Optimisez vos requêtes DB2

par Jean-Alain Baeyens

Date de publication : 20/12/2006

Dernière mise à jour :

Si vos requêtes ne sont pas correctement optimisées, cela peut conduire à des temps de réponses catastrophiques une fois que le serveur entre en pleine charge. DB2 dispose d'un optimiseur qui fait une grande partie du travail. Toutefois, il ne pourra être efficace que si nous respectons certaines règles.

I - Que fait l'optimizer de DB2
II - Les index
III - Les bases de l'optimisation.
- A - Ne lire que ce qui est utile.
- B - Utiliser les index
- C - Utiliser l'Index Only Access
- D - Like plutôt que substr
- E - Veiller à la longueur des chaines
- F - Eviter les opérations arithmétiques
- G - Eviter les conversions
- H - Union et Union All
- I - L'opérateur différent
- J - Préciser le type d'isolation adéquat
- K - Les jointures
  - Inner join
  - 2 - Left join
  - 3 - Right join
- L - Les sous-requêtes

IV - Exemple concret
V - Conclusion

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

I - Que fait l'optimizer de DB2

Pour optimiser nos requêtes, DB2 dispose d'un *optimizer* qui va grandement nous faciliter la tâche. Il va choisir automatiquement l'ordre des opérations internes pour réaliser au mieux la commande. Il choisira d'utiliser ou non un index et si oui, il choisira le plus approprié. Mais l'*optimzer* va plus loin car il est capable de réécrire votre requête pour la rendre plus performante. Par exemple, il pourrait transformer une sous-requête en jointure.

Malgré cela, il est préférable de coder directement au mieux votre commande SQL et de respecter dans la mesure du possible un certain nombre de règles que nous allons voir.

> *Attention, l'optimizer est différent selon les versions et les plateformes utilisées. C'est pourquoi même si les tests effectués dans cet article ne confirment pas nécessairement le gain avec les règles énoncées, il est malgré tout préférable de les respecter.*

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

II - Les index

L'index permet la recherche rapide d'une valeur et le parcoure ordonné selon l'ordre de l'index. Il permet également la sélection et, ou, le tri rapide des enregistrements si la sélection et ou le tri portent sur l'ordre de l'index.

Prenons comme exemple une table contenant 4 colonnes, C1, C2, C3, C4. Il existe un index portant sur les colonnes C1, C2.

```
SELECT * FROM table WHERE c1='xxx'
SELECT * FROM table WHERE c1>'xxx'
SELECT * FROM table WHERE c1= 'xxx' AND c2 = 'yyy'
```

Ces 3 commandes vont permettre un parcoure via l'index.

Par contre,

```
SELECT * FROM table WHERE c2='yyy'
```

ne permet pas le parcoure ordonné.

Toutefois, l'index permettra malgré tout d'accélérer la recherche. DB2 pourra le parcourir séquentiellement au lieu de devoir lire séquentiellement le fichier. On parle alors d'*Index Scan*. DB2 accèdera à la table uniquement pour les valeurs correctes.

Il est également possible d'obtenir un grand gain de performance en utilisant l'*Index-Only Access*. En quoi cela consiste-t-il ? Comme son nom l'indique, la requête n'accèdera qu'à l'index et non à la table pour obtenir les données. Pour permettre ce type d'accès, toutes les données demandées doivent évidemment être reprises dans l'index.

Prenons un exemple, vous avez une table de contacts reprenant une trentaine de champs. L'Id du contact est sa clé primaire. Vous avez un module de recherche qui liste alphabétiquement les contacts par nom et prénom avec une sélection sur base du nom. Cette requête est évidement couramment utilisée. Vous devez également récupérer l'Id pour pouvoir ensuite pointer sur le contact choisi. Afin d'optimiser la sélection et le tri, vous avez créé un index sur le nom et le prénom. Votre requête sera donc du type:

```
SELECT nom, prenom, id FROM contacts WHERE nom like 'DUP%' ORDER BY nom, prenom
```

Pensez à ajouter l'id dans votre index. La requête précédente se fera alors en mode *Index-Only Access*. Evidemment, c'est à faire pour des requêtes couramment utilisées et il ne s'agit pas de dupliquer votre table dans l'index.

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

III - Les bases de l'optimisation.

L'optimisation commence avec des règles simples et évidentes mais qui sont malgré tout couramment bafouées.

Afin de mieux comprendre les différences, créons une base de données.

création de la table

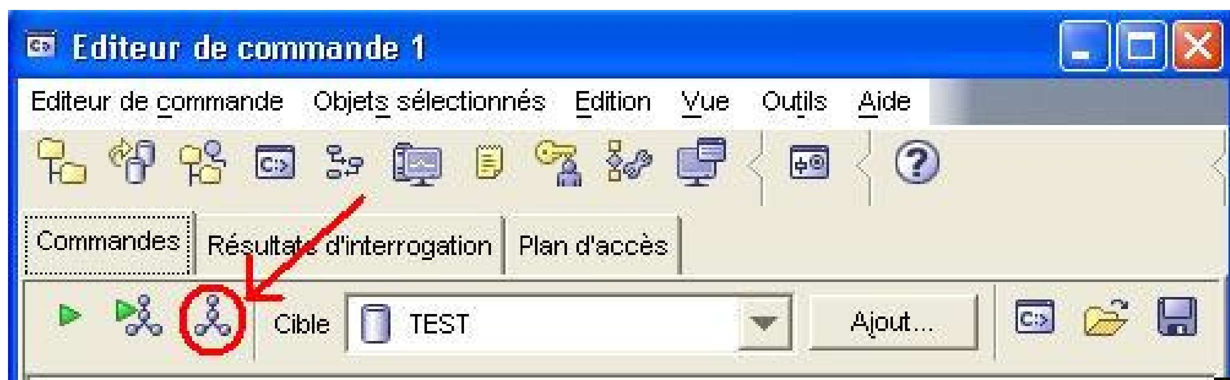
```
CREATE TABLE TEST.OPTIMIZE1
  ( ID BIGINT  NOT NULL  GENERATED ALWAYS AS IDENTITY
                        (START WITH 0, INCREMENT BY 1, NO CACHE )
     , W CHARACTER (7)  NOT NULL
   , X INTEGER  NOT NULL  WITH DEFAULT 0
   , Y VARCHAR (256)
   , Z VARCHAR (256)
     , CONSTRAINT pk_id PRIMARY KEY ( ID)  ) ;
```

création des index

```
CREATE  INDEX TEST.INDEX_X ON TEST.OPTIMIZE1 (X ASC, W ASC)
  PCTFREE 10 MINPCTUSED 10 ALLOW REVERSE SCANS  COLLECT  STATISTICS ;
CREATE  INDEX TEST.INDEX_W ON TEST.OPTIMIZE1 (W ASC)
  PCTFREE 10 MINPCTUSED 10 ALLOW REVERSE SCANS  COLLECT  STATISTICS ;
```

La table contient 100.000 enregistrements dans lesquels la colonne w contient 10 valeurs différentes réparties uniformément et la colonne x contient des valeurs de 0 à 100. Les colonnes y et z contiennent une valeur fixe pour donner une taille plus grande à chaque enregistrement et pour avoir un contenu non inclus dans les index.

Afin d'obtenir rapidement un bon indicateur de performance et pour mieux comprendre ce qui se passe, nous pouvons utiliser l'outil d'analyse fourni dans l'éditeur de commande.



