TD 1 : Algèbre relationnelle

Logistique et Algèbre relationnelle

2024-09-20

Avec solutions

L3 MIASHS/Ingémath
Université Paris Cité
Année 2024
Course Homepage
Moodle

Récapitulatif

Schémas

Les schémas sont une abstraction spécifique à PostgreSQL. Les schémas permettent de faire cohabiter sur une même base de données (ou catalogue dans le jargon PostgreSQL) plusieurs ensembles d'informations de nature différentes. On peut indiquer à `pgcli`, `psql` ou un autre client (`dbeaver` ou autre) quels schémas on veut utiliser *par défaut*.

Dans ce TP, nous allons nous concentrer sur le schéma `world` qui contient des informations concernant des villes et des pays/territoires du monde entier. Au cours de ce semestre, nous serons amenés à utiliser d'autres schémas : `pagila` qui contient des informations concernant des films, …

Pour lister et modifier les schémas de votre environnement de travail :

```
bd_2023-24> SHOW search_path ;                  -- lister
bd_2023-24> SET search_path TO world, public ;  -- modifier
bd_2023-24> SHOW search_path ;       -- visualiser
```

Lister les schémas du catalogue `bd_2023-24`.

```
\dn
```

Pour lister les tables des schemas inscrits sur votre `search_path`.

```
bd_2023-24> \d
+----------+---------------------+--------+----------+
| Schema   | Name                | Type   | Owner    |
|----------+---------------------+--------+----------|
| world    | city                | table  | postgres |
| world    | country             | table  | postgres |
| world    | countrylanguage     | table  | postgres |
+----------+---------------------+--------+----------+
```

Vue d'ensemble du schéma `world`

Suggestion

Si vous travaillez avec `pgcli`/`psql`, utilisez en local votre éditeur préféré (emacs, vi, sublime text, visual studio code, …), transférez votre script sql grâce à `scp`, et chargez le script dans votre session `psql`/`pgcli` à l'aide de `\\i`.

Tables

Pour voir la définition (c'est-à-dire les différentes colonnes) d'une table :

```
bd_2023-24> \d world.country

+--------------------+---------------+-------------+
| Column             | Type          | Modifiers   |
|--------------------+---------------+-------------|
| countrycode        | character(3)  |  not null   |
| name_country       | text          |  not null   |
| continent          | text          |  not null   |
| region             | text          |  not null   |
| surfacearea        | real          |  not null   |
| indepyear          | smallint      |             |
| population_country | integer       |  not null   |
| lifeexpectancy     | real          |             |
| gnp                | numeric(10,2) |             |
| gnpold             | numeric(10,2) |             |
| localname          | text          |  not null   |
| governmentform     | text          |  not null   |
| headofstate        | text          |             |
| capital            | integer       |             |
| code2              | character(2)  |  not null   |
+--------------------+---------------+-------------+
```

Explorer les possibilités de `psql` ou de `pgcli`

```
bd_2023-24> \?
```

Dans la deuxième partie du TP, on s'intéresse aux requêtes, c'est-à-dire, les moyens d'extraire une information pertinente d'une base de données.

Écriture d'une requête

Pour extraire des informations d'une base de données, on utilise l'algèbre relationnelle (pour la théorie) et le langage SQL (pour la pratique).

L'*algèbre relationnelle* est un ensemble d'opération sur les tables. Chaque opération prend en argument une ou plusieurs tables et produit une nouvelle table. Nous commençons par introduire deux opérations importantes qui opèrent sur une table à la fois :

- *Projection* : $\pi_{\texttt{liste de colonnes}}R$. Cette opération ne garde que les colonnes mentionnées de la table R. Par exemple $\pi_{\texttt{name,continent}}\texttt{world.country}$ est la table avec deux colonnes `name, continent` et une ligne pour chaque ligne de la table `world.country`.
- *Sélection* : $\sigma_{\texttt{condition}}R$. Cette opération ne garde que les lignes qui respectent la condition indiquée. Par exemple $\sigma_{\texttt{lifeexpectancy < 50}}\texttt{world.country}$ renvoie la table contenant les pays ayant une espérance de vie de moins de 50 ans.

