

ELBEZ Samuel 21200353
samuel.elbe@gmail.com
JACQUETTE Pierrick 21305551
pierrick.jacquette@gmail.com
STASYSZYN Romain 21305734
romain.stasyszyn@gmail.com

TP n°9 du 08/12/2017
Conception Avancée de Bases de Donnée
PG admin Explain
2017-2018

TP9

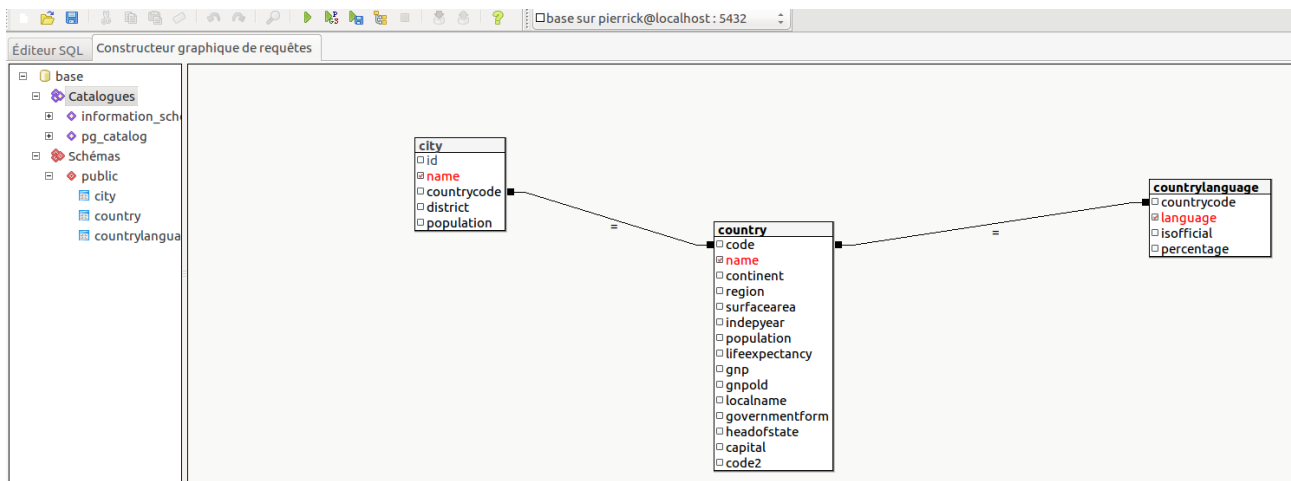


A) Wordl.sql : vive \i !!!

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 rélier les tables.

ELBEZ Samuel 21200353
samuel.elbe@gmail.com
JACQUETTE Pierrick 21305551
pierrick.jacquette@gmail.com
STASYSZYN Romain 21305734
romain.stasyszyn@gmail.com

TP n°9 du 08/12/2017
Conception Avancée de Bases de Donnée
PG admin Explain
2017-2018