L'outil d'analyse dans l'éditeur de commande DB2

A - Ne lire que ce qui est utile.

```
SELECT w,y FROM test.optimize1
```

sera évidemment plus rapide que

```
SELECT * FROM test.optimize1
```

En fait dans le premier cas et pour notre table de test, le temps de parcoure et d'affichage est de 44 secondes et 5

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

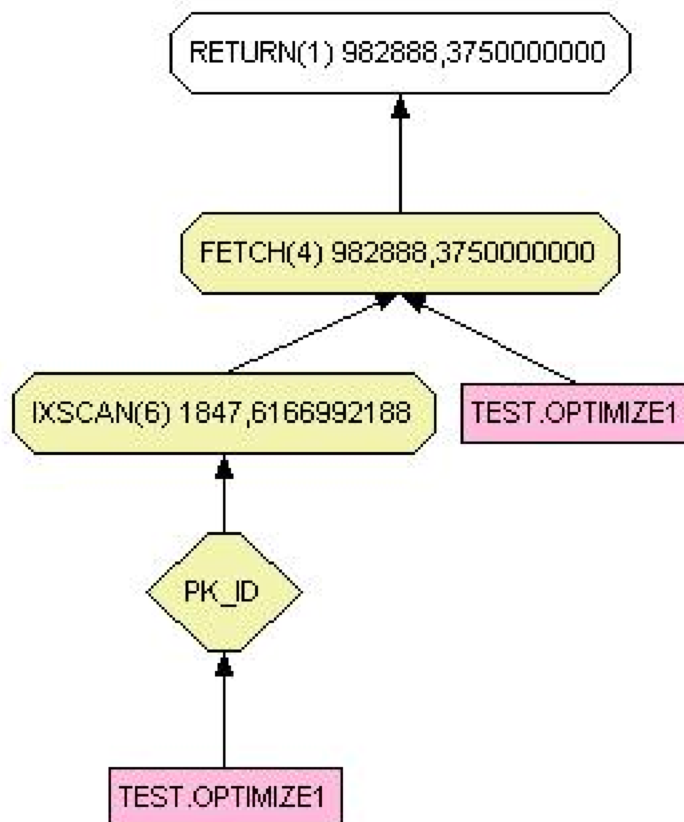
centièmes alors que dans le deuxième cas le temps de parcoure est de 3 minutes 30 secondes et 13 centièmes. Soit quasiment 5 fois plus lent. Pour partie la différence est due au temps d'affichage dans la boite de commande.

B - Utiliser les index

L'utilisation des index pour limiter les recherches est bien connu mais à titre d'exemple, voici les différences obtenues.

```
SELECT * FROM test.optimize1 WHERE w='GTX1000'
```

Sans l'index le chemin est le suivant:

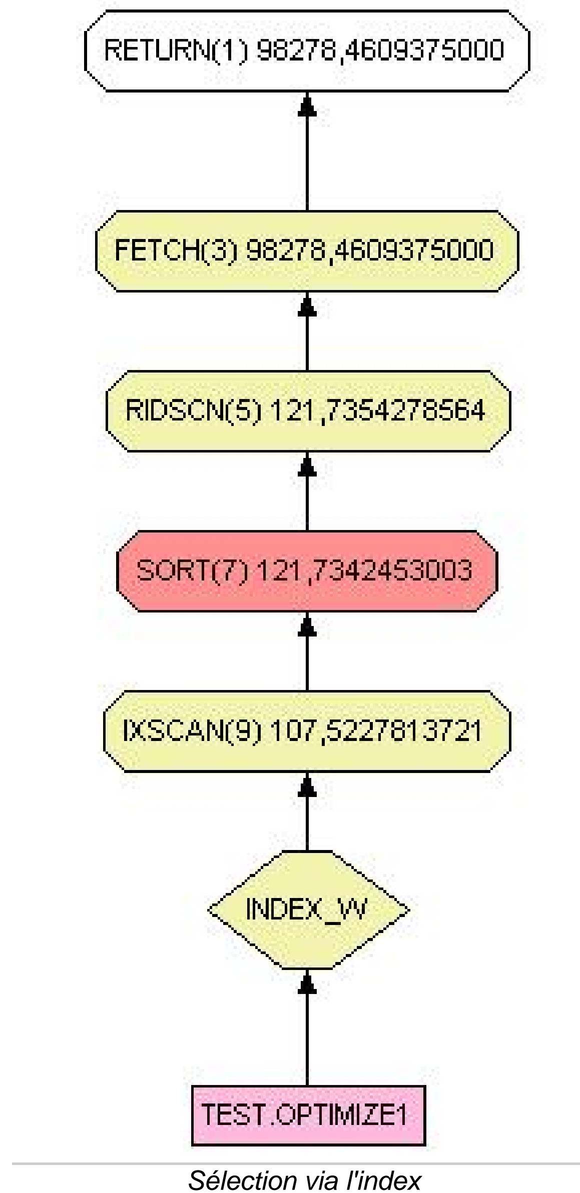


Sélection séquentielle dans la table

Notez que DB2 lit en premier l'index de la clé primaire et ensuite l'enregistrement de la table. Le résultat sera trié sur l'ordre défini par la clé primaire.

Avec l'index "Index_W" créé:

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.



Sélection via l'index

Le temps de parcoure est divisé par 10.

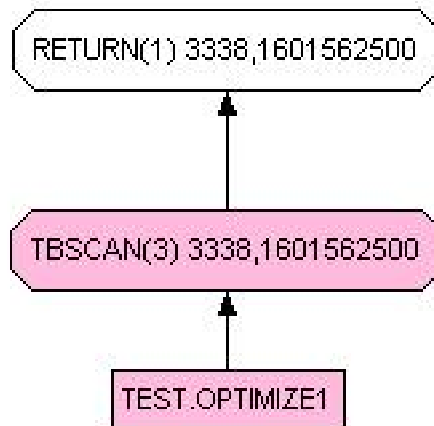
Ne multipliez pas les index à l'excès. Leurs maintenances a aussi un cout.

C - Utiliser l'Index Only Access

Ce principe à déjà été expliqué précédemment mais voyons ce que cela donne sur notre exemple.

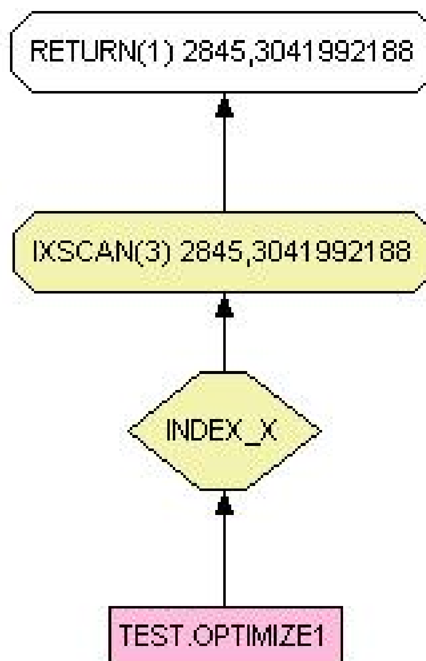
```
SELECT * FROM test.optimize1
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.



Parcoure séquentiel de la table

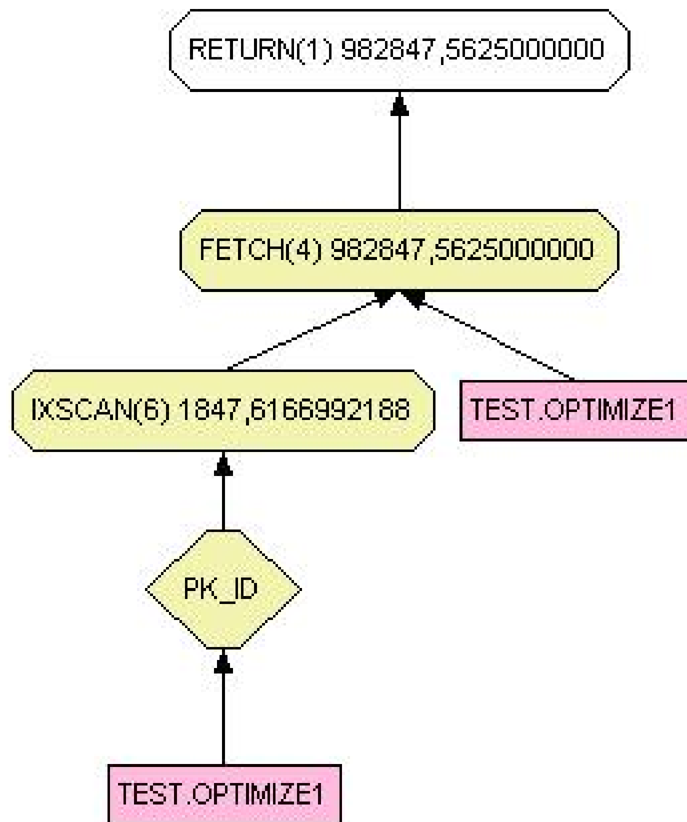
```
SELECT w,x FROM test.optimize1
```



Parcoure par Index Only Access

Dans se second schéma, il n'y a ni *table scan* ni *fetch*. La différence de performance est malgré tout faible d'autant qu'elle est aussi due à la lecture complète de l'enregistrement dans le premier Select. Si vos enregistrements son long, la différence sera beaucoup plus marquée.

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.



Parcoure sans Index Only Access

D - Like plutôt que substr

Imaginons une table ou w est le premier champ dans un des index.

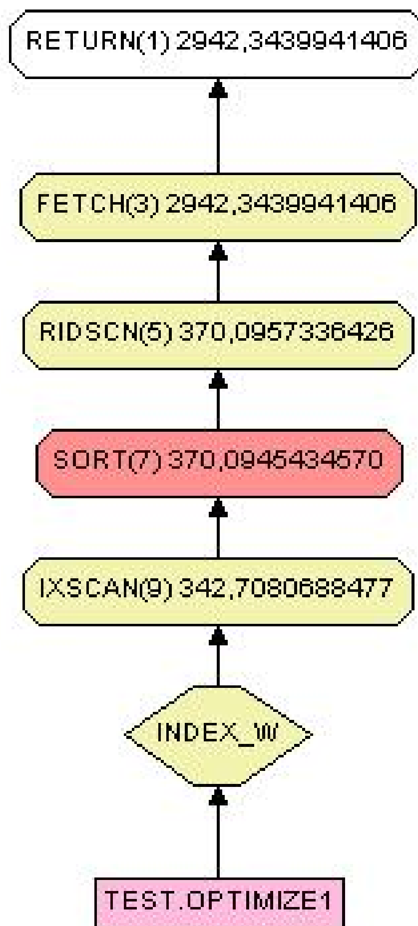
```
SELECT w,x FROM test.optimize1 WHERE substr(w,1,2) = 'GT'
```

est annoncé moins rapide que