Ces deux opérations peuvent être écrites en SQL ainsi :

```
SELECT colonne1, colonne2, ...
FROM table
WHERE condition;
```

Cette opération affiche les colonnes de `table` indiquées après le `SELECT` qui respectent la `condition`.

C'est la transcription de

$$\Pi_{\texttt{colonne1, colonne2}}(\sigma_{\texttt{condition}}(\texttt{table}))$$

Par exemple :

```
SELECT name_country, lifeexpectancy
FROM world.country
WHERE lifeexpectancy < 50 and continent = 'Asia';
```

affichera le nom et l'espérance de vie des pays d'Asie ayant une espérance de vie inférieure à 50 ans. Remarquez l'utilisation des apostrophes simples pour délimiter les chaînes de caractères ('Asia').

Requêtes monotables

Ecrivez des requêtes en algèbre relationnelle et en SQL (dans `pgcli/psql, ...`) pour extraire les informations suivantes du schéma `world` :

- Quelles sont les régions ? (25 lignes)

Solution

$\Pi_{\texttt{region}}(\texttt{world.country})$

```
SELECT DISTINCT region
FROM world.country ;
```

- Quelles sont les régions situées en Europe ? (6 lignes)

Solution

$\Pi_{\texttt{region}}(\sigma_{\texttt{continent = 'Europe'}}(\texttt{world.country}))$

```
SELECT DISTINCT region
FROM world.country
WHERE continent = 'Europe' ;
```

- Quels sont les pays situés en Europe du sud ? (15 lignes)

Solution

$\Pi_{\texttt{name_country}}(\sigma_{\texttt{region = 'Southern Europe'}}(\texttt{world.country}))$

```
SELECT name_country
FROM world.country
WHERE region = 'Southern Europe' ;
```

- Quelles sont les capitales des pays situés en Europe de l'Ouest ? (quel est le type de la colonne `capital` ?) (9 lignes)

Solution

$$\Pi_{\texttt{capital}}(\sigma_{\texttt{region = 'Western Europe'}}(\texttt{world.country}))$$

```
SELECT capital
FROM world.country
WHERE region = 'Western Europe' ;
```

- A partir de la table `countrylanguage`, quels sont les langues qui sont officielles dans au moins un pays ? (102 lignes)

Solution

$\Pi_{\texttt{language}}(\sigma_{\texttt{isofficial}}(\texttt{world.countrylanguage}))$

```
SELECT DISTINCT language
FROM world.countrylanguage
WHERE isofficial ;
```

La variante de l'algèbre relationnelle vue en cours opérant sur les ensembles, `DISTINCT` est toujours implicitement implémenté par la requête algébrique. On peut noter qu'il existe un opérateur d'élimination des doublons dans les variantes multi-ensemblistes de l'algèbre relationnelle.

- Quels sont les codes des pays où le français est langue officielle ? (18 lignes) Même question pour les langues de votre choix ?

Solution

$\Pi_{\texttt{countrycode}}(\sigma_{\texttt{language = 'French'} \wedge \texttt{isofficial}}(\texttt{world.countrylanguage}))$

```
SELECT countrycode
FROM world.countrylanguage
WHERE language = 'French' AND isofficial ;
```

- Quelle est la date d'indépendance de la France ?

Solution

$\Pi_{\texttt{indepyear}}(\sigma_{\texttt{name_country = 'French'}}(\texttt{world.country}))$

```
SELECT indepyear
FROM world.country
WHERE name_country = 'France' ;
```

- Quelles sont les dates d'indépendance des pays d'Europe ? (46 lignes)

Solution

$\Pi_{\texttt{name_country, indepyear}}(\sigma_{\texttt{continent = 'Europe'}}\ (\texttt{world.country}))$

```
SELECT name_country , indepyear
FROM world.country
WHERE continent = 'Europe' ;
```

- Quelles sont les villes françaises de plus de 200 000 habitants ? (10 lignes)

Solution

$\Pi_{\texttt{name, population}}(\sigma_{\texttt{countrycode = 'FRA'} \wedge \texttt{population > 200000}}(\texttt{world.city}))$

```
SELECT name , population
FROM world.city
WHERE countrycode = 'FRA' AND population > 200000 ;
```

