Manipulation d'une base avec extension spatiale (PostgreSQL / PostGIS / PgRouting)
Corrigé

1 Gestion de la métabase

1.1 Donnez le type de l'attribut the_geom de la relation ville

```
SELECT      UDT_SCHEMA, UDT_NAME
FROM        INFORMATION_SCHEMA.COLUMNS
WHERE       TABLE_CATALOG = 'tp' and
            TABLE_NAME = 'villes' and
            COLUMN_NAME = 'the_geom';

udt_schema | udt_name
---------------+----------
  public       | geometry
(1 row)
```

1.2 Donnez les types géométriques des villes

```
SELECT      ID, ST_GeometryType (the_geom) as geometrie
FROM        VILLES;

id |    geometrie
----+-----------------
 1 | ST_Polygon
 2 | ST_MultiPolygon
 3 | ST_Polygon
(3 rows)
```

2 Gestion des données physiques

2.1 Fonction dans le Select (Combien de sommets forment la route 1 ?)

```
SELECT      ST_NumPoints(the_geom) as nbrepoints
FROM        ROUTES
WHERE       ID = 1;

nbrepoints
------------
         3
(1 row)
```

2.2 Gestion des points : fonction dans le Where – centroide (Donnez les bâtiments pour lesquels le centroïde n'est pas dans le bâtiment et la distance aux bords)

```
SELECT      ID, ST_AsText (ST_centroid(the_geom)) as centroide,
            ST_Distance (ST_centroid(the_geom), the_geom) as distance
FROM        BATIMENTS
WHERE       ST_Distance (ST_Centroid(the_geom), the_geom) > 0;

id |              centroide                                    |     distance
----+-----------------------------------------------------------+-------------------
3 | POINT(2.10714285714286 2.14285714285714) | 0.142857142857143
(1 row)
```

2.3 Gestion des points : fonction dans le Select – centroide (Donnez le point associe dans l'objet pour mettre un libelle sur l'objet)

```
SELECT      ST_AsText(ST_PointOnSurface(the_geom)) as label
FROM        BATIMENTS
WHERE       ID = 3;

    label
------------------
POINT(2.75 2.5)
(1 row)
```

2.4 Gestion des zones : fonction dans le Select (Donnez la superficie de la ville 1)

```
SELECT      id, ST_Area(the_geom) as surface
FROM        VILLES
WHERE       ID = 1;

id | surface
----+---------
 1 |      16
(1 row)
```

2.5 Gestion des zones : fonction dans le Where (Donnez les villes dont la superficie est inférieure strictement à la moyenne des superficies des villes)

```
SELECT      ID
FROM        VILLES
WHERE       ST_Area(the_geom) <
                    (SELECT     AVG(ST_Area(the_geom))
                     FROM       VILLES);
id
----
 3
(1 row)
```

2.6 Gestion des lignes : fonction dans le Select (Donnez le périmètre de la ville 2)

```
SELECT      ID, ST_Perimeter(the_geom) as perimetre
FROM        VILLES
WHERE       ID = 2;

 id | perimetre
----+-----------
 2 |       20
(1 row)
```

2.7 Gestion des lignes : orientation (Donnez le point de départ et le point d'arrivée de la route 2)

```
SELECT      ST_AsText (ST_StartPoint(the_geom)) as depart,
            ST_AsText (ST_EndPoint(the_geom)) as arrivee
FROM        ROUTES
WHERE       ID = 2;

   depart        |   arrivee
-----------------+-------------
 POINT(11 7) | POINT(11 9)
(1 row)
```

2.8 Gestion des lignes : distance (Donnez la distance de la route 3 à la ville 2)

```
SELECT      ST_Distance (V.the_geom, R.the_geom) as distance
FROM        VILLES V, ROUTES R
WHERE       V.ID = 2 and R.ID = 3;

 distance
----------
        3
(1 row)
```

2.9 Gestion de la topologie : Zone : fonction dans le Where (Quelles villes contiennent un trou ou qui ne sont pas d'un seul tenant ?)

```
SELECT      ID
FROM        VILLES
WHERE       ST_Nrings(the_geom) > 1;

 id
----
  2
  3
(2 rows)
```