```
SELECT w,x FROM test.optimize1 WHERE w LIKE "GT%"
```

Selon la documentation, les fonctions scalaires ou de concaténations ainsi que le casting rendent l'utilisation de l'index sur les champs concernés impossible. Si vous le pouvez, écrivez votre sélection différemment pour les éviter. Toutefois, lors des tests, le chemin présenté contredit cette affirmation qui est donc à nuancer. Il semble en effet qu'au minimum sur la version DB2 8.2 UDB le moteur est capable d'utiliser l'index.

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.



Le schéma de parcoure pour Like ou Substr

E - Veiller à la longueur des chaines

Lorsque votre commande SQL est le résultat d'un processus, vous pourriez vous retrouver avec une commande de ce type:

```
SELECT w,x FROM test.optimize1 WHERE w = 'GTX1000   '
```

Imaginez que w soit défini comme une colonne caractère de longueur 7 et qu'elle soit indexée. Pour les utilisateurs de DB2 z/os, il serait alors préférable de modifier le processus pour que la commande finale ait la forme suivante:

```
SELECT w,x FROM test.optimize1 WHERE w = 'GTX1000'
```

Pour les autres versions de DB2, il n'y a aucune différence de performance entre les deux formes.

Le schéma de parcoure obtenu dans les deux cas est le même que celui obtenu avec Substr et Like.

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

F - Eviter les opérations arithmétiques

Même si elles peuvent apporter plus de lisibilité au code parce qu'elles représentent une réalité business (ex: le salaire moins des frais forfaitaires doit être inférieur à un plafond) éviter les opérations arithmétiques dans les conditions.

```
SELECT w,x FROM test.optimize1 WHERE x - 10 < 50
```

La commande ci-dessus risque fort d'être pénalisante. Il est en effet possible que pour chaque enregistrement une opération arithmétique ait lieu. De plus, sur certaines versions DB2, l'index ne serait pas utilisé.

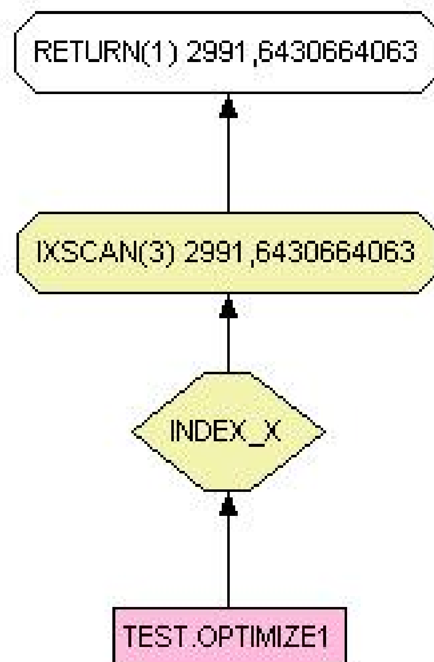


Schéma de parcoure avec opération

La syntaxe correcte est donc :

```
SELECT w,x FROM test.optimize1 WHERE x < 60
```

Toutefois, avec la version DB2 utilisée, le chemin et temps de parcoure sont les mêmes. L'optimiseur prend donc en charge la modification mais prudence, ce n'est peut être pas le cas avec toutes les versions de DB2.

G - Eviter les conversions

Non seulement il faut éviter les fonctions de conversion pour les motifs indiqués précédemment mais il faut également se méfier des conversions implicites.

Exemple
```
SELECT w,x FROM table WHERE w > 75000.00
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Imaginez que w est une colonne de type entier et qu'il existe un index sur la colonne. Dans ce cas, il y aura conversion. Avec DB2 UDB pour Linux/Windows ainsi qu'avec DB2 pour iSeries, les performances ne seront pas réellement affectée mais avec la version pour z/os il en serait autrement.

> *Il est évident que vous n'entrerez jamais ce genre de commande à la main mais n'oubliez pas que le commande SQL peut elle même être le résultat d'un processus et là...*

H - Union et Union All

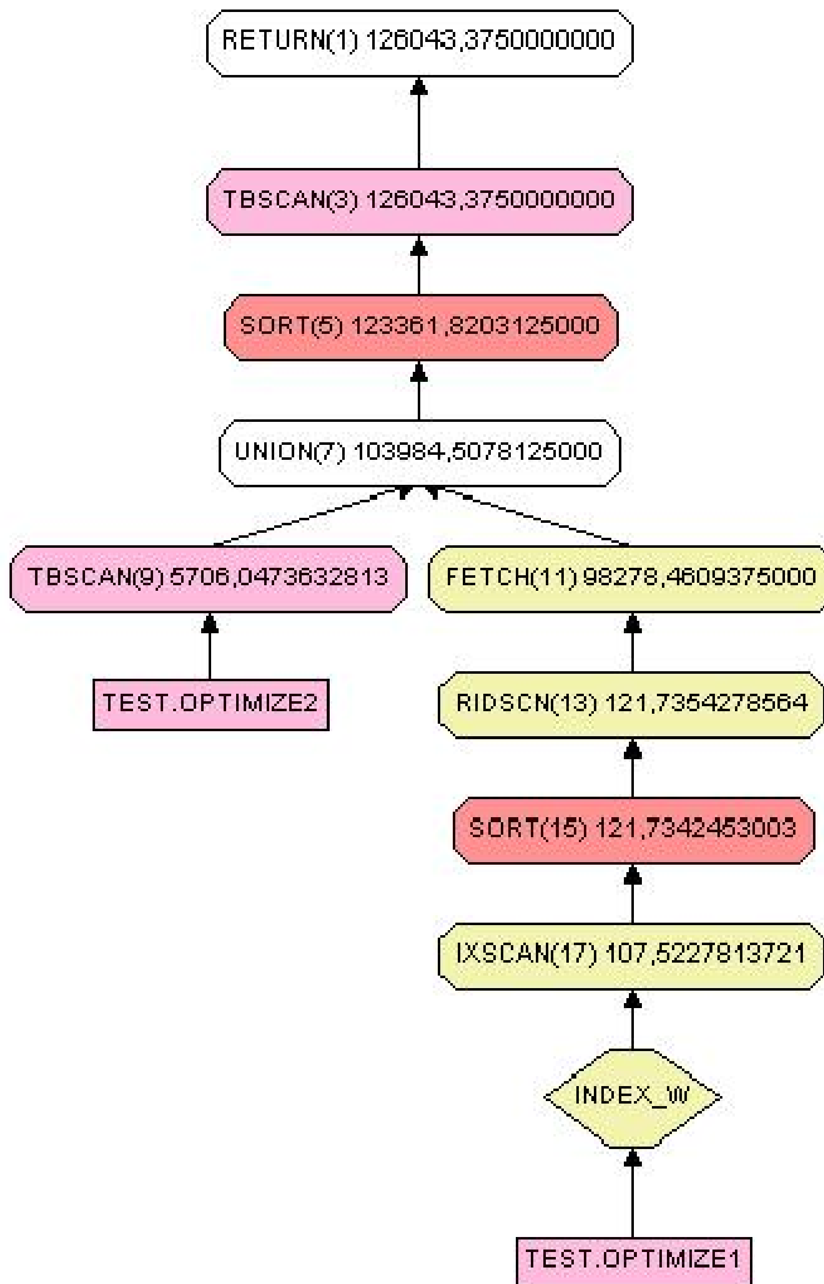
Si vous savez avec certitude que les commandes UNION et UNION ALL vont retourner le même résultat, utilisez UNION ALL. En effet, la commande UNION induit un tri pour éliminer les doublons. L'utilisation de UNION ALL va donc vous faire économiser ce temps.

