Eric THIERRY M2CCI

TP2 Base de données
Agence de voyage

14/12/16

Requête 9 : Donner la liste des numéros des circuits qui passent dans toutes les villes d'un pays donné.

Schéma et spécification :

R9 (numC)<x> ∈ R9 ⇔ Le circuit de numéro x passe par toutes les villes du pays donné.

Requête SQL :

```
ACCEPT nomPays PROMPT 'Donner un nom de pays (ex : France) : ';
SELECT numC
FROM (SELECT numC, COUNT(ville) AS villeVisitee
      FROM (SELECT nomV AS ville
            FROM AGENCE.LesVilles
            WHERE pays =&nomPays)A
/*A(ville)<x> ∈ A ⇔ La ville x est une ville du pays donné*/
      NATURAL JOIN (SELECT numC, vDep AS ville
                    FROM AGENCE.LesCircuits
                    union
                    SELECT numC, vArr
                    FROM AGENCE.LesCircuits
                    union
                    SELECT numC, vEtape
                    FROM AGENCE.LesEtapes)B
/*B(numC, ville)<x,y> ∈ B ⇔ La ville y fait partie du circuit x*/
      GROUP BY numC)C
/*C(numC, villeVisitee)<x,y> ∈ C ⇔ Le circuit de numéro x visite y ville du
pays donné*/
JOIN (SELECT COUNT(nomV) AS nbville
        FROM AGENCE.LesVilles
        WHERE pays = &nomPays)D
/*D(nbVille)<x> ∈ à D ⇔ x est le nombre de ville du pays donné*/
on (nbville = villeVisitee);
```

Résultats et tests de la requête SQL et des sous-requêtes :

Résultat de la requête R9 avec Norvège comme pays donné :

R9 (numC)<x> ∈ R9 ⇔ Le circuit de numéro x passe par toutes les villes du pays donné.

```
Donner un nom de pays (ex : France) : Norvege

     NUMC
----------
       16
       21
```

Tests de la requête R9 :

Pour les tests on suppose que le que l'utilisateur rentre un pays P.

1) Vérifier que le circuit numéro X passe par toutes les villes du pays P. Le numéro X doit sortir dans les résultats de la requête.

```
Donner un nom de pays (ex : France) : Norvege
      NUMC
----------
        16
        21
```

Le test est passant car les circuits 16 et 21 passent bien par toutes les villes de Norvège.

2) Vérifier que le circuit X passe par toutes les villes du pays P mais aussi par des villes d'un ou plusieurs autres pays. Le numéro X doit sortir dans les résultats de la requête.

```
Donner un nom de pays (ex : France) : Norvege
      NUMC
----------
        16
        21
```

Le test est passant car le circuit 16 passe par une ville de Finlande et toutes les villes de Norvège. Et le circuit 21 passe par une ville de Finlande, toutes les villes de Norvège et quelques villes d'Italie.

3) Vérifier que le circuit X passe par certaines villes du pays P. Le numéro X ne doit pas sortir dans les résultats de la requête.

```
Donner un nom de pays (ex : France) : Italie
no rows selected
```

Le test est passant, le circuit 21 vu précédemment passe par une partie des villes d'Italie et ne sort donc pas dans les résultats.

4) Vérifier que le circuit X a pour ville de départ ou d'arrivée une ville du pays P et passe par toutes les autres villes du pays P. Le numéro X doit sortir dans les résultats de la requête.

   Ce test est sortant car les tables de départ ne permettent pas de le vérifier. Afin qu'il soit passant on doit ajouter les villes d'Islande manquantes au circuit numéro 14. Et ainsi ce numéro de circuit figurerait dans les résultats.

   Remarque : le script de peuplement que vous nous proposez nous a été mis à disposition seulement le vendredi 9 décembre. Lors du lancement du script il y a un message d'erreur pour chaque commande d'insertion dans les tables. Comme je n'ai pas eu de retour de Mme Fauvet à ce sujet. Pour les tests nécessitant une modification des tables, j'indiquerai juste les données à insérer ou supprimer comme je l'ai fait pour le test précédent.

Résultat de la sous-requête A avec Norvège comme pays donné :

A(ville)<x> ∈ A ⇔ La ville x est une ville du pays donné

```
VILLE
-------------
Cap Nord
Hammerfest
Inari
Saarijarvi
Salla
Tornio
Ylivieska
```

Tests de la sous-requête A :

1) Vérifier que toutes les villes du pays P sortent dans les résultats de la requête.

   ```
   Donner un nom de pays (ex : France) : Norvege
   VILLE
   -------------
   Cap Nord
   Hammerfest
   Inari
   Saarijarvi
   Salla
   Tornio
   Ylivieska
   ```

   Le test est passant toutes les villes ne Norvège sont présente dans les résultats.

2) Vérifier que l'ajout ou la suppression d'une ville du pays P, modifie bien la liste des villes sortant pour le pays P.

Après ajout de la ville d'Oslo dans la table les villes, elle apparaitrait dans les résultats de la requête en 1)

Après suppression de la ville du Cap Nord de la table les villes, elle ne figurerait plus dans la requête présentée précédemment.

Résultat de la sous-requête B:

B(numC, ville)<x,y> ∈ B ⇔ La ville y fait partie du circuit x

NUMC	VILLE
1	Londres
1	Paris
2	Londres
2	Paris
3	Amsterdam
3	Paris
4	Amsterdam
4	Paris
5	Lyon
5	Venise
6	Paris
6	Venise
7	Hoedic
7	Houat
7	Port Cotton
7	Port Maria
7	Quiberon
8	Besse
8	Clermont Ferrand
8	Laschamps
8	Orcival
9	Bath
9	Exeter
9	Londres
9	Paris
9	Salisbury
9	Sissinghurst
9	St Ives
10	Bantry
10	Cork
10	Dingle
10	Glengariff
10	Kenmare
10	Killarney
10	Shannon
11	Cleggan
11	Clifden
11	Ennis
11	Galway
11	Leenan
11	Louisburgh
11	Shannon
11	Westport
12	Ardara
12	Carrick
12	Donegal
12	Dublin
12	Dungloe

