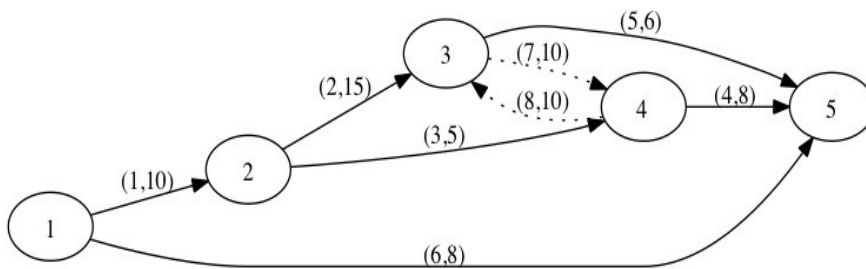


# Manipulation d'une base de données PostgreSQL avec la récursion - Corrigé

## 1 SQL récursif :

A partir de la base tp et de la relation reseau (id int4, origine int4, destination int4, cout int4) :



### 1.1 Accessibilité

Q1 : Déterminez tous les nœuds accessibles à partir de 2

```
n
---
```

3  
4  
5  
(3 rows)

```
With recursive deep (noeud) as (
  Select  destination
  From    reseau
  Where   origine = 2
union
  Select  destination
  From    reseau, deep
  Where   reseau.origine = deep.noeud )
Select *
From    deep;
```

Q2 : Déterminez tous les nœuds permettant d'accéder à 4

```
n
---
```

2  
1  
(2 rows)

```
With recursive deep (noeud) as (
  Select  origine
  From    reseau
  Where   destination = 4
```

```

union
  Select  origine
  From    reseau, deep
  Where   reseau.destination = deep.noetid and
         reseau.destination <> 4)
Select *
From    deep;

```

Ajouter les arcs suivants dans le relation reseau :

id = 7, Origine = 3, Destination = 4, Cout = 10

id = 8, Origine = 4, Destination = 3, Cout = 10

```
insert into reseau values (7, 3,4,10);
```

```
insert into reseau values (8, 4,3,10);
```

Q3 : Déterminez les noeuds accessibles à partir de 1 avec la longueur du chemin sous réserve que le nombre d'arc soit inférieur ou égal à 2 (en utilisant le SQL récursif bien évidemment !).

```

noetid | longueur
-----+-----
  2    |    1
  5    |    1
  3    |    2
  4    |    2
(4 rows)

```

With recursive deep (noetid, longueur) as (

```
  Select  destination, 1
```

```
  From    reseau
```

```
  Where   origine = 1
```

union

```
  Select  destination, longueur+1
```

```
  From    reseau, deep
```

```
  Where   reseau.origine = deep.noetid and
         deep.longueur < 2 )
```

```
Select *
```

```
From    deep
```

```
order by longueur;
```

Q4 : Déterminez les noeuds accessibles sous réserve que le nombre d'arc soit inférieur ou égal à 2 (en utilisant le SQL récursif bien évidemment !) à partir de 1 avec la longueur minimum du chemin pour l'accessibilité triés par ordre de longueur minimum puis après avec la longueur maximum.

```

noetid | min
-----+-----
  5    |    1
  2    |    1
  4    |    2
  3    |    2
(4 rows)

```

```

noetid | max
-----+-----
  2    |    1
  4    |    3
  5    |    3
  3    |    3
(4 rows)

```

```

With recursive deep (noeud, longueur) as (
  Select destination, 1
  From reseau
  Where origine = 1
union
  Select destination, longueur+1
  From reseau, deep
  Where reseau.origine = deep.noeud and
        deep.longueur < 3
)
Select noeud, min (longueur)
From deep
Group by noeud
Order by 2;

```

```

With recursive deep (noeud, longueur) as (
  Select destination, 1
  From reseau
  Where origine = 1
union
  Select destination, longueur+1
  From reseau, deep
  Where reseau.origine = deep.noeud and
        deep.longueur < 3
)
Select noeud, max (longueur)
From deep
Group by noeud
Order by 2;

```

Q5 : Déterminez les nœuds accessibles pour un coût inférieur à 24 à partir de 1 et donnez le coût minimum.

noeud	min
4	15
5	8
2	10

(3 rows)

```

With recursive deep (noeud, cout) as (
  Select destination, cout
  From reseau
  Where origine = 1
union
  Select destination, deep.cout+reseau.cout
  From reseau, deep
  Where reseau.origine = deep.noeud and
        deep.cout + reseau.cout < 24
)
Select noeud, min (cout)
From deep
group by noeud;

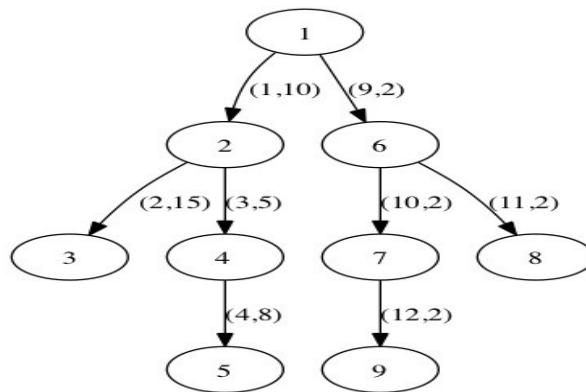
```

Supprimez les arcs 5, 6, 7 et 8, ajoutez les arcs :  
 id = 9, Origine = 1, Destination = 6, Cout = 2  
 id = 10, Origine = 6, Destination = 7, Cout = 2  
 id = 11, Origine = 6, Destination = 8, Cout = 2  
 id = 12, Origine = 7, Destination = 9, Cout = 2

delete from reseau where id > 4;  
 insert into reseau values (9, 1,6,2);  
 insert into reseau values (10, 6,7,2);  
 insert into reseau values (11, 6,8,2);  
 insert into reseau values (12, 7,9,2);

