SGBD : BASES DE DONNÉES AVANCÉES [M3106C]

TD $N^0 2$ - OPTIMISATION DE REQUÊTES SOUS POSTGRESQL

Objectifs

- Utilisation de EXPLAIN
- Notions de *tuning* des requêtes
- Méthodes d'accès

Enoncés

Exercice I :

(1) créer le schéma de base données décrit dans le script `schemao.sql`,

(2) créer une instance de ce schéma en utilisant le script `example_datao.sql`.

On considère la requête suivante :

```
-----------------------------------------------------
  SELECT dy_company, date(dy_timestamp) FROM diary;
-----------------------------------------------------
```

Question 1.1. *Générer le plan d'exécution pour cette requête, et expliquer ce plan.*

Le coût de l'opération `Seq Scan` est déterminé par :

$$Cout = Nombre\_de\_Blocs\_de\_la\_Relation + Nombre\_de\_Tuples\_de\_la\_Relation \times \texttt{CPU\_TUPLE\_COST}$$

augmenté de `CPU_OPERATOR_COST` $\times Nombre\_de\_Tuples\_de\_la\_Relation$ s'il y a un opérateur.

Question 1.2. *Recalculer ce coût en utilisant les informations de votre base de données.*

Question 1.3. *Comparer les plans d'exécution des deux requêtes suivantes en fonction de leur sélectivité. Que déduisez vous ? (note : interdire au planner d'utiliser le chemin d'accès* `bitmap scan`*)*

Date: 20 mars 2014.

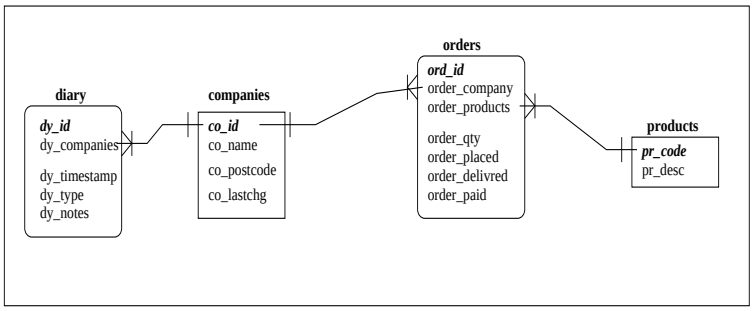
Hocine ABIR - IUT Villetaneuse .

```
-----------------------------------------------------
 SELECT dy_company, date(dy_timestamp)          -- Ri
  FROM diary
  WHERE dy_id >= 900;
-----
 SELECT dy_company, date(dy_timestamp)          -- Rs
  FROM diary
  WHERE dy_id >= 10;
-----------------------------------------------------
```

Question 1.4. *Générer de nouveaux plans d'exécution pour ces deux requêtes. Commenter vos nouveaux plans.*

Question 1.5. *Déterminer le coût de la requête* `Ri` *précédente si l'index n'est pas utilisé.Comparer le resultat a celui de la* **Question 1.4**.

On considère le modèle conceptuel de données suivant associé au schéma.



et la requête :

```
-----------------------------------------------------
SELECT co_name,pr_desc,ord_qty
 FROM companies,orders,products
 WHERE co_id=ord_company
 AND   pr_code=ord_product
 AND  ord_qty > 60;
-----------------------------------------------------
```

Question 1.6. *Traduire la requête* `SQL` *ci-dessus en Français.*

Question 1.7. *Pour chaque opérateur de jointure (Nested Loop, Merge, Hash Join), décriver graphiquement le plan d'exécution de cette requête (produit par Postgres).*

Question 1.8. *Dans le dernier plan obtenu à la* **Question 1.7**, *la jointure entre* `companies` *et* `orders` *est effectuée avant celle entre* `products` *et* `orders`. *Reécrire la requête de sorte que la jointure entre* `products` *et* `orders` *précède celle entre* `companies` *et* `orders`.