12 Killibegs
12 Letterkenny
13 Christianshab
13 Copenhague
13 Disko
13 Egedesmine
13 Godhavn
13 Holsteinborg
13 Jakobshavn
13 Paris
13 Sondre
14 Akureyri
14 Asbyrgi
14 Geysir
14 Gullfoss
14 Hveravellir
14 Myvatn
14 Reykjavik
15 Godafoss
15 Husavik
15 Paris
15 Reykholt
15 Reykjavik
16 Cap Nord
16 Hammerfest
16 Helsinky
16 Inari
16 Paris
16 Saarijarvi
16 Salla
16 Tornio
16 Ylivieska
17 Castelo de Vide
17 Evora
17 Faro
17 Fatima
17 Leiria
17 Lisbonne
17 Porto
17 Urgeirica
17 Vila Real
18 Florence
18 Paris
18 Ravenne
18 Rome
18 Venise
18 Verone
19 Bari
19 Benevento
19 Brindisi
19 Capri
19 Caserte
19 Catanzaro
19 Cosenza
19 Lecce
19 Metaponto
19 Naples
19 Pompei
19 Rome
19 Salerne
20 Briancon
21 Cap Nord
21 Florence
21 Hammerfest
21 Helsinky
21 Inari

```
        21 Paris
        21 Ravenne
        21 Rome
        21 Saarijarvi
        21 Salla
        21 Tornio
        21 Venise
        21 Verone
        21 Ylivieska
```

1) Vérifier que toutes les villes du circuit X sortent dans les résultats de la requête.

```
      NUMC VILLE
---------- -------------
        17 Castelo de Vide
        17 Evora
        17 Faro
        17 Fatima
        17 Leiria
        17 Lisbonne
        17 Porto
        17 Urgeirica
        17 Vila Real
```

Le test est passant toutes les villes du circuit 17 sont présente dans les résultats.

2) Vérifier que l'ajout ou la suppression d'une ville du circuit numéro X, modifie bien la liste des villes sortant pour le circuit numéro X.

Ajout de la ville Lyon, au circuit 17 : donnerait le même résultat que la requête en 1) avec la présence de Lyon.
Suppression de la ville Evora, au circuit 17 : donnerait le même résultat que la requête en 1) aves l'absence d'Evora

Résultat de la sous-requête C avec Norvège comme pays donné :

C(numC, villeVisitee)<x,y> ∈ C ⇔ Le circuit de numéro x visite y ville du pays donné

```
      NUMC VILLEVISITEE
---------- ------------
        16            7
        21            7
```

Tests de la sous-requête C :

1) Vérifier que le circuit X visite V villes du pays P. Sort dans la table les circuits X avec V villes correspondantes au pays P.

```
Donner un nom de pays (ex : France) : Italie
      NUMC VILLEVISITEE
---------- ------------
         6            1
         5            1
        18            5
        19           13
        21            5
```

Le test est passant, un certain nombre de circuit passe par une ou plusieurs villes italiennes.

2) Vérifier que le circuit X n'ayant pas de ville visités pour le pays P ne sorte pas dans les résultats de la requête.

```
Donner un nom de pays (ex : France) : Italie
      NUMC VILLEVISITEE
---------- ------------
         6            1
         5            1
        18            5
        19           13
        21            5
```

Le test est passant, le circuit 1 ne passe par aucune ville italienne et donc il ne figure pas dans les résultats.

3) Vérifier que l'ajout ou la suppression d'une ville du pays P dans le circuit X modifie ne nombre de ville visité.
Ajouter la ville italienne 'Vérone' au circuit 6 : donnerait 2 villes visitées.
Supprimer la ville italienne 'Vérone' au circuit 18 : donnerait 4 villes visitées.

Résultat de la sous-requête D avec Norvège comme pays donné :

D(nbVille)<x> ∈ D ⇔ x est le nombre de ville du pays donné

```
Donner un nom de pays (ex : France) : Norvege
   NBVILLE
----------
         7
```

Tests de la sous-requête D :

1) Vérifier que le nombre de ville du pays P est bien le total des villes de ce pays.

```
Donner un nom de pays (ex : France) : Norvege
   NBVILLE
----------
         7
```

Test passant, le résultat affiche 7 correspondant au total des villes de Norvège.

2) Vérifier que l'ajout ou la suppression d'une ville du pays P modifie le total de ville pour ce pays.
Ajout de la ville Oslo à la table LesVilles : le résultat précédent affichera alors 8.
Suppression de la ville Evora de la tables LesVilles : le résultat précédent affichera 6.

Requête 11 : Donner le numéro, le prix de base (sans tenir compte du prix des monuments visités), la date de départ et le nombre de places disponibles et dont le nombre de jours est inférieur ou égale à un entier donné.

Schéma et spécification de la requête :

R11(numC, dateDep, prix, Dispo)<w,x,y,z> ∈ R11 ⇔ le circuit de numéro *w* a pour date de départ *y*. Son prix est *x* et il reste *z* places disponibles.

Requête SQL :

```
ACCEPT choixJour PROMPT 'Choisissez le nombre de jours pour votre voyage :';
SELECT numC, dateDep, prix, Dispo
FROM (SELECT numC, dateDep, prix, (nbplaces - nvl(nbReserve, 0)) AS Dispo
      FROM AGENCE.LesProgrammations NATURAL LEFT OUTER JOIN
     (SELECT numC, datedep, SUM(nbRes) AS nbReserve FROM
AGENCE.LesReservations
      GROUP BY numC, datedep)A
/*A(numC, dateDep, nbReserve)<x,y,z> ∈ A ⇔ le circuit de numéro x a pour
date de départ y et possède z places de réservées*/
NATURAL JOIN AGENCE.LesCircuits
      WHERE (nbplaces - nvl(nbReserve, 0)) > 0)B
/*B(numC, dateDep, prix, Dispo)<w,x,y,z> ∈ B ⇔ le circuit de numéro w a
pour date de départ x. Son prix est y et il reste z places disponibles.
*/
 NATURAL JOIN (SELECT numC, SUM(nbjours) AS totaljour FROM AGENCE.LesEtapes
      GROUP BY numC
      HAVING SUM(nbjours) <= &choixJour)C
/*C(numC, totaljour)<y,z> ∈ C ⇔ Le circuit de numéro y dure z jours */
ORDER BY numC, dateDep;
```

Résultats et test de la requête SQL et des sous-requêtes :

Résultat de la requête R11 avec 4 comme entier entré :

R11(numC, dateDep, prix, Dispo)<w,x,y,z> ∈ R11 ⇔ le circuit de numéro *w* a pour date de départ *y*. Son prix est *x* et il reste *z* places disponibles.

```
Choisissez le nombre de jours pour votre voyage : 4
      NUMC DATEDEP         PRIX      DISPO
---------- --------- ---------- ----------
         1 04-JAN-10       1160         34
         1 04-FEB-10       1160          8
         1 06-FEB-10       1160         34
         1 24-JUL-10       1160         10
         2 06-JAN-10       1160          2
         2 07-JAN-10       1160          9
         2 06-FEB-10       1160         12
         2 05-SEP-10       1160         30
         3 24-DEC-09       1040         13
         3 31-DEC-09       1040         45
         3 03-JUL-10       1040          1
```

```
         4 30-JUN-10       1270         11
         4 06-AUG-10       1270         10
         4 31-AUG-10       1270          1
         4 06-NOV-10       1270         18
         6 06-FEB-10       2520         22
         6 06-SEP-10       2520         10
         6 06-OCT-10       2520         12
         6 16-NOV-10       2520         34
```

