION_ACTIVE_HOMME     INTEGER,
                    POPULATION_ACTIVE_FEMME     INTEGER  );
```

.) Créez une relation PREFECTURES_HIERARCHIE avec comme attribut supplémentaire de COMMUNES_HIERARCHIE, NOM_PREFET (texte) en utilisant l'héritage de PostgreSQL.

```
CREATE TABLE PREFECTURES_HIERARCHIE (NOM_PREFET TEXT)
        INHERITS (COMMUNES_HIERARCHIE);
```

1.2 Instance

.) Insérez les tuples dans les relations COMMUNES_HIERARCHIE et PREFECTURES_HIERARCHIE à partir des relations COMMUNES (sauf la ville de TOULOUSE) et PREFECTURES de la base tp. Le schéma de COMMUNES est (ID, NOM, POPULATIONTOTALE, ACTIFHOMME, ACTIFFEMME). Le schéma de PREFECTURES est (ID, NOM, POPULATIONTOTALE, ACTIFHOMME, ACTIFFEMME, NOMPREFET).

Si DBLINK n'est pas installé : CREATE EXTENSION dblink ;
Puis

```
INSERT INTO COMMUNES_HIERARCHIE (NUM_COMMUNE, NOM, POPULATION,
        POPULATION_ACTIVE_HOMME, POPULATION_ACTIVE_FEMME)
SELECT *
FROM dblink('dbname=tp',
```

```
        'SELECT ID, NOM, POPULATIONTOTALE,ACTIFHOMME, ACTIFFEMME
        FROM COMMUNES
        WHERE NOM <> "TOULOUSE" ') as COMMUNES_HIERARCHIE
                (NUM_COMMUNE INTEGER, NOM TEXT, POPULATION INTEGER,
                POPULATION_ACTIVE_HOMME INTEGER,
                POPULATION_ACTIVE_FEMME INTEGER );
```

Attention autour de Toulouse c'est deux fois des quotes simples.

```
INSERT INTO PREFECTURES_HIERARCHIE (NUM_COMMUNE, NOM, POPULATION,
     POPULATION_ACTIVE_HOMME, POPULATION_ACTIVE_FEMME, NOM_PREFET)
SELECT *
FROM dblink('dbname=tp',
    'SELECT ID, NOM, POPULATIONTOTALE, ACTIFHOMME, ACTIFFEMME,
            NOMPREFET
            FROM  PREFECTURES') as PREFECTURES_HIERARCHIE
                (NUM_COMMUNE INTEGER, NOM TEXT, POPULATION INTEGER,
                        POPULATION_ACTIVE_HOMME INTEGER,
                        POPULATION_ACTIVE_FEMME INTEGER, NOM_PREFET TEXT );
```

2 Manipulations

2.1 Tuples

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

```
SELECT *
FROM COMMUNES_HIERARCHIE;
```

NUM_COMMUNE	NOM	POPULATION	POPULATION_ACTIVE_HOMME	POPULATION_ACTIVE_FEMME
1	AUCAMVILLE	3807	1009	847
2	AUREVILLE	456	138	93
...				
(63 rows)				

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

```
SELECT      *
FROM        ONLY COMMUNES_HIERARCHIE;
```

NUM_COMMUNE	NOM	POPULATION	POPULATION_ACTIVE_HOMME	POPULATION_ACTIVE_FEMME
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 (INTEGER) RETURNS INTEGER as
        'SELECT       MIN (POPULATION_ACTIVE_HOMME) -
                      MIN (POPULATION_ACTIVE_FEMME)
        FROM     COMMUNES_HIERARCHIE
        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 (POPULATION_ACTIVE_HOMME)
FROM        COMMUNES_HIERARCHIE
WHERE       POPULATION > 9500;

SELECT      MIN (POPULATION_ACTIVE_FEMME)
FROM        COMMUNES_HIERARCHIE
WHERE       POPULATION > 9500;
```

2.3 Domaine tuple

2.3.1 Schéma

Créez une relation REGIONS avec comme attribut : NUM_REGION (entier), NOM (texte) et un

attribut, INFO_PREFECTURE ayant un tuple de la relation PREFECTURES_HIERARCHIE comme domaine.

```
CREATE TABLE REGIONS (
        NUM_REGION          INTEGER,
        NOM                 TEXT,
        INFO_PREFECTURE     PREFECTURES_HIERARCHIE);
```

2.3.2 Manipulation

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

```
CREATE FUNCTION donnePrefecture (TEXT)  RETURNS
PREFECTURES_HIERARCHIE
    as
    ' SELECT    *
      FROM      PREFECTURES_HIERARCHIE
      WHERE     NOM = $1'
    LANGUAGE 'sql';
```

.) Insérez le tuple suivant dans la relation REGIONS

→ NUM_REGION : 14,
NOM : Midi-Pyrénées

```
INSERT INTO REGIONS (NUM_REGION, NOM) 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 INFO_PREFECTURE = donnePrefecture ('TOULOUSE')
WHERE NUM_REGION = 14;

SELECT      *
FROM        REGIONS;
```