Pour illustrer cela, utilisons une seconde table opimize2 avec la même structure et reprenant 25000 enregistrements.

Exemple

```
select * from test.optimize1 where w='FX7000'
union
select * from test.optimize2 where w='GFX1000'
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

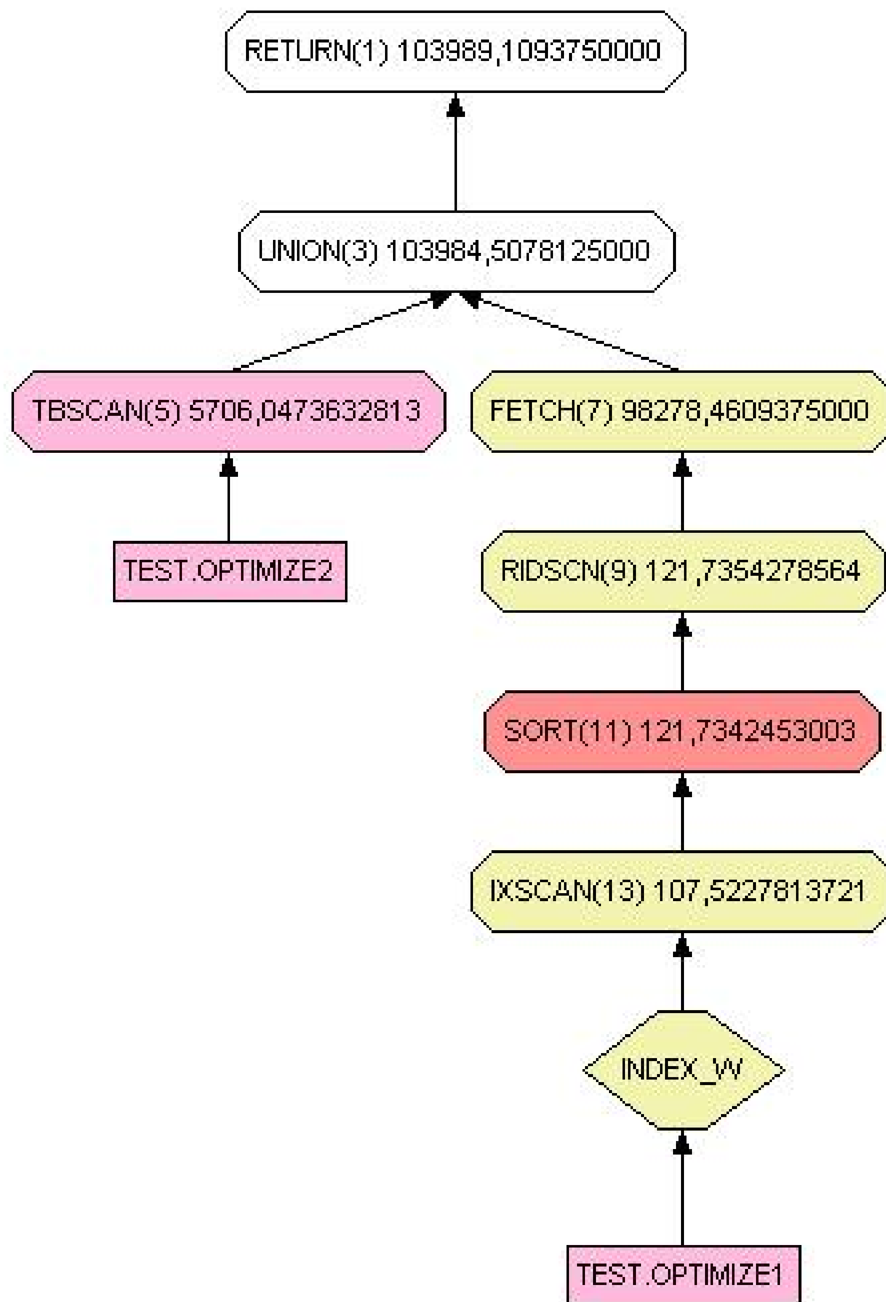


Avec Union

Exemple

```
select * from test.optimize1 where w='FX7000'
union all
select * from test.optimize2 where w='GFX1000'
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.



avec Union All

Comme vous pouvez le constater, dans le second cas, le renvoi du résultat ce fait directement après l'union. Dans l'exemple, le gain est de près de 18%.

> *Le traitement des deux fichiers diffère car optimize2 n'est pas indexé. Son traitement est toutefois plus rapide car sa taille est plus faible et le nombre d'enregistrement retournés plus petit.*

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

I - L'opérateur différent

Vous vous doutez qu'avec cet opérateur, DB2 n'utilisera pas d'index. Ce n'est toutefois pas totalement exact. DB2 UDB pour Linux/Windows pourra dans certaines circonstances l'utiliser. C'est l'optimizer qui va décider.

Exemple
```
SELECT x,y,z FROM table WHERE id <> 50
```

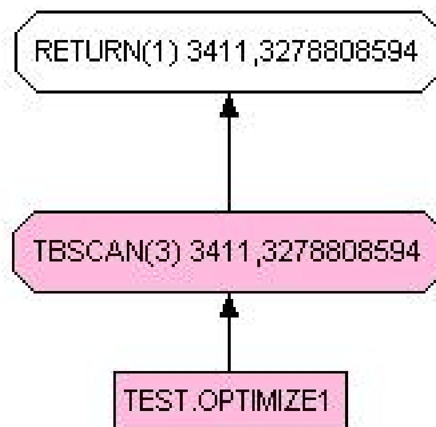


Schéma de parcoure sans index

Exemple
```
SELECT x,y,z FROM table WHERE w <> 'GTX5000'
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

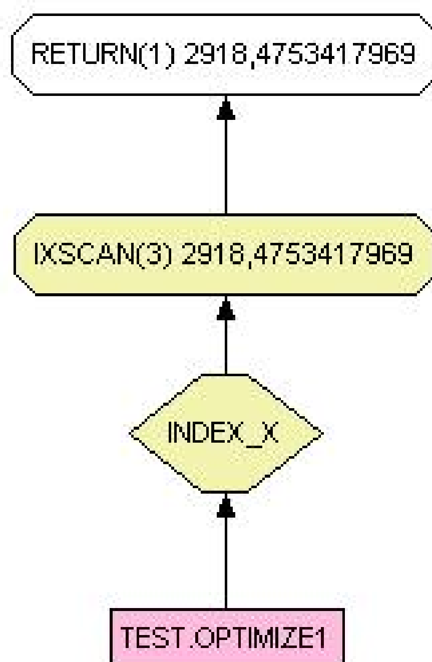


Schéma de parcoure avec index

J - Préciser le type d'isolation adéquat

Par défaut, mais cela dépend de la configuration du serveur, vos requêtes vont généralement provoquer du *locking*. Ce mécanisme prend évidemment des ressources systèmes et mémoires qui peuvent pénaliser le serveur. Il existe plusieurs types de *locking* qui dépendent du niveau d'isolation. Choisissez celui qui est le plus adapté à votre requête et à vos besoins.

Le niveau d'isolation *Uncommited Read* est absolument à préciser pour toutes vos requêtes pour lesquels il n'y aura pas de modification des données. (Select en vue d'un rapport, d'une liste, d'une fonction de recherche,...)