Test de la requête R11 :

On note E l'entier entré par l'utilisateur

1) Vérifier que le circuit numéro X a encore des places disponibles et a une durée inférieure ou égale à E. Le circuit numéro X doit sortir dans les résultats la requête.

```
Choisissez le nombre de jours pour votre voyage : 4
      NUMC DATEDEP         PRIX      DISPO  TOTALJOUR
---------- --------- ---------- ---------- ----------
         1 04-JAN-10       1160         34          2
         1 04-FEB-10       1160          8          2
```

   Le test est passant, exemple avec le circuit 1 pour les programmations du 04-JAN-10 et 04-FEB-10

2) Vérifier que le circuit X a encore des places disponibles et a une durée strictement supérieure à E. Le circuit numéro X ne doit pas sortir dans les résultats de la requête.

```
      NUMC DATEDEP         PRIX      DISPO  TOTALJOUR
---------- --------- ---------- ---------- ----------
         5 31-AUG-10       2740         64          5
         5 06-NOV-10       2740          3          5
```

   Ce test est passant, on voit que le circuit 5 a des places disponibles et se déroule sur 5 jours, et il est bien absent des résultats de la requête pour une durée maximum de 4 jours.

3) Vérifier que le circuit X n'a plus de places disponibles et a une durée inférieure ou égale à E. Le circuit numéro X ne doit pas sortir dans les résultats de la requête.

   Le test est passant pour E = 4, la programmation du 21-JUL-10 pour le circuit 1 a une durée de 2 jours et n'a plus de places disponibles. Cette ligne est bien absente du résultat de la requête.

4) Vérifier que le circuit X n'a plus de places disponibles et a une durée strictement supérieure à E. Le circuit numéro X ne doit pas sortir dans les résultats de la requête.

Le test est passant pour E = 4, la programmation du 14-FEB-10 pour le circuit 8 a une durée de 7 jours et n'a plus de places disponibles. Cette ligne est bien absente du résultat de la requête.

Résultat de la sous-requête A :

A(numC, dateDep, nbReserve)<x,y,z> ∈ A ⇔ le circuit de numéro *x* a pour date de départ *y* et possède *z* places de réservées

NUMC	DATEDEP	NBRESERVE
8	28-FEB-10	7
3	03-JUL-10	11
7	16-DEC-09	51
13	01-JAN-10	10
12	06-FEB-10	79
18	06-DEC-10	11
8	14-FEB-10	1
8	16-FEB-10	1
4	30-JUN-10	88
5	31-AUG-10	2
2	07-JAN-10	25
19	15-APR-10	20
5	06-NOV-10	43
21	15-JAN-10	24
13	31-DEC-09	48
14	26-JUL-10	21
20	27-JAN-10	5
1	21-JUL-10	10
2	05-FEB-10	99
1	04-FEB-10	4
10	01-JAN-10	2
9	30-OCT-10	10
10	11-FEB-10	2

Tests de la sous-requête A :

1) Vérifier qu'un circuit X ayant au moins une place réservée pour une programmation P sort dans les résultats de la requête.

NUMC	DATEDEP	NBRESERVE
8	28-FEB-10	7
3	03-JUL-10	11
7	16-DEC-09	51
13	01-JAN-10	10
12	06-FEB-10	79
18	06-DEC-10	11
8	14-FEB-10	1
8	16-FEB-10	1
4	30-JUN-10	88
5	31-AUG-10	2
2	07-JAN-10	25
19	15-APR-10	20
5	06-NOV-10	43
21	15-JAN-10	24
13	31-DEC-09	48
14	26-JUL-10	21
20	27-JAN-10	5

1	21-JUL-10	10
2	05-FEB-10	99
1	04-FEB-10	4
10	01-JAN-10	2
9	30-OCT-10	10
10	11-FEB-10	2

Le test est passant

2) Vérifier qu'un circuit X ayant aucune réservation pour une programmations P ne figure pas dans les résultats de la requête.

Le test set passant, la programmation du 04-JAN-10 pour le circuit 1 ne sort pas car il n'y a aucune réservation.

Résultat de la sous-requête B :

B(numC, dateDep, prix, Dispo)<w,x,y,z> ∈ B ⇔ le circuit de numéro *w* a pour date de départ *x*. Son prix est *y* et il reste *z* places disponibles.

NUMC	DATEDEP	PRIX	DISPO
1	24-JUL-10	1160	10
1	06-FEB-10	1160	34
1	04-FEB-10	1160	8
1	04-JAN-10	1160	34
2	05-SEP-10	1160	30
2	06-FEB-10	1160	12
2	07-JAN-10	1160	9
2	06-JAN-10	1160	2
3	03-JUL-10	1040	1
3	31-DEC-09	1040	45
3	24-DEC-09	1040	13
4	06-NOV-10	1270	18
4	31-AUG-10	1270	1
4	06-AUG-10	1270	10
4	30-JUN-10	1270	11
5	06-NOV-10	2740	3
5	31-AUG-10	2740	64
6	16-NOV-10	2520	34
6	06-OCT-10	2520	12
6	06-SEP-10	2520	10
6	06-FEB-10	2520	22
7	31-OCT-10	2500	39
7	31-AUG-10	2500	56
7	26-FEB-10	2500	1
7	06-JAN-10	2500	1
7	16-DEC-09	2500	1
8	14-NOV-10	2140	3
8	31-AUG-10	2140	10
8	21-JUL-10	2140	14
8	16-MAY-10	2140	12
8	26-APR-10	2140	31
8	28-FEB-10	2140	4
8	21-FEB-10	2140	13
8	16-FEB-10	2140	11
8	31-DEC-09	2140	10
8	24-DEC-09	2140	18
9	30-OCT-10	5700	21
9	06-FEB-10	5700	3