2.3.3 Composition

.) 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 populationActiveHommeVille (TEXT) RETURNS INTEGER as
        'SELECT      POPULATION_ACTIVE_HOMME
        FROM         COMMUNES_HIERARCHIE
        WHERE        NOM = $1'
        LANGUAGE 'sql';

CREATE or REPLACE FUNCTION extraitPrefectureV1 (INTEGER)
                RETURNS TEXT as
        'SELECT (INFO_PREFECTURE).nom
        FROM   REGIONS
        WHERE  NUM_REGION = $1'
        LANGUAGE 'sql';

SELECT      populationActiveHommeVille (extraitPrefectureV1(14)) as hommeActifs
FROM        REGIONS;



CREATE or REPLACE FUNCTION extraitNomPrefecture
                                        (PREFECTURES_HIERARCHIE)
                RETURNS TEXT as
        'SELECT $1.nom'
        LANGUAGE 'sql';

CREATE or REPLACE FUNCTION extraitPrefectureV2 (INTEGER)
                RETURNS TEXT as
        ' SELECT extraitNomPrefecture (INFO_PREFECTURE)
          FROM   REGIONS
          WHERE  NUM_REGION = $1'
        LANGUAGE 'sql';

SELECT      populationActiveHommeVille (extraitPrefectureV2(14)) as hommeActifs
FROM        REGIONS;
```

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

```
UPDATE      PREFECTURES_HIERARCHIE
SET         POPULATION_ACTIVE_HOMME = 2
WHERE       NUM_COMMUNE = 58;

SELECT       POPULATION_ACTIVE_HOMME
FROM        PREFECTURES_HIERARCHIE
WHERE       NUM_COMMUNE = 58;
```

```
SELECT    populationActiveHommeVille(extraitPrefectureV2(14))
FROM      REGIONS;

SELECT    *
FROM      REGIONS;
```

Conclusion : Nous pouvons constater que les résultats sont différents, il n'y a donc pas de partage de tuples mais seulement une recopie du/des tuples au moment de l'affectation dans REGIONS.

Remise en état de la base :

```
UPDATE    PREFECTURES_HIERARCHIE
SET       POPULATION_ACTIVE_HOMME = 84988
WHERE     NUM_COMMUNE = 58;
```

3 Règles

3.1 Création

Créez une relation COMMUNES copie de la relation COMMUNES_HIERARCHIE et recopiez y les tuples.
Créez une règle, complement, qui garantit que toute insertion faite dans la relation communes sera répercutée dans COMMUNES_HIERARCHIE.

```
CREATE TABLE COMMUNES (NUM_COMMUNE                  INTEGER,
                       NOM                          TEXT,
                       POPULATION                   INTEGER,
                       POPULATION_ACTIVE_HOMME      INTEGER,
                       POPULATION_ACTIVE_FEMME      INTEGER  );

INSERT INTO COMMUNES
    SELECT    *
    FROM      COMMUNES_HIERARCHIE;

CREATE RULE complement as on INSERT to COMMUNES
    do also
    INSERT INTO COMMUNES_HIERARCHIE (NUM_COMMUNE, NOM,
        POPULATION, POPULATION_ACTIVE_HOMME,
        POPULATION_ACTIVE_FEMME)
    VALUES (new.NUM_COMMUNE , new.NOM , new.POPULATION,
    new.POPULATION_ACTIVE_HOMME, new.POPULATION_ACTIVE_FEMME );


– CREATE RULE complement as on INSERT to  COMMUNES
-- do also INSERT INTO COMMUNES_HIERARCHIE;
--      SELECT    *
--      FROM      COMMUNES
--      WHERE     NUM_COMMUNE = new.NUM_COMMUNE ;
```

3.2 Insertion

Insérez une commune avec les caractéristiques (200, 'communeTest', 1000, 300, 300) et montrez qu'une commune identique a été insérée dans COMMUNES_HIERARCHIE.

```
SELECT      COUNT(*) as communesHierarchieAvant
FROM        COMMUNES_HIERARCHIE;

INSERT INTO COMMUNES VALUES (200, 'communeTest', 1000, 300, 300);

SELECT      COUNT(*) as communesHierarchieApres
FROM        COMMUNES_HIERARCHIE;

SELECT      *
FROM        COMMUNES_HIERARCHIE
WHERE       NUM_COMMUNE = 200;

DELETE FROM COMMUNES                WHERE NUM_COMMUNE = 200;
DELETE FROM COMMUNES_HIERARCHIE     WHERE NUM_COMMUNE = 200;
```

4 Nettoyage

.) Faîtes la suppression manuelle des éléments créés.

```
DROP FUNCTION differenceMinimum (INTEGER);

DROP FUNCTION extraitPrefectureV1(INTEGER);
DROP FUNCTION extraitPrefectureV2(INTEGER);

DROP FUNCTION extraitNomPrefecture (PREFECTURES_HIERARCHIE);

DROP FUNCTION donnePrefecture (TEXT);

DROP FUNCTION  populationActiveHommeVille (TEXT);

DROP TABLE REGIONS;


DROP RULE complement ON COMMUNES;

DROP TABLE PREFECTURES_HIERARCHIE;

DROP TABLE COMMUNES_HIERARCHIE;

DROP TABLE COMMUNES;
```