2.10 Gestion de la topologie : Zone : fonction dans le Select (Donnez les coordonnées du rectangle englobant du Magasin 4)

```
SELECT    ST_XMin(the_geom) as xmin, ST_YMax(the_geom) as xmax,
          ST_YMin(the_geom) as ymin, ST_YMax(the_geom) as ymax
FROM      BATIMENTS
WHERE     ID = 4;
```

```
 xmin | xmax | ymin | ymax
-------+-------+-------+------
   0   |   1    |  10   |  11
(1 row)
```

2.11 Topologie : opérateurs et prédicats (Où se coupent les routes 1 et 2 ?)

```
SELECT    ST_AsText(ST_Intersection(R1.the_geom, R2.the_geom)) as intersection
FROM      ROUTES R1, ROUTES R2
WHERE     ST_Intersects (R1.the_geom, R2.the_geom) and
          R1.ID = 1 and R2.ID = 2;
```

```
       intersection
---------------------------
 POINT(11 7.45454545454545)
(1 row)
```

2.12 Opérateur physique indépendant de la forme (Donnez la longueur de la route 2)

```
SELECT    ST_Length(the_geom) as longueur
FROM      ROUTES
WHERE     ID = 2;
```

```
 longueur
----------
    2
(1 row)
```

2.13 Composition d'opération (Quelle est la distance relative entre le début de la route 2 et son croisement avec la route 1?)

```
SELECT    ST_Line_Locate_Point
              (R1.the_geom,
               ST_Intersection(R1.the_geom, R2.the_geom)) as Pourcentage
FROM      ROUTES R1, ROUTES R2
WHERE     R1.ID = 2 and R2.ID = 1;
```

```
 distancerelative
------------------
```

```
0.227272727272727
(1 row)
```

Intersection : (11, 7.45) -> 0,45 / 2

2.14 Composition d'opérations (Dans quelle ville suis-je si je parcours la moitié de la route 1 ?)

```
SELECT      V.ID as ville
FROM        VILLES V, ROUTES R
WHERE       ST_Intersects (V.the_geom,
                ST_Line_Interpolate_Point(R.the_geom, 0.5)) and
            R.ID = 1;
 ville
-------
     2
(1 row)
```

2.15 Donnez les coordonnées de la deuxième moitié de la route 1. (Validez le résultat obtenu (aux arrondis près)).

```
SELECT      ST_AsText(ST_Line_Substring(the_geom, 1/2::real, 1::real)) as route
FROM        ROUTES
WHERE       ID = 1;

                    route
-----------------------------------------------------
LINESTRING(7.81684335937158 5.71827819602086,12 8,12 5)
(1 row)
```

Validation :

(1 2) (12 8) (12 5)

11 * 11 + 6 * 6 = 157 RacineCarrée (157) + 3 = 15,52

Lg = 7,76

Coordonnées (7,8 ; 5,7)

6,8 * 6,8 + 3, 7 * 3,7 = 59,93 RacineCarrée (59,93) = 7,74

2.16 Gestion de la topologie : agrégation (Donnez les coordonnées du rectangle englobant tous les magasins)

```
SELECT      ST_XMin(ST_Extent(the_geom)) as xmin,
            ST_XMax(ST_Extent(the_geom)) as xmax,
            ST_YMin(ST_Extent(the_geom)) as ymin,
            ST_YMax(ST_Extent(the_geom)) as ymax
FROM        BATIMENTS
WHERE       NOM = 'Magasin';
 xmin | xmax | ymin | ymax
------+------+------+------
    0 |    3 |    1 |   11
(1 row)
```

3 Gestion des données logiques

3.1 Prédicat intersection (Donnez les routes qui traversent la ville 1)

```
SELECT      R.ID
FROM        VILLES V, ROUTES R
WHERE       ST_Intersects (R.the_geom, V.the_geom) and
            V.ID = 1;
 id
----
 1
(1 row)
```

3.2 Autojointure spatiale (Donnez les routes qui se croisent)

```
SELECT      R1.ID, R2.ID
FROM        ROUTES R1, ROUTES R2
WHERE       ST_Intersects (R1.the_geom, R2.the_geom) and
            R1.ID < R2.ID;
 id | id
----+----
  1 |  2
(1 row)
```