```
SELECT x,y,z FROM table WHERE w = 'GTX2000' WITH UR
```

K - Les jointures

Bien que vous ne puissiez pas y faire grand chose, le type de jointure à un impacte directe sur les performances. Regardons de plus près les schémas de parcoure pour les différents types de jointure.

Inner join

Inner Join

```
select * from test.optimize1 inner join optimize2 on optimize1.id = optimize2.id
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

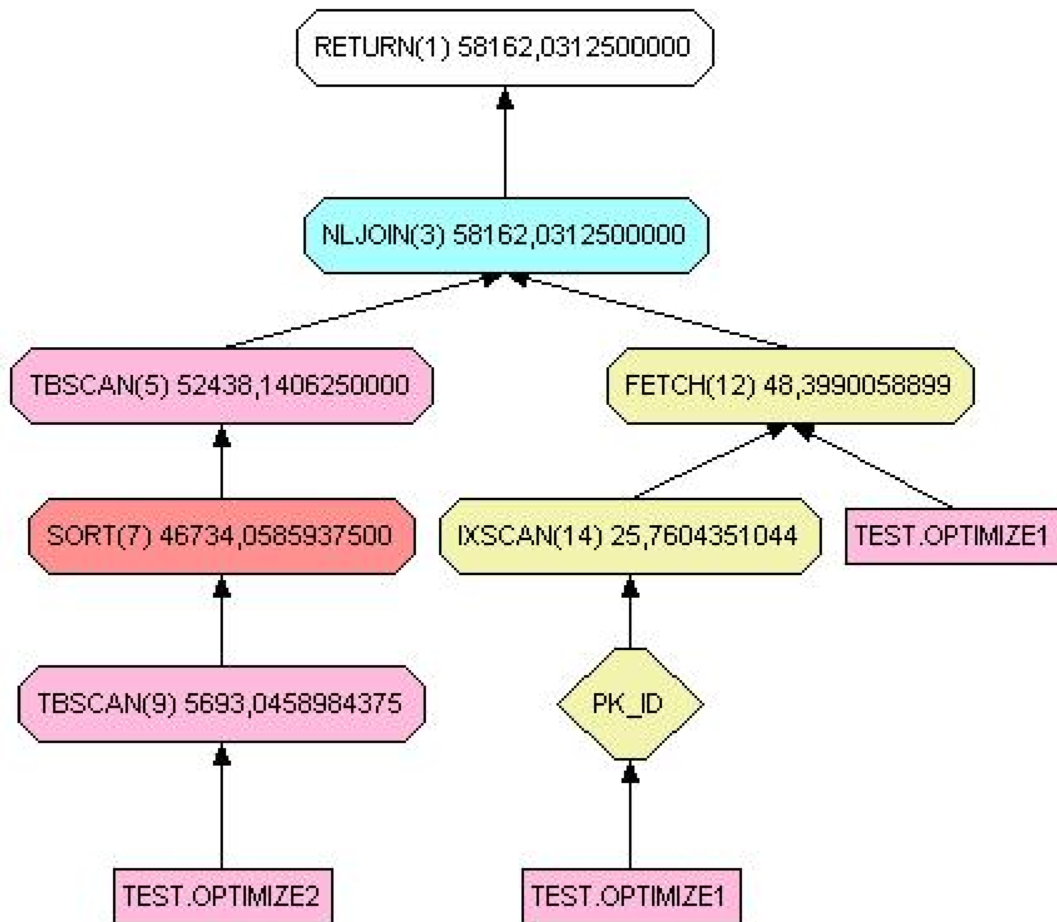


Schéma de parcoure pour inner join sans index

La jointure est faite sur la clé primaire.

Inner Join (Optimize2 n'est pas indexé sur w)

```
select * from test.optimize1 inner join optimize2 on optimize1.w = optimize2.w
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.
http://jab.developpez.com/tutoriels/db2/optimisation

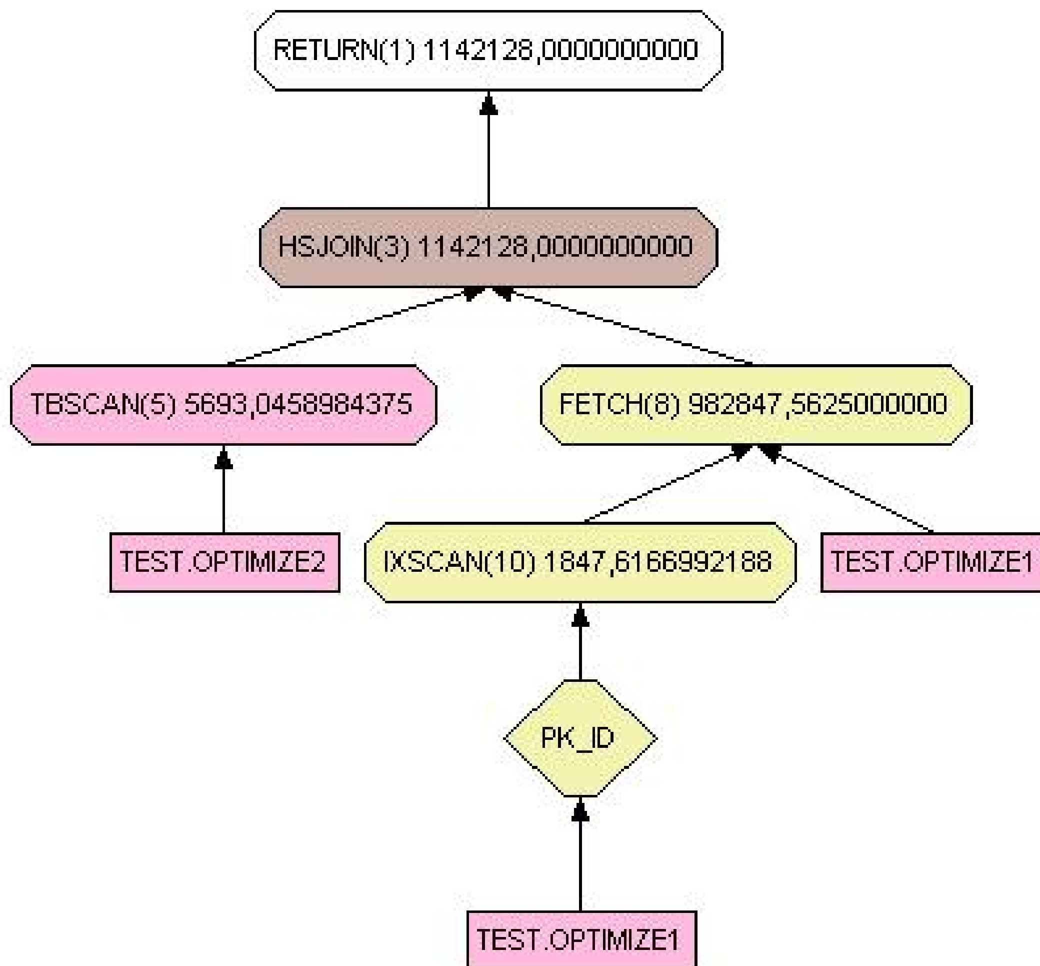


Schéma de parcoure pour inner join avec un index

Vous pouvez constater une grande différence entre les deux schémas. Cela est du au fait que dans le premier cas, il s'agit d'un index unique.

L'utilisation d'un index sur chacune des tables n'influence pas le chemin de parcoure pour la jointure. En effet, il faudra de toute façon parcourir séquentiellement une des deux tables.

2 - Left join

Left join

```
select * from test.optimize1 left join optimize2 on optimize1.id = optimize2.id
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