10	28-FEB-10	6170	22
10	11-FEB-10	6170	28
10	21-JAN-10	6170	45
10	01-JAN-10	6170	78
11	30-JUN-10	6270	12
11	29-MAY-10	6270	34
11	28-FEB-10	6270	3
11	06-FEB-10	6270	13
12	06-FEB-10	6170	111
13	31-DEC-10	18590	44
13	30-JUN-10	18590	52
13	31-MAY-10	18590	3
13	06-MAY-10	18590	60
13	30-APR-10	18590	15
13	06-FEB-10	18590	99
13	01-JAN-10	18590	58
13	31-DEC-09	18590	3
14	26-JUL-10	7700	4
14	06-JUL-10	7700	12
15	31-AUG-10	8560	18
15	06-FEB-10	8560	11
16	31-AUG-10	10180	17
16	06-FEB-10	10180	12
16	06-JAN-10	10180	3
17	31-DEC-10	3990	3
17	26-FEB-10	3990	34
17	16-FEB-10	3990	12
17	06-FEB-10	3990	3
18	06-DEC-10	6690	1
18	06-OCT-10	6690	40
18	30-JUL-10	6690	90
18	30-APR-10	6690	15
19	20-DEC-10	10400	11
19	10-DEC-10	10400	28
19	16-SEP-10	10400	10
19	06-SEP-10	10400	20
19	05-AUG-10	10400	12
19	15-APR-10	10400	25
19	06-FEB-10	10400	10
21	06-APR-10	15000	99
21	26-FEB-10	15000	12
21	25-JAN-10	15000	3
21	15-JAN-10	15000	42
20	03-APR-10	450	87
20	01-APR-10	450	13
20	06-FEB-10	450	1
20	02-FEB-10	450	14
20	20-JAN-10	450	12
20	22-DEC-09	450	18

Tests de la sous-requête B :

1) Vérifier que le circuit X avec la programmation P possède encore des places disponibles. La programmation P du circuit X doit sortir dans les résultats de la requête.

NUMC	DATEDEP	PRIX	DISPO
1	24-JUL-10	1160	10
1	06-FEB-10	1160	34
1	04-FEB-10	1160	8
1	04-JAN-10	1160	34

Le test est passant, les programmations pour le circuit 1 ayant encore des places disponibles sont présentes dans les résultats de la requête.

2) Vérifier que le circuit X avec la programmation P n'ayant plus de places disponibles ne sort pas dans les résultats de la requête.

   Le test est passant. La programmation du 21-JUL-10 pour le circuit 1 n'a plus de places disponibles. Cette ligne est bien absente du résultat de la requête.

Résultat de la sous-requête C :

C(numC, totaljour)<y,z> ∈ C ⇔ Le circuit de numéro y dure z jours

```
Choisissez le nombre de jours pour votre voyage : 4
      NUMC  TOTALJOUR
---------- ----------
         1          2
         2          2
         3          3
         4          4
         6          4
```

Tests de la sous-requête C :

1) Vérifier que le circuit X ayant une durée inférieure ou égale à l'entier E sort dans les résultats de la requête.

```
Choisissez le nombre de jours pour votre voyage : 4
      NUMC  TOTALJOUR
---------- ----------
         1          2
         2          2
         3          3
         4          4
         6          4
```

   Le test est passant, les circuits 1, 2, 3, 4, 6 ont une durée inférieur ou égale à 4.

2) Vérifier que le circuit X ayant une durée supérieure à l'entier E ne sort pas dans les résultats de la requête.

   Ce test est passant pour E = 4, le circuit numéro 5 n'apparait pas dans les résultats car sa durée est de 5 jours.

Requête 13 : Pour chaque programmation de circuit, retrouver les pays dans lequel passe le circuit et la date à laquelle le circuit arrive dans ce pays

Schéma et spécification de la requête :

R13(numC, dateDep, pays, dateEntree)<w,x,y,z> ∈ R13 ⇔ la programmation de la date x du circuit w arrive dans le pays y à la date z.

Requête SQL :

```
WITH X AS (
      SELECT numC, rang, pays
      FROM (
      SELECT numC, rang, vEtape AS ville
      FROM AGENCE.LesEtapes
      UNION
      SELECT numC, (rmin-1) AS rang, vDep
      FROM AGENCE.LesCircuits
      NATURAL JOIN ( SELECT numC, min(rang) AS rmin
                     FROM AGENCE.LesEtapes
                     GROUP BY numC )
      UNION
      SELECT numC, (rmax+1) AS rang, vArr
      FROM AGENCE.LesCircuits
      NATURAL JOIN ( SELECT numC, max(rang) AS rmax
                     FROM AGENCE.LesEtapes
                     GROUP BY numC ) )
      JOIN AGENCE.LesVilles ON ( ville=nomV )
      ),
/*X(numC, rang, pays)<x,y,z> ∈ X ⇔ Le numéro de circuit x possède une ville
de rang y ayant pour pays z*/
     Y AS (
      SELECT numC, rang, vEtape AS ville, nbJours
      FROM AGENCE.LesEtapes
      UNION
      SELECT numC, (rmin-1) AS rang, vDep, 0 AS nbJours
      FROM AGENCE.LesCircuits
      NATURAL JOIN ( SELECT numC, min(rang) AS rmin
                     FROM AGENCE.LesEtapes
                     GROUP BY numC )
      UNION
      SELECT numC, (rmax+1) AS rang, vArr, 0 AS nbJours
      FROM AGENCE.LesCircuits
      NATURAL JOIN ( SELECT numC, max(rang) AS rmax
                     FROM AGENCE.LesEtapes
                     GROUP BY numC )
      )
/*Y(numC, rang, ville, nbJours)<v,x,y,z> ∈ Y ⇔ Le circuit de numéro v reste z
jour dans la ville y de rang x */
SELECT numC, dateDep, pays, (dateDep+nvl(nbJ,0)) AS dateEntree
FROM (
      SELECT X1.numC, X1.rang, X1.pays
```

```
      FROM ( X ) X1
      JOIN ( X ) X2
      ON ( X1.numC=X2.numC AND X1.pays!=X2.pays AND X2.rang=(X1.rang-1) )
      UNION
      SELECT numC, rang, pays
      FROM X
      WHERE rang=0 ) A
/*A(numC, rang, pays)<x,y,z> ∈ A ⇔ Le pays z est différent du pays de la
ville de rang y précédente pour le circuit numéro x*/
NATURAL LEFT OUTER JOIN (
      SELECT Y1.numC, Y1.rang, sum(Y2.nbJours) AS nbJ
      FROM ( Y ) Y1
      JOIN ( Y ) Y2
      ON ( Y1.numC=Y2.numC AND Y1.rang>Y2.rang )
      GROUP BY Y1.numC, Y1.rang ) B
/*B(numC, rang, nbJ)<x,y,z> ∈ B ⇔ le nombre de jour z est la somme des jours
passés dans les villes de rang y précédentes pour les circuits de numéro x*/
NATURAL JOIN AGENCE.LesProgrammations
ORDER BY numC, dateDep, dateEntree;
```

Résultats et tests de la requête SQL et des sous-requêtes :

Résultat de la requête R13 :

R13(numC, dateDep, pays, dateEntree)<w,x,y,z> ∈ R13 ⇔ la programmation de la date x du circuit w arrive dans le pays y à la date z.