3.3 Prédicat inclusion (Donnez les identifiant et nom des bâtiments dans la ville 2)

```
SELECT      B.ID, B.NOM
FROM        BATIMENTS B, VILLES V
WHERE       V.ID = 2 and
            ST_WithIn (B.the_geom, V.the_geom);

 id |  nom
----+--------
  1 | Mairie
  2 | Ecole
(2 rows)
```

```
SELECT      B.ID, B.NOM
FROM        BATIMENTS B, VILLES V
WHERE       V.ID = 2 and
            ST_Contains (V.the_geom, B.the_geom);
```

3.4 Prédicat adjacence (Donnez les 'Mairie' ou 'Radio' installées en bordure de villes.)

```
SELECT      B.ID as batiment
FROM        BATIMENTS B, VILLES V
WHERE       (B.NOM = 'Mairie' or B.NOM = 'Radio') and
```

```
                ST_Contains (V.the_geom, B.the_geom)  and
                not (ST_ContainsProperly (V.the_geom, B.the_geom));

 batiment
----------
        1
(1 row)
```

4 Analyse spatiale

4.1 Distance (Donnez la route la plus proche du bâtiment 'Mairie')

```
SELECT      R.ID
FROM        ROUTES R, BATIMENTS B
WHERE       B.NOM = 'Mairie' and
            ST_Distance (R.the_geom, B.the_geom) =
                (SELECT     MIN (ST_Distance (R.the_geom, B.the_geom))
                FROM        ROUTES R, BATIMENTS B
                WHERE       B.NOM = 'Mairie');

 id
----
  1
(1 row)
```

4.2 Inclusion (Donnez les villes contenant un magasin.)

```
SELECT      DISTINCT  V.ID, V.NOM
FROM        VILLES V, BATIMENTS B
WHERE       B.NOM = 'Magasin' and
            ST_Contains (V.the_geom, B.the_geom);

 id | nom
----+-----
  1 | V1
(1 rows)

SELECT      DISTINCT  V.ID, V.NOM
FROM        VILLES V, BATIMENTS B
WHERE       B.NOM = 'Magasin' and
            ST_ContainsProperly (V.the_geom, B.the_geom);
```

4.3 Agrégat (Donnez pour chaque route la longueur de sa partie urbaine)

```
SELECT      R.ID, sum (ST_length (ST_Intersection(R.the_geom, V.the_geom)))
FROM        VILLES V, ROUTES  R
WHERE       ST_Intersects (R.the_geom, V.the_geom)
GROUP BY    R.ID
UNION
SELECT      R.ID, 0
```

```
FROM        VILLES  V, ROUTES R
WHERE       not exists (    SELECT      *
                            FROM        VILLES V
                            WHERE       ST_Intersects (R.the_geom, V.the_geom))
```

```
 id |       longueur
----+------------------
  3 |              0
  1 | 5.69543822097349
  2 |              0
(3 rows)
```

4.4 Buffer (Quelles sont les villes qui ne sont pas impactees par le bruit d'une route (i.e., à plus de 3 unités d'une route) ?)

```
SELECT      V.ID, V.NOM
FROM        VILLES V
WHERE       not exists (    SELECT      *
                            FROM        ROUTES R
                            WHERE       ST_Intersects (ST_Buffer(R.the_geom, 3.0),
                                                       V.the_geom));
```

```
 id | nom
----+-----
  3 | V3
(1 row)
```

4.5 Inter-visibilité (Le bâtiment de la radio est-il visible de l'école ?)

```
    SELECT      B1.ID as ecole, B2.ID as radio, B.ID as batiment, 'NON' as reponse
    FROM        BATIMENTS B1, BATIMENTS B2, BATIMENTS B
    WHERE       ST_Intersects (ST_MakeLine(B1.the_geom, B2.the_geom), B.the_geom) and
                B1.NOM = 'Ecole' and B2.NOM = 'Radio' and
                B.ID <> B1.ID and B.ID <> B2.ID
UNION
    SELECT      B1.ID, B2.ID, 0, 'OUI' as reponse
    FROM        BATIMENTS B1, BATIMENTS B2
```

```
    WHERE       not exists (    SELECT      *
                                FROM        BATIMENTS B
                                WHERE       ST_Intersects (ST_MakeLine(B1.the_geom,
                                                                       B2.the_geom),
                                                           B.the_geom) and
                                            B.ID <> B1.ID and B.ID <> B2.ID) and
                    B1.NOM = 'Ecole' and B2.NOM = 'Radio';

     ecole | radio | batiment | reponse
    -------+-------+-----------+---------
        2  |    5  |       1     | NON
    (1 row)
```