- Pour chaque pays européen, calculer la densité, le GNP par habitant, et l'espérance de vie, ordonner par densité décroissante. (46 lignes)

Solution

$\rho_{\texttt{population_country/surfacearea} \mapsto \texttt{density,gnp/population_country} \mapsto \texttt{gnp_per_hab}}(\Pi_{\texttt{name_country, population_country / surfacearea}}(\sigma_{\texttt{conti}}$

```
SELECT name_country , population_country / surfacearea as density ,
       gnp / population_country as gnp_per_hab , lifeexpectancy
FROM country
WHERE continent = 'Europe'
ORDER BY density DESC;
```

La contrepartie algébrique de `ORDER BY` n'a pas été présentée dans la variante de l'algèbre relationnelle vue en cours, même si elle est évidemment implémentée (via des opérateurs de tri) dans les variantes utilisées en pratique par les SGBDs.

- Quels sont les pays où l'espérance de vie n'est pas inférieure à 77 ans et le pnb par habitant n'est pas supérieur à (0.010) ? (10 lignes)

Solution

$$\Pi_{\texttt{name_country}}(\sigma_{\neg \texttt{lifeexpectancy < 77} \wedge \neg \texttt{(gnp / population_country) > 0.01}}(\texttt{world.country}))$$

ou, de manière équivalente \

$$\Pi_{\texttt{name_country}}(\sigma_{\texttt{lifeexpectancy >= 77} \wedge \texttt{(gnp / population_country) >= 0.01}}(\texttt{world.country}))$$

```
SELECT name_country
FROM world.country
WHERE NOT (lifeexpectancy < 77) AND
      NOT (gnp / population_country) > 0.01 ;
```

- Quels sont les pays tels que la condition (espérance de vie supérieure ou égale à 77 ans ou PNB par habitant inférieur à (0.01)) n'est pas vérifée ? (16 lignes)

Solution

$$\Pi_{\texttt{name_country}}(\sigma_{\neg \texttt{(lifeexpectancy >= 77} \vee \texttt{(gnp / population_country) < 0.01)}}(\texttt{world.country}))$$

ou, de manière équivalente

$$\Pi_{\texttt{name_country}}(\sigma_{\texttt{lifeexpectancy < 77} \wedge \texttt{(gnp / population_country) >= 0.01}}(\texttt{world.country}))$$

```
SELECT name_country
FROM world.country
WHERE NOT (lifeexpectancy >= 77 OR (gnp / population_country) < 0.01) ;
```

- Quels sont les pays où une langue est officielle sans être parlée par au moins la moitié de la population ? (92 lignes)

Solution

$$\Pi_{\texttt{countrycode}}(\sigma_{\texttt{isofficial} \wedge \texttt{percentage < 50}}(\texttt{world.countrylanguage}))$$

```
SELECT DISTINCT countrycode
FROM world.countrylanguage
WHERE isofficial AND percentage < 50 ;
```

- Quels sont les pays qui ont au moins une langue officielle ? (190 lignes)

Solution

$$\Pi_{\texttt{countrycode}}(\sigma_{\texttt{isofficial}}(\texttt{world.countrylanguage}))$$

```
SELECT DISTINCT countrycode
FROM world.countrylanguage
WHERE isofficial ;
```

- Quels sont les noms des pays qui comptent plus de 100 000 000 d'habitants ? (10 lignes)

Solution

$$\Pi_{\texttt{name_country}}(\sigma_{\texttt{population_country > 100000000}}(\texttt{world.country}))$$

```
SELECT name_country
FROM world.country
WHERE population_country > 100000000 ;
```

Requêtes multi-tables

On peut aussi combiner plusieurs tables. Pour ce TP, nous allons seulement présenter le produit cartésien de deux tables : $T \times S$ est la table dont les colonnes sont les colonnes de S et les colonnes de T. Ces lignes contient tous les couples (l_1, l_2) où l_1 est une ligne de T et l_2 est une ligne de S. En SQL, on écrira :

```
SELECT col1,col2
FROM table1,table2
WHERE condition;
```

Par exemple,

```
SELECT language
FROM world.country as c, world.countrylanguage as l
WHERE c.countrycode = l.countrycode and c.continent = 'Europe';
```

affichera les langues parlées en Europe. Remarquez l'utilisation des `as` pour donner de nouveaux noms aux tables et l'utilisation de `c.countrycode` pour lever l'ambiguité sur des noms de colonnes qui seraient éventuellement les mêmes.