NUMC	DATEDEP	PAYS	DATEENTRE
1	04-JAN-10	France	04-JAN-10
1	04-JAN-10	Angleterre	04-JAN-10
1	04-JAN-10	France	06-JAN-10
1	04-FEB-10	Angleterre	04-FEB-10
1	04-FEB-10	France	04-FEB-10
1	04-FEB-10	France	06-FEB-10
1	06-FEB-10	Angleterre	06-FEB-10
1	06-FEB-10	France	06-FEB-10
1	06-FEB-10	France	08-FEB-10
1	21-JUL-10	France	21-JUL-10
1	21-JUL-10	Angleterre	21-JUL-10
1	21-JUL-10	France	23-JUL-10
1	24-JUL-10	Angleterre	24-JUL-10
1	24-JUL-10	France	24-JUL-10
1	24-JUL-10	France	26-JUL-10
2	06-JAN-10	France	06-JAN-10
2	06-JAN-10	Angleterre	06-JAN-10
2	06-JAN-10	France	08-JAN-10
2	07-JAN-10	Angleterre	07-JAN-10
2	07-JAN-10	France	07-JAN-10
2	07-JAN-10	France	09-JAN-10
2	05-FEB-10	Angleterre	05-FEB-10
2	05-FEB-10	France	05-FEB-10
2	05-FEB-10	France	07-FEB-10
2	06-FEB-10	Angleterre	06-FEB-10
2	06-FEB-10	France	06-FEB-10
2	06-FEB-10	France	08-FEB-10
2	05-SEP-10	Angleterre	05-SEP-10
2	05-SEP-10	France	05-SEP-10

2	05-SEP-10	France	07-SEP-10
3	24-DEC-09	Hollande	24-DEC-09
3	24-DEC-09	France	24-DEC-09
3	24-DEC-09	France	27-DEC-09
3	31-DEC-09	Hollande	31-DEC-09
3	31-DEC-09	France	31-DEC-09
3	31-DEC-09	France	03-JAN-10
3	03-JUL-10	Hollande	03-JUL-10
3	03-JUL-10	France	03-JUL-10
3	03-JUL-10	France	06-JUL-10
4	30-JUN-10	Hollande	30-JUN-10
4	30-JUN-10	France	30-JUN-10
4	30-JUN-10	France	04-JUL-10
4	06-AUG-10	France	06-AUG-10
4	06-AUG-10	Hollande	06-AUG-10
4	06-AUG-10	France	10-AUG-10
4	31-AUG-10	France	31-AUG-10
4	31-AUG-10	Hollande	31-AUG-10
4	31-AUG-10	France	04-SEP-10
4	06-NOV-10	France	06-NOV-10
4	06-NOV-10	Hollande	06-NOV-10
4	06-NOV-10	France	10-NOV-10
5	31-AUG-10	Italie	31-AUG-10
5	31-AUG-10	France	31-AUG-10
5	31-AUG-10	France	05-SEP-10
5	06-NOV-10	Italie	06-NOV-10
5	06-NOV-10	France	06-NOV-10
5	06-NOV-10	France	11-NOV-10
6	06-FEB-10	Italie	06-FEB-10
6	06-FEB-10	France	06-FEB-10
6	06-FEB-10	France	10-FEB-10
6	06-SEP-10	France	06-SEP-10
6	06-SEP-10	Italie	06-SEP-10
6	06-SEP-10	France	10-SEP-10
6	06-OCT-10	Italie	06-OCT-10
6	06-OCT-10	France	06-OCT-10
6	06-OCT-10	France	10-OCT-10
6	16-NOV-10	Italie	16-NOV-10
6	16-NOV-10	France	16-NOV-10
6	16-NOV-10	France	20-NOV-10
7	16-DEC-09	France	16-DEC-09
7	06-JAN-10	France	06-JAN-10
7	26-FEB-10	France	26-FEB-10
7	31-AUG-10	France	31-AUG-10
7	31-OCT-10	France	31-OCT-10
8	24-DEC-09	France	24-DEC-09
8	31-DEC-09	France	31-DEC-09
8	14-FEB-10	France	14-FEB-10
8	16-FEB-10	France	16-FEB-10
8	21-FEB-10	France	21-FEB-10
8	28-FEB-10	France	28-FEB-10
8	26-APR-10	France	26-APR-10
8	16-MAY-10	France	16-MAY-10
8	21-JUL-10	France	21-JUL-10
8	31-AUG-10	France	31-AUG-10
8	14-NOV-10	France	14-NOV-10
9	06-FEB-10	Angleterre	06-FEB-10
9	06-FEB-10	France	06-FEB-10
9	06-FEB-10	France	14-FEB-10
9	30-OCT-10	Angleterre	30-OCT-10
9	30-OCT-10	France	30-OCT-10
9	30-OCT-10	France	07-NOV-10
10	01-JAN-10	Irlande	01-JAN-10
10	21-JAN-10	Irlande	21-JAN-10
10	11-FEB-10	Irlande	11-FEB-10
10	28-FEB-10	Irlande	28-FEB-10