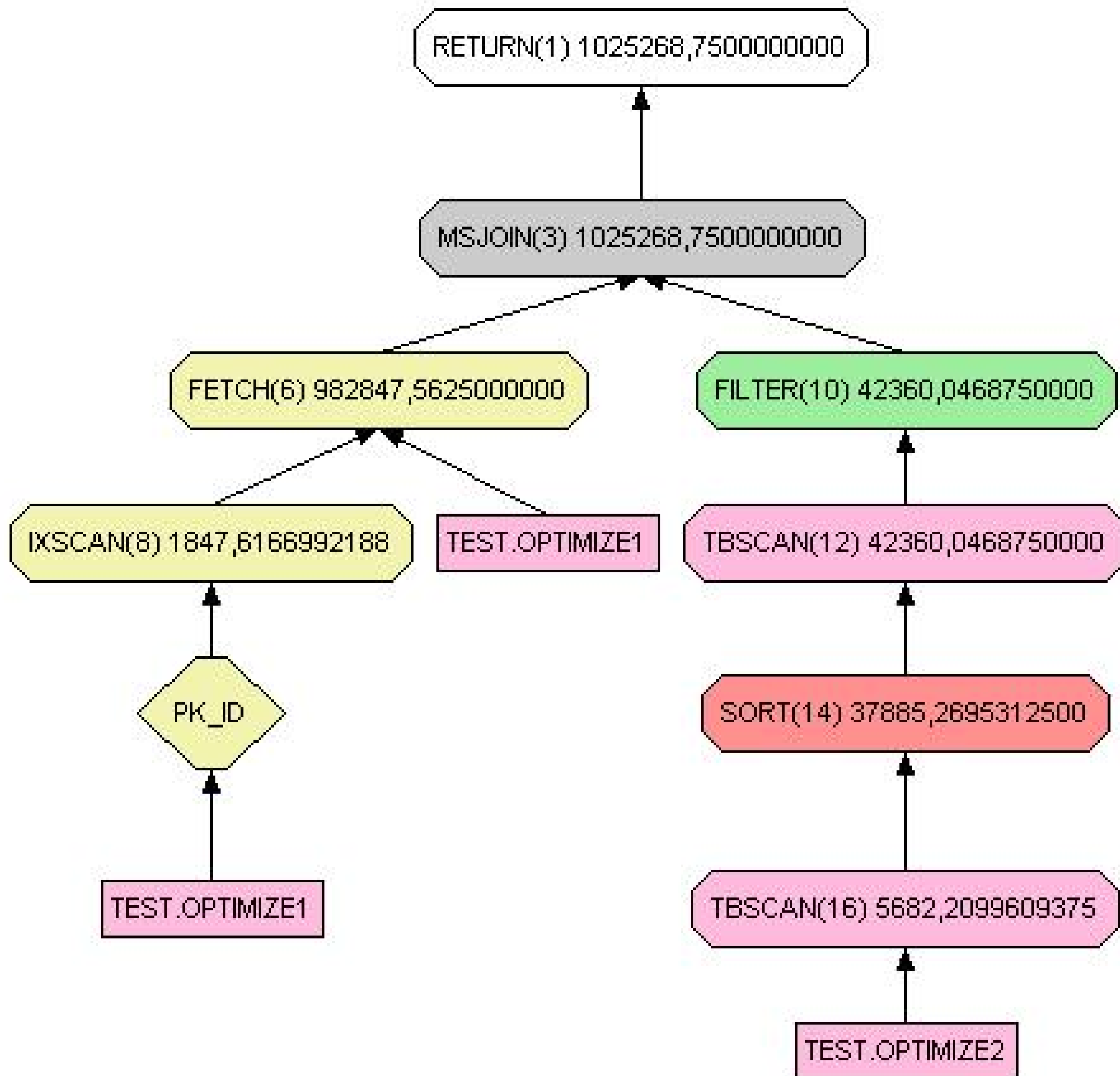


Schéma de parcoure avec une clé unique

Left join

```
select * from test.optimize1 left join optimize2 on optimize1.w = optimize2.w
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

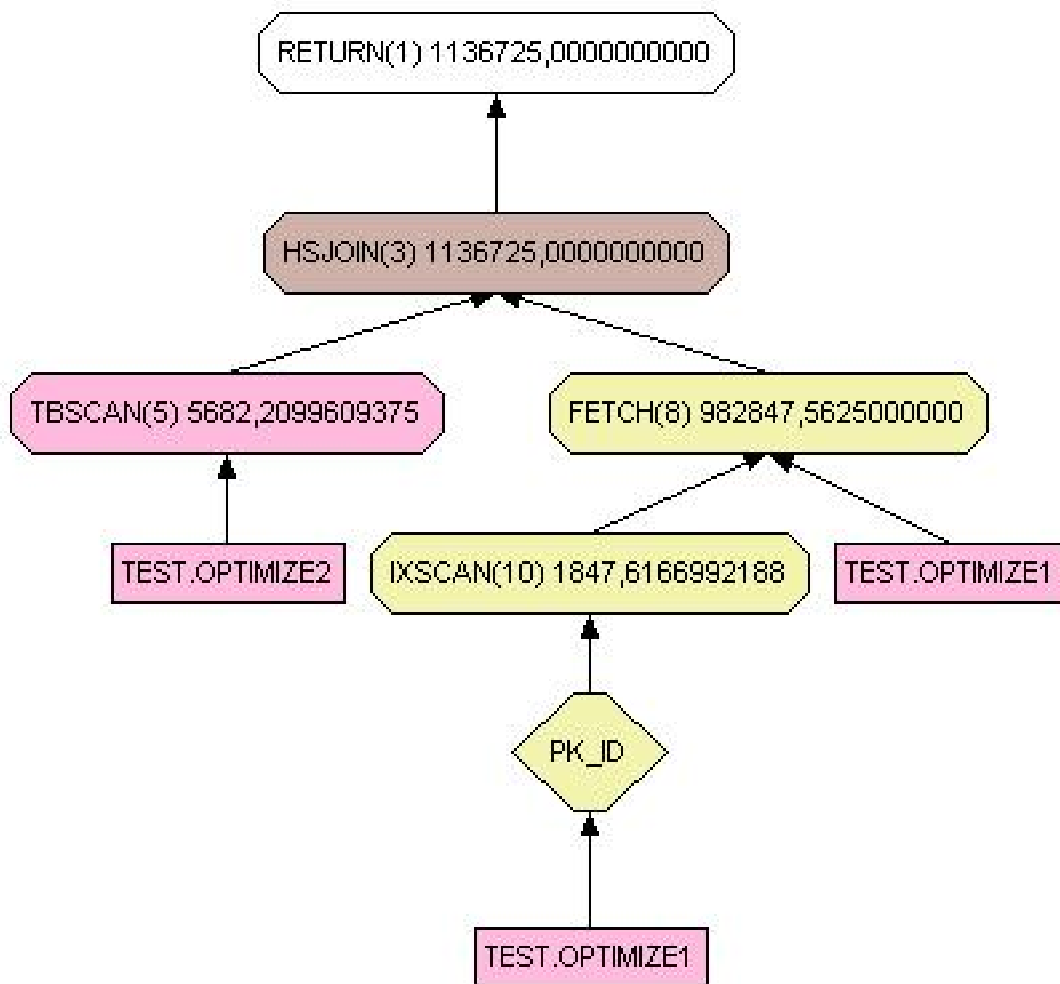


Schéma de parcoure sans clé unique

3 - Right join

Right join

```
select * from test.optimize1 right join optimize2 on optimize1.id = optimize2.id
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