Avec ça, écrivez des requêtes pour les questions suivantes :

- Quels sont les noms des capitales Sud-Américaines ? (14 lignes)

Solution

$$\Pi_{\texttt{name}}(\sigma_{\texttt{capital = id} \wedge \texttt{region = 'South America'}}(\rho_{\texttt{countrycode} \mapsto \textit{city_countrycode}}(\texttt{world.city}) \times \texttt{world.country}))$$

```
SELECT name
FROM world.country , world.city
WHERE capital = id AND region = 'South America' ;
```

Attention ! En toute rigueur le produit cartésien ne s'applique que sur des relations de schémas disjoints, d'où le renommage utilisé dans la requête algébrique ci-dessus. En pratique on utilisera tout de même :

$$\Pi_{\texttt{name}}(\sigma_{\texttt{capital = id} \wedge \texttt{region = 'South America'}}(\texttt{world.country} \times \texttt{world.city}))$$

comme abréviation pour :

$$\Pi_{\texttt{name}}(\sigma_{\texttt{capital = id} \wedge \texttt{region = 'South America'}}(\rho_{\texttt{countrycode} \mapsto \texttt{country.countrycode}}(\texttt{world.country}) \times \rho_{\texttt{countrycode} \mapsto \texttt{city.countrycod}}$$

Donc, dès que l'on opère un produit cartésien sur deux tables, on supposera implicitement que les attributs en commun sont renommés en les préfixant par le nom de la relation à laquelle ils appartiennent. Si les deux tables sont deux copies d'une seule et même table (cf plus loin question 4, on parle alors d'auto-jointure) on fera suivre le nom de la première copie par 1 et le nom de la seconde copie par 2 et on renommera tous les attributs de chacune des copies en les préfixant par ce nouveau nom.

- Quels sont les noms des pays où le français est langue officielle ? (18 lignes)

Solution

$$\Pi_{\texttt{name_country}}(\sigma_{\texttt{isofficial} \wedge \texttt{language = 'French'}}(\texttt{world.country} \bowtie \texttt{world.countrylanguage}))$$

```
SELECT name_country
FROM world.country NATURAL JOIN world.countrylanguage
WHERE isofficial AND
      language = 'French' ;
```

Dans la variante algébrique, pour utiliser la jointure naturelle on a bien vérifié au préalable que le seul attribut en commun sur les deux relations était bien `countrycode`. En effet, si au contraire, certains attributs s'étaient trouvé avoir le même nom de manière fortuite, il aurait fallu les renommer au préalable dans l'une des deux relations. Une autre solution aurait combiné sélection et produit cartésien afin d'émuler la requête SQL ci-dessus, mais il aurait alors fallu en toute rigueur renommer au préalable l'attribut en commun `countrycode` dans l'une des deux tables en utilisant l'opérateur ρ, le produit cartésien ne pouvant être réalisé que sur des relations de schémas disjoints (c.f., question précédente).

- Quelles sont les pays où l'espagnol est langue officielle et la forme de gouvernement est `Federal Republic` ? (3 lignes)

Solution

$\Pi_{\texttt{name_country}}(\sigma_{\texttt{ttisofficial} \wedge \texttt{language = 'Spanish'} \wedge \texttt{governmentform = 'Federal Republic'}}(\texttt{world.country} \bowtie \texttt{world.countrylang}$

```
SELECT name_country
FROM country NATURAL JOIN countrylanguage
WHERE isofficial AND
      language = 'Spanish' AND
      governmentform = 'Federal Republic' ;
```

- Quels sont les pays qui ont au moins deux langues officielles ? (38 lignes)

