

TP9

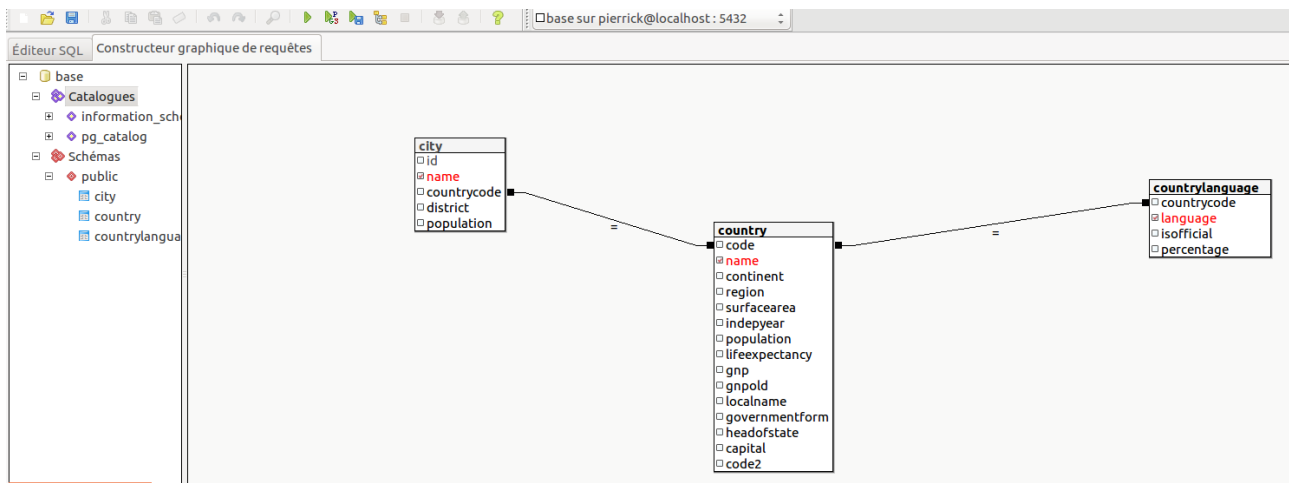


A) Wordl.sql : vive \i !!!

Créer une base de données, lancement du script

B) Pgadmin

Une fois l'installation du logiciel et de la BDD effectuée, on a pu réaliser la requête.



Version graphique de la requête

Il suffisait de cliquer sur les tables dans le menu de gauche puis de sélectionner les champs et enfin de relier les tables.

On a coupé l'output du résultat car cela fait 30670 lignes..... (économisons le papier)

The screenshot shows a SQL IDE interface. At the top, there's a toolbar and a connection bar showing 'base sur pierrick@localhost : 5432'. Below that, the 'Éditeur SQL' (SQL Editor) tab is active, displaying a SQL query. The query selects city names, country names, and country languages from a 'public' schema, joining 'city', 'country', and 'countrylanguage' tables. The 'WHERE' clause filters for matching country codes. Below the editor, the 'Panneau sortie' (Output Panel) is visible, showing the 'Sortie de données' (Data Output) tab. It displays a table with 4 columns: 'name text', 'name text', and 'language text'. The table contains 12 rows of data, followed by an ellipsis indicating more rows, and then rows 30661 through 30670. The data shows cities from Afghanistan and Palestine with their respective languages.