4.6 Inter-visibilité (Le bâtiment de la radio est-il visible du bâtiment de la TV ?)

```
    SELECT      B1.ID as TV, B2.ID as Radio, B.ID as batiment, 'NON' as reponse
    FROM        BATIMENTS B1, BATIMENTS B2, BATIMENTS B
    WHERE       ST_Intersects (ST_MakeLine(B1.the_geom, B2.the_geom), B.the_geom) and
                B1.NOM = 'TV' and B2.NOM = 'Radio' and
                B.ID <> B1.ID and B.ID <> B2.ID
UNION
    SELECT      B1.ID, B2.ID, -1, 'OUI' as reponse
    FROM        BATIMENTS B1, BATIMENTS B2
    WHERE       not exists (    SELECT      *
                                FROM        BATIMENTS B
                                WHERE       ST_Intersects (ST_MakeLine(B1.the_geom,
                                                                       B2.the_geom),
                                                           B.the_geom) and
                                            B.ID <> B1.ID and B.ID <> B2.ID) and
                    B1.NOM = 'TV' and B2.NOM = 'Radio';


     tv  | radio | batiment | reponse
    ----+-------+-----------+---------
      6 |    5  |      -1     | OUI
    (1 row)



    SELECT      B1.ID as TV, B2.ID as Radio, B.ID as batiment, 'NON' as reponse
    FROM        BATIMENTS B1, BATIMENTS B2, BATIMENTS B
    WHERE       ST_Intersects (ST_MakeLine(B1.the_geom, B2.the_geom), B.the_geom) and
                B1.NOM = 'TV' and B2.NOM = 'Radio' and
                B.ID <> B1.ID and B.ID <> B2.ID
UNION
    SELECT      B1.ID, B2.ID, -1, 'OUI' as reponse
    FROM        BATIMENTS B1, BATIMENTS B2
    WHERE       not exists (    SELECT      *
                                FROM        BATIMENTS B
                                WHERE       ST_Intersects (ST_MakeLine(B1.the_geom,
                                                                       B2.the_geom),
                                                           B.the_geom) and
```

```
                                                    B.ID <> B1.ID and B.ID <> B2.ID) and
                        B1.NOM = 'TV' and B2.NOM = 'Radio';


 tv | radio | batiment | reponse
----+-------+----------+---------
  6 |     5 |       -1 | OUI
(1 row)
```

4.7 Gestion d'un opérateur

Créer une relation urbaine qui contient les parties urbaines des routes avec leur nom: 1 tuple par intersection route/ville
Visualisez le résultat sous QGIS

```
CREATE TABLE URBAINE (ID serial, NOM text, the_geom GEOMETRY) with oids;

INSERT INTO URBAINE (NOM, the_geom)
                SELECT      R.NOM, ST_Intersection (R.the_geom, V.the_geom)
                FROM        ROUTES  R, VILLES V
                WHERE       ST_Intersects (R.the_geom, V.the_geom);

DROP TABLE URBAINE;
```

5 Composition d'opérateurs

5.1 Approche fonctionnelle

Créez et chargez les relations. A partir de ces tuples effectuez les opérations permettant de calculer :

- Dans AIB : l'intersection de A et de B avec les identifiants associés
- Dans CIAIB : l'intersection de C avec l'intersection de A et de B en une seule requête.
- Dans DIAIB : l'intersection de D avec l'intersection de A et de B en une seule requête.