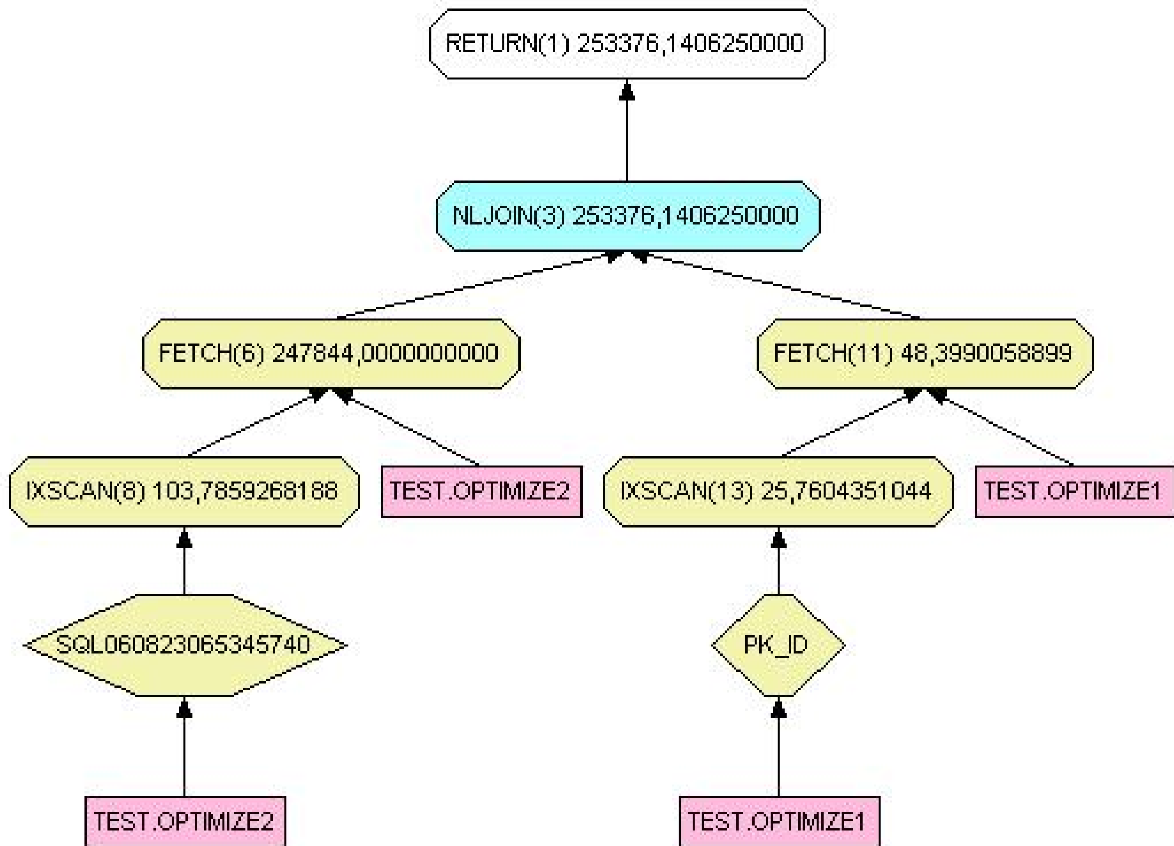


Schéma de parcoure avec clé unique

Right join

```
select * from test.optimize1 right join optimize2 on optimize1.w = optimize2.w
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

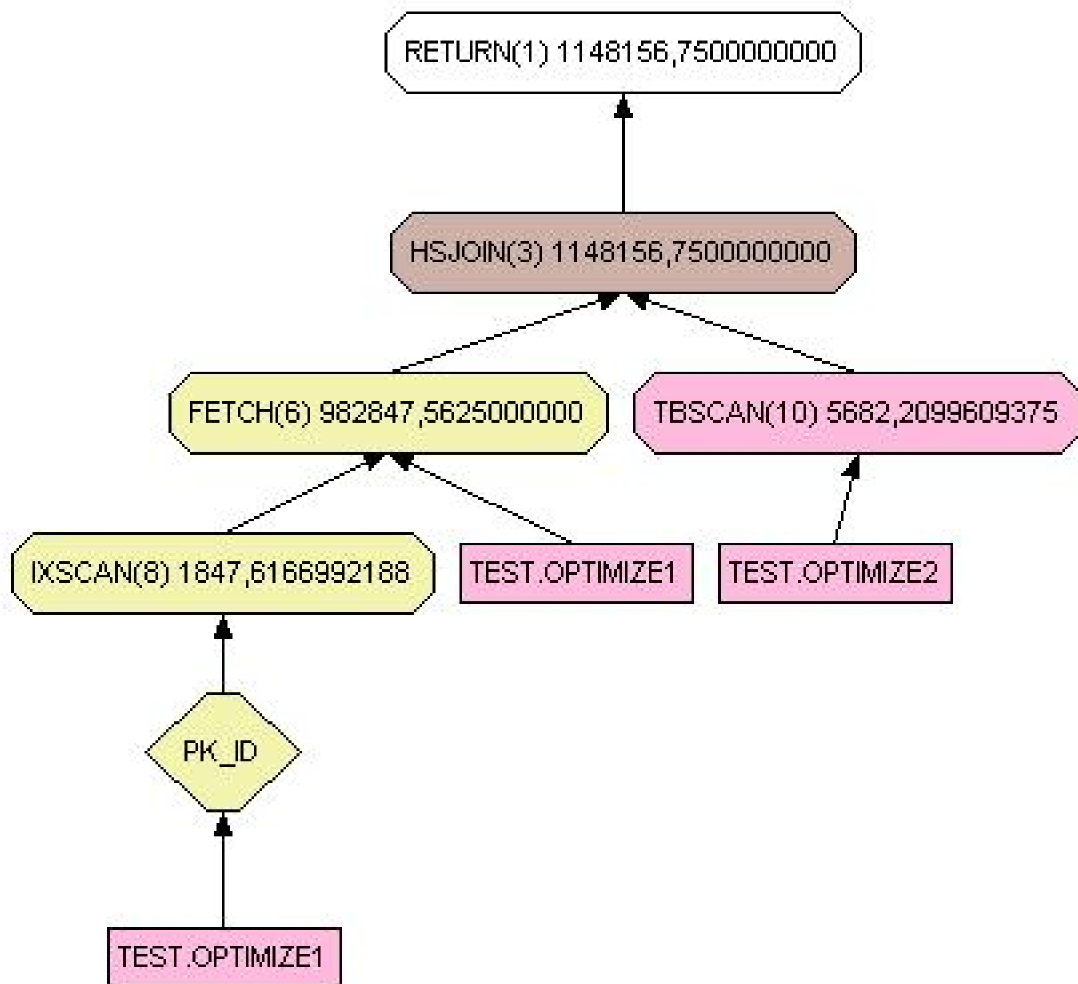


Schéma de parcoure sans clé unique

Pour les 3 types de jointure, si aucune clé unique quelle soit primaire ou non ne peut être utilisée, le schéma sera similaire. Par contre avec une clé unique les 3 schémas sont clairement différents. Sans index, le temps de réponse deviendrait vite catastrophique.

L - Les sous-requêtes

Comme vous pouvez le constater dans le schéma ci-dessous, DB2 va réaliser une jointure avec le résultat de la sous-requête.

Utilisation d'une sous-requête

```
select * from test.optimize1 where id in (select id from test.optimize2)
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