11	06-FEB-10	Irlande	06-FEB-10
11	28-FEB-10	Irlande	28-FEB-10
11	29-MAY-10	Irlande	29-MAY-10
11	30-JUN-10	Irlande	30-JUN-10
12	06-FEB-10	Irlande	06-FEB-10
13	31-DEC-09	France	31-DEC-09
13	31-DEC-09	Danemark	31-DEC-09
13	31-DEC-09	Groenland	04-JAN-10
13	31-DEC-09	France	18-JAN-10
13	01-JAN-10	France	01-JAN-10
13	01-JAN-10	Danemark	01-JAN-10
13	01-JAN-10	Groenland	05-JAN-10
13	01-JAN-10	France	19-JAN-10
13	06-FEB-10	France	06-FEB-10
13	06-FEB-10	Danemark	06-FEB-10
13	06-FEB-10	Groenland	10-FEB-10
13	06-FEB-10	France	24-FEB-10
13	30-APR-10	France	30-APR-10
13	30-APR-10	Danemark	30-APR-10
13	30-APR-10	Groenland	04-MAY-10
13	30-APR-10	France	18-MAY-10
13	06-MAY-10	Danemark	06-MAY-10
13	06-MAY-10	France	06-MAY-10
13	06-MAY-10	Groenland	10-MAY-10
13	06-MAY-10	France	24-MAY-10
13	31-MAY-10	France	31-MAY-10
13	31-MAY-10	Danemark	31-MAY-10
13	31-MAY-10	Groenland	04-JUN-10
13	31-MAY-10	France	18-JUN-10
13	30-JUN-10	France	30-JUN-10
13	30-JUN-10	Danemark	30-JUN-10
13	30-JUN-10	Groenland	04-JUL-10
13	30-JUN-10	France	18-JUL-10
13	31-DEC-10	Danemark	31-DEC-10
13	31-DEC-10	France	31-DEC-10
13	31-DEC-10	Groenland	04-JAN-11
13	31-DEC-10	France	18-JAN-11
14	06-JUL-10	Islande	06-JUL-10
14	26-JUL-10	Islande	26-JUL-10
15	06-FEB-10	Islande	06-FEB-10
15	06-FEB-10	France	06-FEB-10
15	06-FEB-10	France	14-FEB-10
15	31-AUG-10	Islande	31-AUG-10
15	31-AUG-10	France	31-AUG-10
15	31-AUG-10	France	08-SEP-10
16	06-JAN-10	Finlande	06-JAN-10
16	06-JAN-10	France	06-JAN-10
16	06-JAN-10	Norvege	08-JAN-10
16	06-JAN-10	France	16-JAN-10
16	06-FEB-10	Finlande	06-FEB-10
16	06-FEB-10	France	06-FEB-10
16	06-FEB-10	Norvege	08-FEB-10
16	06-FEB-10	France	16-FEB-10
16	31-AUG-10	Finlande	31-AUG-10
16	31-AUG-10	France	31-AUG-10
16	31-AUG-10	Norvege	02-SEP-10
16	31-AUG-10	France	10-SEP-10
17	06-FEB-10	Portugal	06-FEB-10
17	16-FEB-10	Portugal	16-FEB-10
17	26-FEB-10	Portugal	26-FEB-10
17	31-DEC-10	Portugal	31-DEC-10
18	30-APR-10	Italie	30-APR-10
18	30-APR-10	France	30-APR-10
18	30-APR-10	France	12-MAY-10
18	30-JUL-10	France	30-JUL-10
18	30-JUL-10	Italie	30-JUL-10

```
18 30-JUL-10 France              11-AUG-10
18 06-OCT-10 Italie              06-OCT-10
18 06-OCT-10 France              06-OCT-10
18 06-OCT-10 France              18-OCT-10
18 06-DEC-10 Italie              06-DEC-10
18 06-DEC-10 France              06-DEC-10
18 06-DEC-10 France              18-DEC-10
19 06-FEB-10 Italie              06-FEB-10
19 15-APR-10 Italie              15-APR-10
19 05-AUG-10 Italie              05-AUG-10
19 06-SEP-10 Italie              06-SEP-10
19 16-SEP-10 Italie              16-SEP-10
19 10-DEC-10 Italie              10-DEC-10
19 20-DEC-10 Italie              20-DEC-10
20 22-DEC-09 France              22-DEC-09
20 20-JAN-10 France              20-JAN-10
20 27-JAN-10 France              27-JAN-10
20 02-FEB-10 France              02-FEB-10
20 06-FEB-10 France              06-FEB-10
20 01-APR-10 France              01-APR-10
20 03-APR-10 France              03-APR-10
21 15-JAN-10 France              15-JAN-10
21 15-JAN-10 Finlande            15-JAN-10
21 15-JAN-10 Norvege             17-JAN-10
21 15-JAN-10 Italie              25-JAN-10
21 25-JAN-10 Finlande            25-JAN-10
21 25-JAN-10 France              25-JAN-10
21 25-JAN-10 Norvege             27-JAN-10
21 25-JAN-10 Italie              04-FEB-10
21 26-FEB-10 Finlande            26-FEB-10
21 26-FEB-10 France              26-FEB-10
21 26-FEB-10 Norvege             28-FEB-10
21 26-FEB-10 Italie              08-MAR-10
21 06-APR-10 France              06-APR-10
21 06-APR-10 Finlande            06-APR-10
21 06-APR-10 Norvege             08-APR-10
21 06-APR-10 Italie              16-APR-10
```

Tests de la requête R13 :

1) Je vérifie que le circuit X commence, se déroule et se finit dans un seul pays, sort bien dans les résultats de la requête.

```
      NUMC DATEDEP   PAYS                 DATEENTRE
---------- --------- -------------------- ---------
         7 31-OCT-10 France               31-OCT-10
         8 24-DEC-09 France               24-DEC-09
        14 06-JUL-10 Islande              06-JUL-10
```

Le test est passant. Les circuits 7, 8 et 14 se déroule dans un seul et même pays, ils apparaissent bien qu'une seule fois dans la table résultat avec les dates de départ et d'entrée identiques.

2) Je vérifie que le circuit X commence, se déroule et se finit dans des pays différents, sort bien dans les résultats de la requête.

NUMC	DATEDEP	PAYS	DATEENTRE
21	06-APR-10	France	06-APR-10
21	06-APR-10	Finlande	06-APR-10
21	06-APR-10	Norvege	08-APR-10
21	06-APR-10	Italie	16-APR-10

Le test est passant. Le circuit 21 est présent 4 fois pour la programmation du 06-APR-10) car il entre dans 4 pays différents.

3) Je vérifie que le circuit X a sa dernière étape et sa ville d'arrivée dans le même pays et passe dans un pays différent au cours du séjour, sort bien dans les résultats de la requête.

NUMC	DATEDEP	PAYS	DATEENTRE
21	06-APR-10	France	06-APR-10
21	06-APR-10	Finlande	06-APR-10
21	06-APR-10	Norvege	08-APR-10
21	06-APR-10	Italie	16-APR-10

Le test est passant. Le circuit 21 a pour dernière étape et pour ville d'arriver deux villes italiennes. On a bien le pays Italie qui apparait une seule fois.

4) Je vérifie que le circuit X passant plusieurs fois dans le même pays avec des dates d'entrée différentes sort dans les résultats de la requête.

NUMC	DATEDEP	PAYS	DATEENTRE
1	04-JAN-10	France	04-JAN-10
1	04-JAN-10	Angleterre	04-JAN-10
1	04-JAN-10	France	06-JAN-10
13	31-DEC-09	France	31-DEC-09
13	31-DEC-09	Danemark	31-DEC-09
13	31-DEC-09	Groenland	04-JAN-10
13	31-DEC-09	France	18-JAN-10

Le test est passant. Les circuits 1 et 13, le pays France est présent deux fois car les circuits entre dans ce pays à deux fois dans ce pays à des dates différentes.

Résultat de la sous-requête X :

X(numC, rang, pays)<x,y,z> ∈ X ⇔ Le numéro de circuit x possède une ville de rang y ayant pour pays z.