Schémas des relations :

```
CREATE TABLE A (ID INTEGER, the_geom GEOMETRY);
CREATE TABLE B ( ID INTEGER, the_geom GEOMETRY);
CREATE TABLE C ( ID INTEGER, the_geom GEOMETRY);
CREATE TABLE D ( ID INTEGER, the_geom GEOMETRY);
CREATE TABLE AIB (AID INTEGER, BID INTEGER, the_geom GEOMETRY);
CREATE TABLE CIAIB ( AID INTEGER, BID INTEGER,
                                CID INTEGER, the_geom GEOMETRY);
CREATE TABLE DIAIB ( AID INTEGER, BID INTEGER,
                                DID INTEGER, the_geom GEOMETRY);
```

Insertions des tuples :

```
INSERT INTO A VALUES
        (10, ST_GeometryFromText ('POLYGON ((1 1, 1 7, 3 7, 3 1, 1 1))',-1));
INSERT INTO B VALUES
        (20, ST_GeometryFromText ('POLYGON ((1 2, 1 8, 3 8, 3 2, 1 2))',-1));
```

```
INSERT INTO C VALUES
        (30, ST_GeometryFromText ('POLYGON ((2 5, 2 6, 4 6, 4 5, 2 5))',-1));
INSERT INTO D VALUES
        (40, ST_GeometryFromText ('POLYGON ((2 3, 2 4, 4 4, 4 3, 2 3))',-1));

INSERT INTO A VALUES
        (11, ST_GeometryFromText ('POLYGON ((5 1, 5 7, 7 7, 7 1, 5 1))',-1));
INSERT INTO B VALUES
        (21, ST_GeometryFromText ('POLYGON ((5 2, 5 8, 7 8, 7 2, 5 2))',-1));
INSERT INTO C VALUES
        (31, ST_GeometryFromText ('POLYGON ((6 5, 6 6, 8 6, 8 5, 6 5))',-1));
-- pas de D

INSERT INTO A VALUES
        (12, ST_GeometryFromText ('POLYGON ((9 1, 9 7, 11 7, 11 1, 9 1))',-1));
INSERT INTO B VALUES
        (22, ST_GeometryFromText ('POLYGON ((9 2, 9 8, 11 8, 11 2, 9 2))',-1));
-- pas de C
INSERT INTO D VALUES
        (42, ST_GeometryFromText ('POLYGON ((10 3, 10 4, 12 4, 12 3, 10 3))',-1));
```

Résultats attendus :

```
INSERT INTO AIB
        SELECT      A.ID, B.ID, ST_Intersection (A.the_geom, B.the_geom)
        FROM        A, B
        WHERE       ST_intersects (A.the_geom, B.the_geom);

SELECT      * as interab
FROM        AIB;
```

a	b	interab
10	20	POLYGON((1 2,1 7,3 7,3 2,1 2))
11	21	POLYGON((5 2,5 7,7 7,7 2,5 2))
12	22	POLYGON((9 2,9 7,11 7,11 2,9 2))
(3 rows)		

```
INSERT INTO CIAIB
        SELECT      A.ID as a, B.ID as b, C.ID as c,
                    ST_Intersection (C.the_geom, ST_Intersection (A.the_geom, B.the_geom))
        FROM        A, B, C
        WHERE       ST_Intersects (A.the_geom, B.the_geom) and
                    ST_Intersects (C.the_geom, ST_Intersection (A.the_geom, B.the_geom));

SELECT      * as intercinterab
FROM        CIAIB;
```

a	b	c	intercinterab

10	20	30	POLYGON((2 5,2 6,3 6,3 5,2 5))
11	21	31	POLYGON((6 5,6 6,7 6,7 5,6 5))
(2 rows)			

```
INSERT INTO DIAIB
        SELECT      A.ID, B.ID, D.ID,
                    ST_Intersection (D.the_geom, ST_Intersection (A.the_geom, B.the_geom))
        FROM        A, B, D
        WHERE       ST_Intersects (A.the_geom, B.the_geom) and
                    ST_Intersects (D.the_geom, ST_Intersection (A.the_geom, B.the_geom));

SELECT      A, B, D, ST_AsText(the_geom) as interdinterab
FROM        DIAIB;
```

a	b	d	interdinterab
10	20	40	POLYGON((2 3,2 4,3 4,3 3,2 3))
12	22	42	POLYGON((10 3,10 4,11 4,11 3,10 3))
(2 rows)			

```
DROP TABLE A;
DROP TABLE B;
DROP TABLE C;
DROP TABLE D;

DROP TABLE AIB;
DROP TABLE CIAIB;
DROP TABLE DIAIB;
```