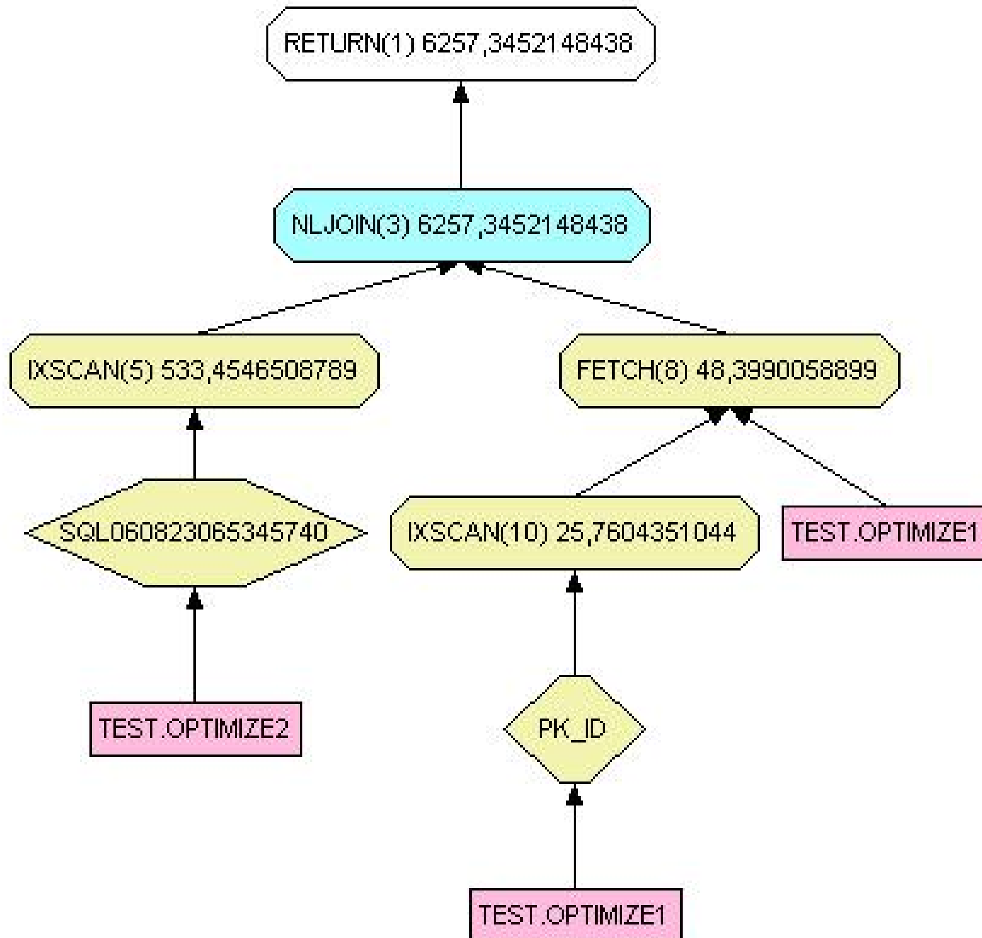


Schéma de parcoure avec une sous-requête

Que ce serait-il passé si nous avions directement réalisé la jointure ?

Exemple

```
select * from test.optimize1,test.optimize2 where optimize1.id = Optimize2.id
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

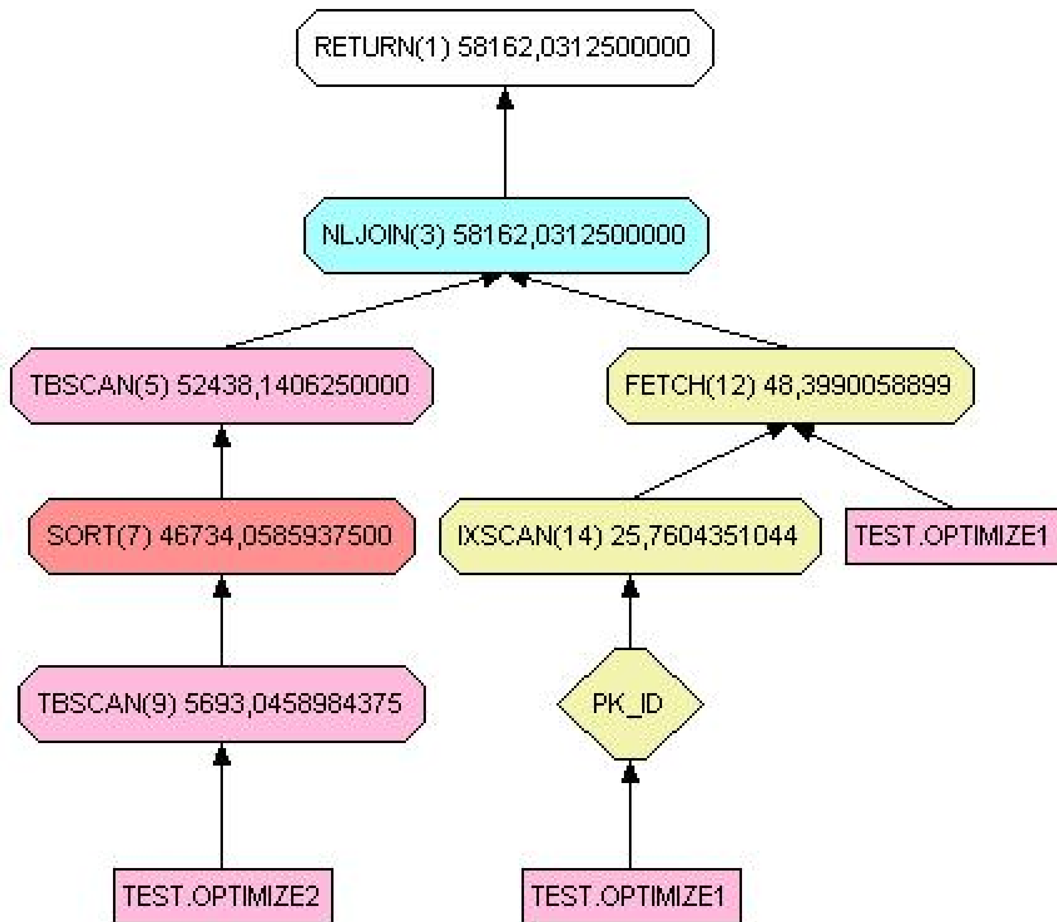


Schéma de parcoure avec join

Etonnamment, la jointure directe s'avère nettement moins performante. Soyez donc vigilant dans vos choix.

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.
http://jab.developpez.com/tutoriels/db2/optimisation

IV - Exemple concret

Afin d'illustrer cet article, voici un exemple tiré de notre forum. Il s'agit d'une simple sélection sur une table. Le nombre de données retournées est relativement faible mais la base de données contient beaucoup d'enregistrements.

Problème

```
SELECT A.FIELD1 , A.FIELD2 , A.FIELD3, A.FIELD4, A.FIELD5
FROM TABLE A
WHERE A.FIELD1 = x
AND A.FIELD2 >= y
AND A.FIELD3 >= z
AND (
    (A.FIELD4 = w AND A.FIELD5 = 'val')
     OR
    (A.FIELD4 = v AND A.FIELD5 <> 'val')
   )
ORDER BY A.FIELD1, A.FIELD2, A.FIELD3
```

La clé primaire est composée de Field1, Field2 et Field4.

Cette requête pourtant assez simple s'avère très peu performante et entraine un long temps d'attente.

Quel est le problème ?

Si vous avez bien suivi les explications précédentes, vous comprendrez aisément que le problème vient du OR et de l'impossibilité d'utiliser complètement les index. En effet, les deux premiers termes de la condition permettent une sélection sur l'index. Cette sélection via l'index devrait encore renvoyer une sélection plus étroite en utilisant la condition sur field4 mais la condition ou défini deux plages différentes.

L'idée pour améliorer la requête est donc d'optimiser l'utilisation des index. Pour cela nous allons diviser la requête en deux et rassembler les résultats. De plus comme les deux conditions ainsi créés sont mutuellement exclusives, nous pouvons nous contenter de faire une union sans fusion. Au finale, le temps de réponse a été divisé par 10.

> *Il s'agissait de DB2 MVS avec une application J2EE. Avec d'autres versions DB2, il n'aurait peut être pas été nécessaire de faire cette optimisation, l'optimizer s'en serait chargé à votre place.*

Solution

```
SELECT *
FROM (
        SELECT A.FIELD1 , A.FIELD2 , A.FIELD3, A.FIELD4, A.FIELD5
                FROM TABLE A
                WHERE A.FIELD1 = x
                      AND A.FIELD2 >= y
                      AND A.FIELD3 >= z
            AND A.FIELD4 = w
            AND A.FIELD5 = 'val'
        UNION ALL
        SELECT A.FIELD1 , A.FIELD2 , A.FIELD3, A.FIELD4, A.FIELD5
           FROM TABLE A
           WHERE A.FIELD1 = x
             AND A.FIELD2 >= y
             AND A.FIELD3 >= z
             AND A.FIELD4 = v
             AND A.FIELD5 <> 'val'
        ) AS TEMP
ORDER BY A.FIELD1, A.FIELD2, A.FIELD3
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

V - Conclusion

Comme nous avons pu le voir surtout pour des requêtes complexes qui sont les plus susceptible d'entrainer des problèmes de performance, il n'est pas évident de déterminer d'embler la meilleur requête. En dehors des règles fondamentales, le filling et l'expérience du développeur sera son meilleur atout. Dans les cas complexes, il est bon d'envisager plusieurs approches et de les tester avec un jeu de données vraisemblables aussi bien en quantité qu'en qualité.

L'optimisation est importante et trop souvent négligée. Généralement le développeur se contentera d'optimiser les requêtes longues mais n'oubliez pas que même une petite différence peut lors de la montée en charge avoir de grands effets. N'oubliez pas également que vous n'êtes vraisemblablement pas le seul à utiliser le serveur. Alors optimiser les requêtes complexes mais également les requêtes fréquemment utilisées.

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - J-A. Baeyens. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.