Exercice II :

On considère le schéma de base de données suivant :

```
create table Fournisseur
(
   fou_id         int primary key,
   fou_nom        varchar(30),
    fou_ville     varchar(30),
   fou_adresse    varchar(60)
);

create table Piece
(
   pie_id      int primary key,
   pie_nom     varchar(30),
   pie_stock   int,
    pie_prix   float,
   fou_id_fk   int
               references   Fournisseur (fou_id)
);
```

```
insert into fournisseur values
       (10,'Peugeot','Paris','Bd Diderot'),
       (20,'Renault','Amiens','Bd Zola');

insert into piece values
      (1,'Pneu', 11, 59 , 20),
      (2,'Huile', 20, 18 , 10);
```

et la requête :

```
-- Ra
select fou_nom,pie_nom
    from   piece  , fournisseur
     where  fou_id=fou_id_fk;
```

Deux plans sont possibles pour exécuter la requête |Ra| si l'on utilise uniquement la jointure par boucles imbriquées (de base) :

```
-- Plan FP
                                QUERY PLAN
-----------------------------------------------------------------------
 Nested Loop  (cost=1.02..2.13 rows=2 width=13)
   Join  Filter :  ( fournisseur .fou_id  =  piece.fou_id_fk)
   ->  Seq Scan on fournisseur  (cost=0.00..1.02 rows=2 width=12)
   ->   Materialize   (cost=1.02..1.04 rows=2 width=9)
         ->  Seq Scan on piece  (cost=0.00..1.02 rows=2 width=9)
(5 rows)
```

```
-- Plan PF
                                QUERY PLAN
-----------------------------------------------------------------------
 Nested Loop  (cost=1.02..2.13 rows=2 width=13)
   Join  Filter :  (piece.fou_id_fk  =  fournisseur.fou_id)
   ->  Seq Scan on piece  (cost=0.00..1.02 rows=2 width=9)
   ->   Materialize   (cost=1.02..1.04 rows=2 width=12)
         ->  Seq Scan on fournisseur
               (cost=0.00..1.02 rows=2 width=12)
(5 rows)
```

Question 2.1. *Quelle commande SQL doit-on écrire pour forcer l'optimiseur (planner) à générer le plan* `Plan FP`.

Question 2.2. *Quelle commande SQL doit-on écrire pour forcer l'optimiseur (planner) à générer le plan* `Plan PF`.

Question 2.3. *Donner le résultat de l'exécution du plan* `Plan FP` *(dans l'ordre).*

Question 2.4. *Donner le résultat de l'exécution du plan* `Plan PF` *(dans l'ordre).*

On considère les deux commandes suivantes :

```
-- C1
# explain  select * from x();
                        QUERY PLAN
-------------------------------------------------------------
 Function Scan on x  (cost=0.00..260.00 rows=1000 width=4)
(1 row)

```

```
-- C2
# select * from x();
 x
---
 1
 2
(2 rows)
```

Question 2.5. *Les résultats de /C1/ et /C2/ sont incohérents : Montrer cette incohérence.*

On considère maintenant la requête suivante :

```
explain  Select 2*pie_id +new_id as new, nom ,
    10 as pie_stock,90 as pie_prix,fou_id
  from x() as n(new_id),
       piece ,  fournisseur ,
       (values(10,'Pneu Keber'),
              (20,'Pneu Michelin')) t(i,nom)
  where fou_id=fou_id_fk
    and fou_id = i;
```

```
                        QUERY PLAN
-------------------------------------------------------------------
 Nested Loop  (cost=1.02..2.33 rows=4 width=44)
   ->  Nested Loop  (cost=1.02..2.23 rows=2 width=40)
         Join  Filter :  (piece.fou_id_fk =  fournisseur.fou_id)
         ->  Nested Loop  (cost=0.00..1.12 rows=2 width=44)
               Join  Filter :  (piece.fou_id_fk =
                                "*VALUES*".column1)
               ->  Seq Scan on piece
                     (cost=0.00..1.02 rows=2 width=8)
               ->  Values Scan on "*VALUES*"
                     (cost=0.00..0.03 rows=2 width=36)
         ->   Materialize   (cost=1.02..1.04 rows=2 width=4)
               ->  Seq Scan on fournisseur
                     (cost=0.00..1.02 rows=2 width=4)
   ->  Function Scan on x n  (cost=0.00..0.02 rows=2 width=4)
(10 rows)
```

Question 2.6. *Donner le résultat affiché par la requête /Rb/ ci-dessous après l'exécution du script suivant :*

```
insert into piece
(Select 2*pie_id +new_id as new, nom ,
    10 as pie_stock,90 as pie_prix,fou_id
 from x() as n(new_id),
      piece ,  fournisseur ,
       (values(10,'Pneu Keber'),
              (20,'Pneu Michelin')) t(i,nom)
 where fou_id=fou_id_fk
    and fou_id = i);
```

```
-- Rb
 Select * from piece;

-- Rc
  select  reltuples  from pg_class
       where relname='piece';
```

Question 2.7. *Donner le résultat affiché par la requête |Rc| ci-dessus. Justifier ce résultat !*