NUMC	RANG	PAYS
1	0	France
1	1	Angleterre
1	2	France
2	0	France
2	1	Angleterre
2	2	France
3	0	France
3	1	Hollande
3	2	France
4	0	France
4	1	Hollande
4	2	France
5	0	France
5	1	Italie
5	2	France
6	0	France
6	1	Italie
6	2	France
7	0	France
7	1	France
7	2	France
7	3	France
7	4	France
7	5	France
7	6	France
8	0	France
8	1	France
8	2	France
8	3	France
8	4	France
9	0	France
9	1	Angleterre
9	2	Angleterre
9	3	Angleterre
9	4	Angleterre
9	5	Angleterre
9	6	Angleterre
9	7	France
10	0	Irlande
10	1	Irlande
10	2	Irlande
10	3	Irlande
10	4	Irlande
10	5	Irlande
10	6	Irlande
10	7	Irlande
10	8	Irlande
11	0	Irlande
11	1	Irlande
11	2	Irlande
11	3	Irlande
11	4	Irlande
11	5	Irlande
11	6	Irlande
11	7	Irlande
11	8	Irlande
12	0	Irlande
12	1	Irlande

12	2	Irlande
12	3	Irlande
12	4	Irlande
12	5	Irlande
12	6	Irlande
12	7	Irlande
12	8	Irlande
13	0	France
13	1	Danemark
13	2	Groenland
13	3	Groenland
13	4	Groenland
13	5	Groenland
13	6	Groenland
13	7	Groenland
13	8	Groenland
13	9	France
14	0	Islande
14	1	Islande
14	2	Islande
14	3	Islande
14	4	Islande
14	5	Islande
14	6	Islande
14	7	Islande
14	8	Islande
15	0	France
15	1	Islande
15	2	Islande
15	3	Islande
15	4	Islande
15	5	France
16	0	France
16	1	Finlande
16	2	Norvege
16	3	Norvege
16	4	Norvege
16	5	Norvege
16	6	Norvege
16	7	Norvege
16	8	Norvege
16	9	France
17	0	Portugal
17	1	Portugal
17	2	Portugal
17	3	Portugal
17	4	Portugal
17	5	Portugal
17	6	Portugal
17	7	Portugal
17	8	Portugal
17	9	Portugal
17	10	Portugal
18	0	France
18	1	Italie
18	2	Italie
18	3	Italie
18	4	Italie
18	5	Italie
18	6	France
19	0	Italie
19	1	Italie
19	2	Italie
19	3	Italie
19	4	Italie
19	5	Italie

19	6	Italie
19	7	Italie
19	8	Italie
19	9	Italie
19	10	Italie
19	11	Italie
19	12	Italie
19	13	Italie
19	14	Italie
20	0	France
20	1	France
20	2	France
21	0	France
21	1	Finlande
21	2	Norvege
21	3	Norvege
21	4	Norvege
21	5	Norvege
21	6	Norvege
21	7	Norvege
21	8	Norvege
21	9	Italie
21	10	Italie
21	11	Italie
21	12	Italie
21	13	Italie
21	14	Italie

Tests de la sous-requête X :

1) Vérifier que le circuit X a bien un rang 0 pour le pays de départ et que le rang du pays d'arrivé est bien le plus grand.

NUMC	RANG	PAYS
1	0	France
1	1	Angleterre
1	2	France
2	0	France
2	1	Angleterre
2	2	France

Le test est passant. Exemple pour les circuits 1 et 2, on voit que la France étant le pays de départ et d'arrivé de ces circuits a bien les rang 0 et maximum.

Résultat de la sous-requête Y :

Y(numC, rang, ville, nbJours)<v,x,y,z> ∈ Y ⇔ Le circuit de numéro v reste z jour dans la ville y de rang x.

NUMC	RANG	VILLE	NBJOURS
1	0	Paris	0
1	1	Londres	2
1	2	Paris	0
2	0	Paris	0
2	1	Londres	2
2	2	Paris	0
3	0	Paris	0
3	1	Amsterdam	3

3	2	Paris	0
4	0	Paris	0
4	1	Amsterdam	4
4	2	Paris	0
5	0	Lyon	0
5	1	Venise	5
5	2	Lyon	0
6	0	Paris	0
6	1	Venise	4
6	2	Paris	0
7	0	Quiberon	0
7	1	Quiberon	2
7	2	Port Cotton	2
7	3	Port Maria	1
7	4	Houat	1
7	5	Hoedic	1
7	6	Hoedic	0
8	0	Clermont Ferrand	0
8	1	Laschamps	2
8	2	Orcival	3
8	3	Besse	2
8	4	Besse	0
9	0	Paris	0
9	1	Sissinghurst	2
9	2	Salisbury	2
9	3	Exeter	1
9	4	St Ives	1
9	5	Bath	1
9	6	Londres	1
9	7	Paris	0
10	0	Shannon	0
10	1	Shannon	1
10	2	Dingle	3
10	3	Killarney	2
10	4	Kenmare	1
10	5	Glengariff	3
10	6	Bantry	2
10	7	Cork	1
10	8	Cork	0
11	0	Shannon	0
11	1	Galway	2
11	2	Clifden	2
11	3	Cleggan	2
11	4	Leenan	1
11	5	Louisburgh	2
11	6	Westport	2
11	7	Ennis	2
11	8	Shannon	0
12	0	Dublin	0
12	1	Donegal	1
12	2	Killibegs	2
12	3	Carrick	2
12	4	Ardara	2
12	5	Dungloe	2
12	6	Letterkenny	1
12	7	Dublin	2
12	8	Dublin	0
13	0	Paris	0
13	1	Copenhague	4
13	2	Jakobshavn	2
13	3	Christianshab	2
13	4	Godhavn	1
13	5	Disko	2
13	6	Egedesmine	1
13	7	Holsteinborg	4
13	8	Sondre	2

13	9	Paris	0
14	0	Reykjavik	0
14	1	Reykjavik	3
14	2	Gullfoss	2
14	3	Geysir	2
14	4	Hveravellir	2
14	5	Akureyri	2
14	6	Myvatn	2
14	7	Asbyrgi	2
14	8	Reykjavik	0
15	0	Paris	0
15	1	Reykjavik	2
15	2	Reykholt	2
15	3	Husavik	2
15	4	Godafoss	2
15	5	Paris	0
16	0	Paris	0
16	1	Helsinky	2
16	2	Ylivieska	1
16	3	Tornio	1
16	4	Cap Nord	1
16	5	Hammerfest	1
16	6	Inari	1
16	7	Salla	1
16	8	Saarijarvi	2
16	9	Paris	0
17	0	Lisbonne	0
17	1	Lisbonne	2
17	2	Leiria	1
17	3	Porto	2
17	4	Vila Real	2
17	5	Urgeirica	1
17	6	Fatima	2
17	7	Castelo de Vide	1
17	8	Evora	2
17	9	Faro	1
17	10	Lisbonne	0
18	0	Paris	0
18	1	Rome	3
18	2	Florence	3
18	3	Ravenne	2
18	4	Verone	1
18	5	Venise	3
18	6	Paris	0
19	0	Rome	0
19	1	Rome	2
19	2	Naples	1
19	3	Pompei	1
19	4	Capri	1
19	5	Salerne	1
19	6	Cosenza	2
19	7	Catanzaro	1
19	8	Metaponto	1
19	9	Lecce	1
19	10	Brindisi	1
19	11	Bari	2
19	12	Benevento	2
19	13	Caserte	2
19	14	Rome	0
20	0	Briancon	0
20	1	Briancon	6
20	2	Briancon	0
21	0	Paris	0
21	1	Helsinky	2
21	2	Ylivieska	1
21	3	Tornio	1