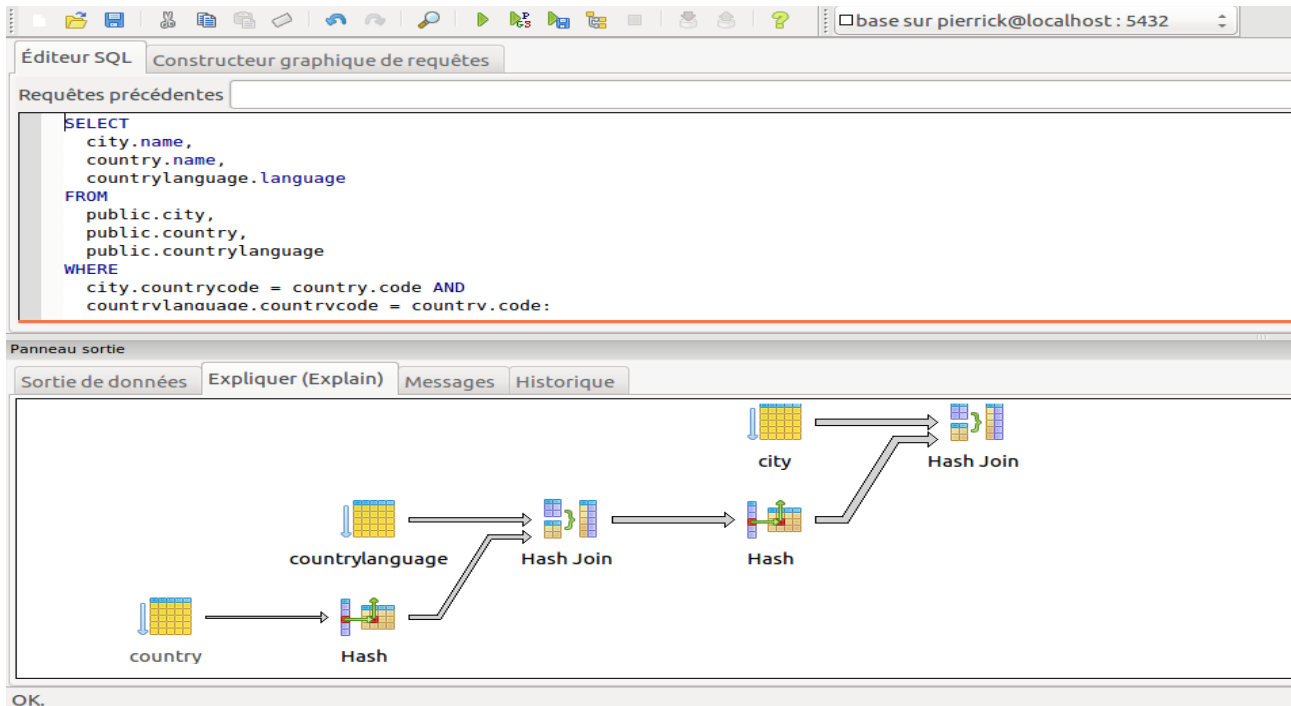
```
SELECT
  city.name,
  country.name,
  countrylanguage.language
FROM
  public.city,
  public.country,
  public.countrylanguage
WHERE
  city.countrycode = country.code AND
  country.code = countrylanguage.countrycode;
```

	name text	name text	language text
1	Kabul	Afghanistan	Balochi
2	Kabul	Afghanistan	Turkmenian
3	Kabul	Afghanistan	Uzbek
4	Kabul	Afghanistan	Dari
5	Kabul	Afghanistan	Pashto
6	Qandahar	Afghanistan	Balochi
7	Qandahar	Afghanistan	Turkmenian
8	Qandahar	Afghanistan	Uzbek
9	Qandahar	Afghanistan	Dari
10	Qandahar	Afghanistan	Pashto
11	Herat	Afghanistan	Balochi
12	Herat	Afghanistan	Turkmenian
.....			
30661	Khan Yunis	Palestine	Hebrew
30662	Khan Yunis	Palestine	Arabic
30663	Hebron	Palestine	Hebrew
30664	Hebron	Palestine	Arabic
30665	Jabaliya	Palestine	Hebrew
30666	Jabaliya	Palestine	Arabic
30667	Nablus	Palestine	Hebrew
30668	Nablus	Palestine	Arabic
30669	Rafah	Palestine	Hebrew
30670	Rafah	Palestine	Arabic

OK.

Résultat de la requête

On exécute avec explain...