## 1.2 Même génération

Q6 : Donnez les cousins de même génération et le numéro de la génération en commençant à compter les générations à partir de 1, présentés dans l'ordre croissant des générations.



noeud	generation
3	2
4	2
7	2
8	2
5	3
9	3

(6 rows)

With recursive deep (noeud, longueur) as (  
 Select destination, 1  
 From reseau  
 Where origine = 1  
 union  
 Select destination, longueur+1  
 From reseau, deep  
 Where reseau.origine = deep.noeud)  
 Select noeud, longueur as generation  
 From deep  
 Where longueur >= 2  
 order by longueur;

## 2 Manipulation de PgRouting

### 2.1 Chaîne de coût minimal entre le noeud 1 et 88

```
Select *
From shortest_path ('
                    Select gid as id, source::integer, target::integer,
                        length::double precision as cost
                    From kanagawa',
                    1, 88, false, false);
```

vertex_id	edge_id	cost
1	2	0.00438339917408409
3	84	0.00120369903133505
81	68	0.000729434757423326
80	91	0.000978074036916293
87	74	0.00058531686045292
88	-1	0

(6 rows)

### 2.2 Longueur de la chaîne de coût minimal entre les nœuds 1 et 88

```
Select count (*) as longueur
From shortest_path ('
                    Select gid as id, source::integer, target::integer,
                        length::double precision as cost
                    From kanagawa',
                    1, 88, false, false);
```

longueur

```
-----
6
(1 row)
```

### 2.3 Cout total de la chaîne de coût minimal entre les nœuds 1 et 88

```
Select sum(cost) as cout
From shortest_path ('
                    Select gid as id, source::integer, target::integer,
                        length::double precision as cost
                    From kanagawa',
                    1, 88, false, false);
```

```

      cout
-----
0.00787992386021168
(1 row)

```

## 2.4 Coût total du chemin de coût minimal

```

Select      vertex_id, cost as cout
From        shortest_path ('
                Select      gid as id, source::integer, target::integer,
                length::double precision as cost
                From        kanagawa',
                1, 88, true, false);

```

```

vertex_id |      cout
-----+-----
      1   | 0.00760092142093486
      2   | 0.00216442157577254
      5   | 0.00138529047740129
      88  |                0
(4 rows)

```

```

Select      sum(cost) as cout
From        shortest_path ('
                Select      gid as id, source::integer, target::integer,
                length::double precision as cost
                From        kanagawa',
                1, 88, true, false);

```

```

      cout
-----
0.0111506334741087
(1 row)

```

## 2.5 Déterminez la longueur de la chaîne entre le nœud 1 et le nœud le plus éloigné

```

distancemaximum
-----
0.598455513354199
(1 row)

```

On calcule la distance maximum (équivalent à la longueur d'un chemin hamiltonien). On utilise cette valeur comme constante dans le calcul de distance max d'un isoline.

```
Select sum (length) as coutMaximumPossible  
From kanagawa;
```

```
    coutmaximumpossible  
-----  
218.449895440810127808467  
(1 row)
```

```
Select max (cost) as distanceMaximum  
From driving_distance (  
    'Select gid as id,  
        source::integer,  
        target::integer,  
        length::double precision as cost  
    From kanagawa',  
    1, 219,  
    true, false  
);
```

## **2.6 Déterminez le numéro du nœud le plus éloigné du nœud 1**

```
noeudlepluseloigne  
-----  
41749  
(1 row)
```

```
Select vertex_id as noeudLePlusEloigne  
From driving_distance (  
    'Select gid as id,  
        source::integer,  
        target::integer,  
        length::double precision as cost  
    From kanagawa',  
    1, 0.60,  
    true, false  
)  
Where cost = (Select max (cost)  
    From driving_distance ('Select gid as id,  
        source::integer, target::integer,  
        length::double precision as cost  
    From kanagawa', 1, 0.60, true, false  
    )  
);
```

## **2.7 Déterminez le nombre d'arcs du chemin pour aller à ce nœud à partir du nœud 1**

nombrearcs

```
-----  
558  
(1 row)
```

```
Select count (edge_id) as nombreArcs  
From shortest_path (  
    Select      gid as id, source::integer, target::integer,  
                length::double precision as cost  
    From        kanagawa',  
    1, 41749, true, false);
```

## **2.8 Déterminez le nombre de nœuds accessibles à partir du nœud 1 pour un coût inférieur à 0,02 et donnez ces nœuds**

```
Select *  
From driving_distance (  
    'Select gid as id,  
        source::integer,  
        target::integer,  
        length::double precision as cost  
    From kanagawa',  
    1, 0.02,  
    true, false  
    )  
Where cost <> 0;
```

```
Select count (*) as nombreNoeud  
From driving_distance (  
    'Select gid as id,  
        source::integer,  
        target::integer,  
        length::double precision as cost  
    From kanagawa',  
    1, 0.02,  
    true, false  
    )  
Where cost <> 0;
```

nombrenoeud

```
-----  
16  
(1 row)
```



vertex_id	edge_id	cost
2	5	0.00760092142093486
3	84	0.0117701131933485
5	4	0.0097653429967074
6	67	0.00872336061888513
7	6	0.012874508129403
8	103	0.0177799232661332
9	7	0.0131352266581606
14	11	0.0145761700025048
22	105	0.019619112441012
80	91	0.00983697940459011
81	84	0.0105664141620134
87	91	0.0108150534415064
88	74	0.0111506334741087
110	107	0.000399427058805463
2317	3143	0.0160403864289264
2697	4106	0.0160866011719174

(16 rows)