IUT2 Grenoble Université Pierre-Mendès-France Informatique

S3
Année 2015-2016

Bases de données avancées

Optimisation des requêtes

Dans ce TP, nous travaillons sur deux types de bases de données des villes, régions et départements de France : la première base est la même base que celle vue dans le TP précédent et comporte le schéma suivant :

TOWNS (id, code, article, name, department)
REGIONS(id, code, capital, name)
DEPARTMENTS(id, code, capital, region, name)

La deuxième base de données comporte le schéma suivant :

all_in_one (code_region, capital_region, name_region, code_departement, capital_departement, name_departement, code_town, article, name_town)

L'objectif de ce TP est de :

1. Analyser et comparer les plans d'exécution des requêtes générés par l'optimiseur de Postgres dans les deux bases de données lors de l'exécution de la même requête.
2. Comparer une base de données normalisée (base du TP précédent) et une autre non normalisée en terme d'insertion de données, modification, suppression.
3. Voir l'intérêt de la normalisation.

Préambule : Création de la base de données et insertion des données

- Lancez un terminal (*Terminal ou Konsole*) et connectez-vous sur Postgres par la commande

  psql –h postgres-info –U users3a01 bases3a01 (par exemple)

- Gardez les tables de la base de données TOWNS du TP précédent.
- Afin d'éviter de créer une autre base de données pour le deuxième schéma, nous allons utiliser la même base que pour le premier schéma (bases3a01 par exemple).
- Exécutez la commande ***\i create_all_towns.sql*** pour créer les tables de la base de données et y insérer les données.
- Vérifiez que la table ***all_in_one*** a été créée par la commande ***\d.*** Vous remarquerez aussi qu'une table de type « SEQUENCE » a également été créée.
  Mais contrairement au TP précédent (première base), vous remarquerez qu'il n'existe qu'un seul index qui a été crée automatiquement par Postgres pour le seul attribut UNIQUE. Ceci est une première comparaison entre les deux bases.

Reprenez les résultats obtenus pour les exercices qui ont été vus dans le TP précédent. Sinon, retrouvez-les en ouvrant deux Konsoles (une Konsole pour la première base et une Konsole pour la deuxième base).

Exercice 1 : Etude des statistiques du schéma

Tapez la commande suivante :

SELECT relname, relpages, reltuples from pg_class order by relname ;

Pour la deuxième base de données, notez les résultats obtenus pour les relnames et comparez-les avec la première base (TP précédent): *departments, departments_capital_key, departments_code_key, departments_id_key, departments_id_seq, departments_name_key, towns, towns_id_key, towns_id_seq, towns_code_department_key, regions, regions_code_key, regions_id_key, regions_id_seq, regions_name_key*.

Question 2 :

Rappel :

Afin que l'estimation du temps d'exécution d'une requête par Postgres soit le plus exact possible, il est nécessaire de mettre à jour les statistiques des tables.

Pour cela, exécutez la commande **ANALYZE**.

Ré-exécutez la commande :

SELECT relname, relpages, reltuples from pg_class order by relname ;

Et examinez les différences concernant les chiffres obtenus en question 1 en les comparant avec celles obtenus dans la deuxième base de données.

Exercice 2 : Etude des plans d'exécution de différents types de requêtes

Dans cet exercice, nous utiliserons la commande EXPLAIN qui donne le plan d'exécution et calcule les coûts d'exécution prévus par l'optimiseur de Postgres dans le cadre d'une requête.

Question 1 : Pour chacune des commandes suivantes :

EXPLAIN SELECT * FROM TOWNS ;

EXPLAIN SELECT code_towns, article, name_ town, code_department FROM all_in_one ;

Comparez les différents coûts générés par ces commandes ?

Quel est le plan d'exécution généré dans ces cas ?

Question 2 :

Examinez et comparez les plans d'exécution des requêtes suivantes :

1. Noms, départements et régions de toutes les villes de France

```
SELECT D.name, department, region
FROM TOWNS T, DEPARTMENTS D
WHERE T.department = D.code ;
```

Comparez cette commande avec

```
SELECT DISTINCT code_region, capital_region, name_region, code_departement, capital_departement, name_departement
FROM all_in_one;
```

2. Noms et départements des villes de la région Rhône-Alpes (avec une jointure)

```
SELECT D.name, department
FROM TOWNS T, DEPARTMENTS D
WHERE T.department = D.code and region = 'Rhône-Alpes' ;
```

```
SELECT  name_town,  name_departement
FROM all_in_one
WHERE name_region= 'Rhône-Alpes';
```

Exercice 3 : Ajout/Modification/Suppression

Question 1 :
Insérez une nouvelle ville au niveau des tables TOWNS pour la première base de données et all_in_one pour la deuxième base :

```
INSERT INTO towns VALUES (36685, 1987, ' ' , 'GRELYON' , 38);

INSERT INTO all_in_one (code_town, article, name_town) VALUES (1987, ' ', 'GRELYON' );
```

Y a t-il une erreur ? Si oui corrigez et complétez.
Comparer les plans d'exécution de chacune de ces requêtes.

Question 2 :

Suite au projet de loi du gouvernement concernant la fusion des régions, nous allons procéder aux mises à jour de quelques régions sur nos bases de données de la manière suivante :

```
--base1--
UPDATE  REGIONS SET name = 'Rhône-Auvergne'  WHERE name = 'Rhône-Alpes' OR name = 'Auvergne'
--base2--
UPDATE  all_in_one SET name_region  = 'Rhône-Auvergne'  WHERE name_region = 'Rhône-Alpes' OR name_region = 'Auvergne'

--base1--
UPDATE  REGIONS SET capital = 690123  WHERE name = 'Rhône-Auvergne'
--base2--
UPDATE  all_in_one SET capital_region = 690123  WHERE name_region = 'Rhône-Auvergne'
```

Comparer les plans d'exécution de chacune de ces requêtes. Que constatez-vous ?

Question 3 :
Supprimez les insertions qui ont été faites dans la question 1.

```
DELETE FROM TOWNS WHERE name = 'GRELYON'

DELETE FROM all_in_one WHERE name_town = 'GRELYON'
```

Comparer les plans d'exécution de chacune de ces requêtes. Que constatez-vous ?.