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' tab is active, displaying a SQL query. The query selects city names, country names, and country languages from a public schema, filtering by country code. The 'Panneau sortie' (Output Panel) is visible below the editor, showing the results of the query. The results are displayed in a table with four columns: 'name text', 'name text', and 'language text'. The table is divided into two sections by a gap of 30660 rows. The first section contains 12 rows of data for Afghanistan, and the second section contains 10 rows of data for Palestine. The status bar at the bottom shows 'OK'.

```
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

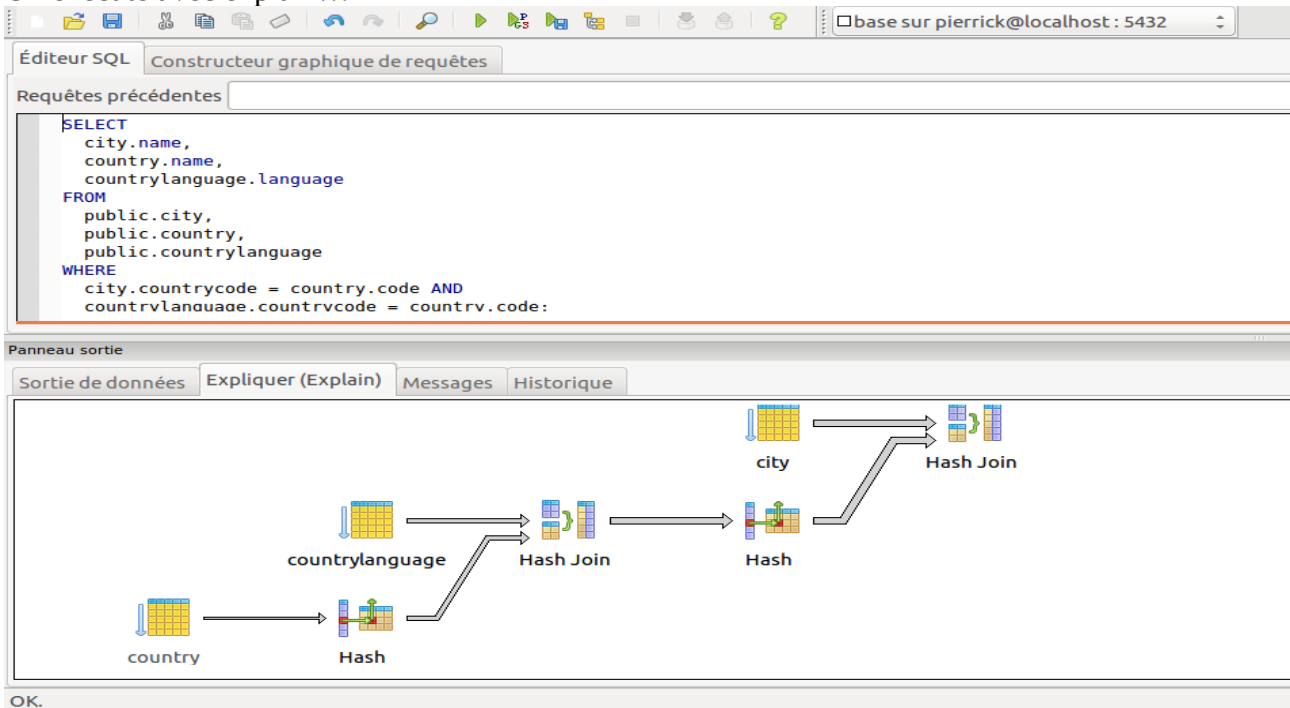
OK.

Résultat de la requête

ELBEZ Samuel 21200353
samuel.elbe@gmail.com
JACQUETTE Pierrick 21305551
pierrick.jacquette@gmail.com
STASYSZYN Romain 21305734
romain.stasyszyn@gmail.com

TP n°9 du 08/12/2017
Conception Avancée de Bases de Donnée
PG admin Explain
2017-2018

On exécute avec explain...



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

On change de fenêtre pour obtenir la version graphique.

The screenshot shows the same PGAdmin 3 interface, but the 'Panneau sortie' is set to 'Sortie de données' (Data Output). It displays the textual execution plan for the same query.

	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)

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

ELBEZ Samuel 21200353
samuel.elbe@gmail.com
JACQUETTE Pierrick 21305551
pierrick.jacquette@gmail.com
STASYSZYN Romain 21305734
romain.stasyszyn@gmail.com

TP n°9 du 08/12/2017
Conception Avancée de Bases de Donnée
PG admin Explain
2017-2018

C) Explain cost

Ici on a choisit KABUL comme clause pour le where

The screenshot shows the PostgreSQL Explain interface. The SQL 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 query plan (QUERY PLAN) is as follows:

Step	Operation	Cost	Rows	Width
1	Hash Join	(cost=52.05..333.52)	rows=28791	width=28
2	Hash Cond: (city.countrycode = country.code)			
3	-> Seq Scan on city	(cost=0.00..82.99)	rows=4078	width=13
4	Filter: (name <> 'Kabul'::text)			
5	-> Hash	(cost=39.75..39.75)	rows=984	width=27
6	-> Hash Join	(cost=10.38..39.75)	rows=984	width=27
7	Hash Cond: (countrylanguage.countrycode = country.code)			
8	-> Seq Scan on countrylanguage	(cost=0.00..15.84)	rows=984	width=12
9	-> Hash	(cost=7.39..7.39)	rows=239	width=15
10	-> Seq Scan on country	(cost=0.00..7.39)	rows=239	width=15

The diagram on the right illustrates the execution plan: a sequential scan on the 'city' table is joined with a hash of a sequential scan on the 'countrylanguage' table. This result is then joined with a hash of a sequential scan on the 'country' table. The final output is a sequential scan on the 'country' table.

Requête avec un « seq scan »

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

The screenshot shows the PostgreSQL Explain interface. The SQL 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 query plan (QUERY PLAN) is as follows:

Step	Operation	Cost	Rows	Width
1	Nested Loop	(cost=0.42..91.76)	rows=7	width=28
2	Join Filter: (city.countrycode = countrylanguage.countrycode)			
3	-> Nested Loop	(cost=0.14..91.16)	rows=1	width=28
4	-> Seq Scan on city	(cost=0.00..82.99)	rows=1	width=13
5	Filter: (name = 'Kabul'::text)			
6	-> Index Scan using country_pkey on country	(cost=0.14..8.16)	rows=1	width=15
7	Index Cond: (code = city.countrycode)			
8	-> Index Only Scan using countrylanguage_pkey on countrylanguage	(cost=0.28..0.55)	rows=4	width=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 'country_pkey' index. This result is then joined with an index only scan on the 'countrylanguage' table using the 'countrylanguage_pkey' index. The final output is a sequential scan on the 'country' table.

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.