Version graphique de l'arbre d'exécution de la requête

On change de fenêtre pour obtenir la version graphique.

Requêtes précédentes

```
SELECT
  city.name,
  country.name,
  countrylanguage.language
FROM
  public.city,
  public.country,
  public.countrylanguage
WHERE
  city.countrycode = country.code AND
  countrylanguage.countrycode = country.code;
```

Panneau sortie

Sortie de données Expliquer (Explain) Messages Historique

	QUERY PLAN
	text
1	Hash Join (cost=52.05..323.37 rows=28798 width=28)
2	Hash Cond: (city.countrycode = country.code)
3	-> Seq Scan on city (cost=0.00..72.79 rows=4079 width=13)
4	-> Hash (cost=39.75..39.75 rows=984 width=27)
5	-> Hash Join (cost=10.38..39.75 rows=984 width=27)
6	Hash Cond: (countrylanguage.countrycode = country.code)
7	-> Seq Scan on countrylanguage (cost=0.00..15.84 rows=984 width=12)
8	-> Hash (cost=7.39..7.39 rows=239 width=15)
9	-> Seq Scan on country (cost=0.00..7.39 rows=239 width=15)

OK.

Version textuelle de l'arbre d'exécution de la requête

C) Explain cost

Ici on a choisit KABUL comme clause pour le where

The screenshot shows the PostgreSQL Explain Cost output for a query. The query is:

```
SELECT
  country.name,
  city.name,
  countrylanguage.language
FROM
  public.country,
  public.city,
  public.countrylanguage
WHERE
  city.countrycode = country.code AND
  countrylanguage.countrycode = country.code AND
  city.name != 'Kabul';
```

The Explain Cost output shows the following query plan:

Step	Operation	Cost	Rows	Width
1	Hash Join	52.05	28791	28
2	Hash Cond: (city.countrycode = country.code)			
3	Seq Scan on city	0.00	4078	13
4	Filter: (name <> 'Kabul'::text)			
5	Hash	39.75	984	27
6	Hash Join	10.38	984	27
7	Hash Cond: (countrylanguage.countrycode = country.code)			
8	Seq Scan on countrylanguage	0.00	1584	12
9	Hash	7.39	239	15
10	Seq Scan on country	0.00	239	15

The diagram on the right illustrates the execution plan: a sequential scan on the 'country' table is joined with a hash of the 'countrylanguage' table. This result is then joined with a hash of the 'city' table, which has been filtered to exclude 'Kabul'.

Requête avec un « seq scan »

Ici on a choisit tout sauf KABUL comme clause pour le where

The screenshot shows the PostgreSQL Explain Cost output for a query. The query is:

```
SELECT
  country.name,
  city.name,
  countrylanguage.language
FROM
  public.country,
  public.city,
  public.countrylanguage
WHERE
  city.countrycode = country.code AND
  countrylanguage.countrycode = country.code AND
  city.name != 'Kabul';
```

The Explain Cost output shows the following query plan:

Step	Operation	Cost	Rows	Width
1	Nested Loop	0.42	91.76	28
2	Join Filter: (city.countrycode = countrylanguage.countrycode)			
3	Nested Loop	0.14	91.16	28
4	Seq Scan on city	0.00	82.99	13
5	Filter: (name = 'Kabul'::text)			
6	Index Scan using country_pkey on country	0.14	8.16	15
7	Index Cond: (code = city.countrycode)			
8	Index Only Scan using countrylanguage_pkey on countrylanguage	0.28	0.55	12
9	Index Cond: (countrycode = country.code)			

The diagram on the right illustrates the execution plan: a sequential scan on the 'city' table is joined with an index scan on the 'country' table using the primary key. The result is then joined with an index only scan on the 'countrylanguage' table using the primary key.

Requête avec un « index on »

Si la condition du WHERE est assez sélective, le REQUEST PLANNER fait un index scan à la place d'un SEQ SCAN.