21	4	Cap Nord	1
21	5	Hammerfest	1
21	6	Inari	1
21	7	Salla	1
21	8	Saarijarvi	2
21	9	Rome	3
21	10	Florence	3
21	11	Ravenne	2
21	12	Verone	1
21	13	Venise	3
21	14	Rome	0

Tests de la sous-requête Y :

1) Vérifier que les villes de départ et d'arrivé d'un circuit X sortent dans les résultats avec un nombre de jour égale à 0.

NUMC	RANG	VILLE	NBJOURS
1	0	Paris	0
1	1	Londres	2
1	2	Paris	0
2	0	Paris	0
2	1	Londres	2
2	2	Paris	0

Le test est passant. Pour les circuits 1 et 2, Paris est ville de départ et d'arrivée. Ces lignes ont bien un nombre de jours égale à 0.

Résultat de la sous-requête A :

A(numC, rang, pays)<x,y,z> ∈ A ⇔ Le pays z est différent du pays de la ville de rang y précédente pour le circuit numéro x.

NUMC	RANG	PAYS
1	0	France
1	1	Angleterre
1	2	France
2	0	France
2	1	Angleterre
2	2	France
3	0	France
3	1	Hollande
3	2	France
4	0	France
4	1	Hollande
4	2	France
5	0	France
5	1	Italie
5	2	France
6	0	France
6	1	Italie
6	2	France
7	0	France
8	0	France
9	0	France
9	1	Angleterre
9	7	France
10	0	Irlande

11	0	Irlande
12	0	Irlande
13	0	France
13	1	Danemark
13	2	Groenland
13	9	France
14	0	Islande
15	0	France
15	1	Islande
15	5	France
16	0	France
16	1	Finlande
16	2	Norvege
16	9	France
17	0	Portugal
18	0	France
18	1	Italie
18	6	France
19	0	Italie
20	0	France
21	0	France
21	1	Finlande
21	2	Norvege
21	9	Italie

Tests de la sous-requête A :

1) Vérifier que le circuit X a un pays de rang r différent du pays de rang r-1, sort dans les résultats de la requête.

NUMC	RANG	PAYS
13	0	France
13	1	Danemark
13	2	Groenland
13	9	France
21	0	France
21	1	Finlande
21	2	Norvege
21	9	Italie

Le test est passant, les circuits 13 et 21 ont bien des rangs croissants entre deux pays différents.

2) Vérifier que le circuit X n'a jamais deux pays identiques pour des rangs qui se suivent.

NUMC	RANG	PAYS
13	0	France
13	1	Danemark
13	2	Groenland
13	9	France

Le test est passant, le circuit 13 entre deux fois en France, mais il y a bien des pays de rangs différents entre les deux entrées en France

Résultat de la sous-requête B :

B(numC, rang, nbJ)<x,y,z> ∈ B ⇔ le nombre de jour z est la somme des jours passés dans les villes de rang y précédentes pour les circuits de numéro x

NUMC	RANG	NBJ
1	1	0
1	2	2
2	1	0
2	2	2
3	1	0
3	2	3
4	1	0
4	2	4
5	1	0
5	2	5
6	1	0
6	2	4
7	1	0
7	2	2
7	3	4
7	4	5
7	5	6
7	6	7
8	1	0
8	2	2
8	3	5
8	4	7
9	1	0
9	2	2
9	3	4
9	4	5
9	5	6
9	6	7
9	7	8
10	1	0
10	2	1
10	3	4
10	4	6
10	5	7
10	6	10
10	7	12
10	8	13
11	1	0
11	2	2
11	3	4
11	4	6
11	5	7
11	6	9
11	7	11
11	8	13
12	1	0
12	2	1
12	3	3
12	4	5
12	5	7
12	6	9
12	7	10
12	8	12
13	1	0
13	2	4
13	3	6
13	4	8
13	5	9

13	6	11
13	7	12
13	8	16
13	9	18
14	1	0
14	2	3
14	3	5
14	4	7
14	5	9
14	6	11
14	7	13
14	8	15
15	1	0
15	2	2
15	3	4
15	4	6
15	5	8
16	1	0
16	2	2
16	3	3
16	4	4
16	5	5
16	6	6
16	7	7
16	8	8
16	9	10
17	1	0
17	2	2
17	3	3
17	4	5
17	5	7
17	6	8
17	7	10
17	8	11
17	9	13
17	10	14
18	1	0
18	2	3
18	3	6
18	4	8
18	5	9
18	6	12
19	1	0
19	2	2
19	3	3
19	4	4
19	5	5
19	6	6
19	7	8
19	8	9
19	9	10
19	10	11
19	11	12
19	12	14
19	13	16
19	14	18
20	1	0
20	2	6
21	1	0
21	2	2
21	3	3
21	4	4
21	5	5
21	6	6
21	7	7
21	8	8

21	9	10
21	10	13
21	11	16
21	12	18
21	13	19
21	14	22

Tests de la sous-requête B :

1) Vérifier que le circuit X le rang 1 sort avec un nombre de jour égale à 0

NUMC	RANG	NBJ
1	1	0
1	2	2
2	1	0
2	2	2
3	1	0
3	2	3

Le test est passant. Les circuits 1, 2 et 3 ont bien un nombre de jours égale à 0 pour le rang 1.

2) Vérifier que le circuit X a bien pour chaque rang un nombre de jour égale à la somme des jours des rangs précédents.

NUMC	RANG	NBJ	NBJOURS
21	1	0	2
21	2	2	1
21	3	3	1
21	4	4	1
21	5	5	1
21	6	6	1
21	7	7	1
21	8	8	1
21	9	10	2
21	10	13	3
21	11	16	3
21	12	18	2
21	13	19	1
21	14	22	3

Le test est passant. Pour le circuit 21, chaque valeur de l’attribut NBJ correspond à la somme des lignes de NBJOURS de rangs précédents