Solution

$$\Pi_{\texttt{name_country}}$$

$(\sigma_{\texttt{c1.isofficial} \wedge \texttt{c2.isofficial} \wedge \texttt{c1.language} \neq \texttt{c2.language} \wedge \texttt{countrycode = c1.countrycode} \wedge \texttt{countrycode = c2.countrycode}}(\texttt{world.country}$

```
SELECT DISTINCT name_country
FROM country AS c , countrylanguage AS l1 , countrylanguage AS l2
WHERE c.countrycode = l1.countrycode AND
      c.countrycode = l2.countrycode AND
      l1.isofficial AND
      l2.isofficial AND
      NOT (l1.language = l2.language) ;
```

Attention à bien noter que l'on a utilisé ici dans la requête algébrique l'abréviation de renommage mentionné à la question 1. Pour simplifier l'écriture de la requête on a également abrévié `countrylanguage` en `ci` pour $i \in \{1, 2\}$.
On dispose aussi de la syntaxe `JOIN … USING (a ,…,a )` pour faire des jointures qui ne considèrent que les attributs communs `a` à `a` .

```
SELECT DISTINCT name_country
FROM country NATURAL JOIN
     countrylanguage AS l1 JOIN
     countrylanguage AS l2 USING (countrycode)
WHERE l1.isofficial AND l2.isofficial
AND l1.language <> l2.language ;
```

- Quels sont les pays qui n'ont pas de langue officielle ? (49 lignes)

Solution

$\Pi_{\texttt{name_country}}(\texttt{world.country}) - \Pi_{\texttt{name_country}}(\sigma_{\texttt{isofficial}}(\texttt{world.country} \bowtie \texttt{world.countrylanguage}))$

```
(SELECT name_country
 FROM world.country
)

EXCEPT

(SELECT name_country
 FROM world.country NATURAL JOIN world.countrylanguage
 WHERE isofficial
);
```

- Quels sont les pays qui comportent au moins deux villes de plus de 1 000 000 habitants ? (32 lignes)

Solution

$$\Pi_{\texttt{name_country}}$$

$\setminus$

$(\sigma_{\texttt{c1.population_city > 1000000} \wedge \texttt{c2..population_city > 1000000} \wedge \texttt{c1.id} \neq \texttt{c2.id} \wedge \texttt{countrycode = c1.countrycode} \wedge \texttt{countrycode = c2.countrycode}}$

```
SELECT DISTINCT name_country
FROM country as c , city as v1 , city as v2
WHERE c.countrycode = v1.countrycode  AND
      c.countrycode = v2.countrycode AND
      v1.population_city > 1000000 AND
      v2.population_city > 1000000 AND
      (NOT v1.id = v2.id) ;
```

Ici `c1` (respectivement, `c2`) est utilisé comme abbréviation pour `city1` (respectivement, `city2`).

- Quelles sont les régions qui ne comportent qu'une seule forme de gouvernement ? (3 lignes)

Solution

$$\Pi_{\texttt{region}}(\texttt{world.country}) - \Pi_{\texttt{region}}$$

$$(\sigma_{\texttt{c1.region = c2.region} \wedge \texttt{c1.countrycode<>c2.countrycode} \wedge c1.governmentform <> c2.governmentform}(\texttt{world.c1} \bowtie \texttt{world.c2}))$$

```
(SELECT region
FROM country)
EXCEPT
(SELECT c1.region
 FROM country as c1 , country as c2
WHERE c1.region = c2.region AND
      c1.countrycode <> C2.countrycode AND NOT
      c1.governmentform = c2.governmentform ) ;
```

Ici `c1` (respectivement, `c2`) est utilisé comme abbréviation pour `country1` (respectivement, `country2`).

- Quelles sont les régions où on ne trouve pas de monarchie ? (9 lignes)

Solution

Une solution algébrique aurait la forme générale de la requête de la question 5. En revanche, nous n'avons pas présenté en cours d'implémentation pour l'opérateur de comparaison `LIKE` qui devrait être utilisé dans l'opérateur de sélection. Nous préférons donc en toute rigueur ne pas proposer ici de traduction algébrique.

```
(SELECT region
FROM country)

EXCEPT

(SELECT region
FROM country
WHERE governmentform LIKE '%Monarchy%') ;
```