

Il n'est pas aisé de se lancer dans un chantier d'optimisation, car le contexte n'est généralement pas simple et très rarement identique d'un client à l'autre.

Ce qui est dommage, c'est qu'au lieu d'anticiper on en arrive toujours à mettre des chantiers d'optimisations en place lorsqu'apparaissent des traitements d'alimentation ou des « fenêtres de chargement » de plus en plus longs.

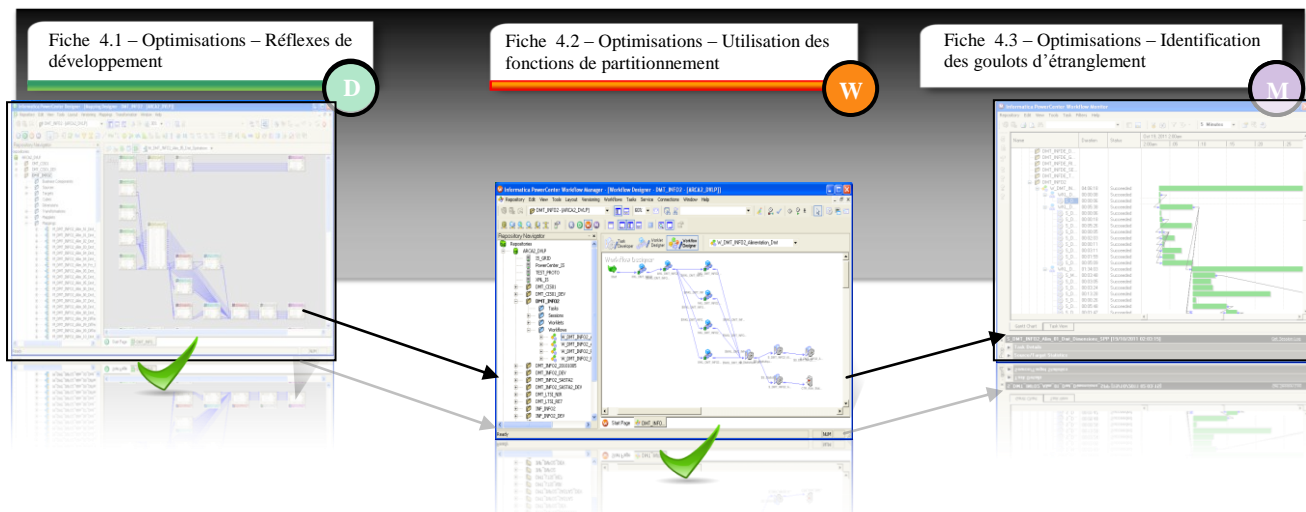
Après audit des environnements on s'aperçoit souvent qu'une majorité des problèmes auraient pu être évitée, si l'on avait suivi quelques conseils, qui s'apparentent plus à de la logique qu'à de l'expertise.

Dans ce Best practice, nous n'avons pas vocation à traiter ici des problématiques d'architecture, de machine ou encore de réseau. Bien que ceux-ci fassent partie intégrante d'un chantier optimisation, nous traiterons de ces sujets ultérieurement, avec notamment une fiche spécifiquement orientée hardware mettant en avant l'impact que peut avoir un serveur sur les performances des traitements de batches ou encore de requêtes : en effet, les ressources d'une machine ne se limitent pas aux seuls CPU et RAM.

L'objectif de ce document, c'est de se concentrer sur les principaux modules de développements d'Informatica portant sur les phases:

- ✔ de développement des mappings [Designer],
- ✔ d'ordonnancement des traitements avec l'option de partitionnement [Workflow manager] et enfin
- ✔ d'analyse de l'exécution des traitements [Workflow monitor].


Ces sujets se déclineront en 3 fiches, dont vous retrouverez la première (fiche 4.1 - optimisation - réflexes de développement) sur le site [www.decideo.fr](http://www.decideo.fr) ou sur notre site [www.unovia.fr](http://www.unovia.fr)



Vous trouverez dans cette fiche [4.2 - optimisation - utilisation des fonctions de partition](#) une rapide description des notions de partition et des différents types de partition à utiliser selon le contexte dans lequel vous vous trouvez.

Ces documents sont le fruit d'experts provenant de personnes provenant du milieu des SSII ou encore d'indépendants. Sans eux, ces fiches n'existeraient pas.

Merci à chacun d'entre eux pour leur contribution.

  
AUTEUR



Christophe Fournel



Fabien Duprey



Mario Gradel



Funji Matemou



Joël Blandin





SOMMAIRE

<b><u>INTRODUCTION</u></b> .....	<b>3</b>
<b><u>LES PARTITIONS DE BASE DE DONNEES</u></b> .....	<b>3</b>
A. Le principe .....	3
B. Les notions.....	3
C. Les types de partitions .....	4
<b><u>LES PARTITIONS DANS INFORMATICA</u></b> .....	<b>5</b>
A. Le principe.....	5
B. Les notions.....	6
1. De pipeline.....	6
2. De threads, partitions, et points de partitions.....	7
les threads .....	7
Les partitions .....	8
Les points de partitions .....	9
C. Les types de partitions .....	10
1. Définition.....	10
Partition Types – Pass Through .....	10
2. Exemple d'utilisation des types de partition .....	12
D. Retour d'expérience .....	13
1. Durée de traitement .....	13
2. Bug rencontré .....	14
3. Erreur récurrente.....	15
<b><u>CONCLUSION</u></b> .....	<b>15</b>
<b><u>DOCUMENTS DE REFERENCE</u></b> .....	<b>16</b>



---

## INTRODUCTION

---

Lorsque l'on parle de « partition », il n'est pas rare de confondre l'option de partition de la base de données, avec l'option de partition disponible sous Informatica PowerCenter.

Ces deux options sont en fait complémentaires et ; selon leur pertinence ; peuvent être toutes les deux utilisées dans la mise en place d'une solution d'optimisation.

Cette fiche traite principalement l'option de partitionnement d'Informatica, mais vous trouverez ci-dessous une brève explication des partitions de bases de données. Le but est de permettre à ceux qui ne connaissent pas ou peu les partitions de bien différencier les deux notions.

---

## LES PARTITIONS DE BASE DE DONNEES

---

### A. LE PRINCIPE

---

En base de données, lorsque les volumétries commencent à devenir importantes (plusieurs dizaines de millions de lignes), il est recommandé de partitionner les tables volumineuses. Cette fonctionnalité existe chez différents éditeurs de base de données. A titre indicatif, cette méthode de tuning a été introduite chez Oracle (en option) dans la version 8i.

### B. LES NOTIONS

---

Absolument nécessaire dans les environnements de base de données haute performance et haute disponibilité, la partition permet de fractionner des tables et des index en « morceaux » plus petits facilitant ainsi leur gestion : selon le critère de partitionnement, Oracle n'interrogera ou n'effectuera ses transactions que sur la(les) partition(s) concernée(s).

En pratique, le développeur ne verra qu'une seule table sous son outil client (SqlPlus, Toad, SqlDeveloper, etc.), alors que le moteur Oracle en verra plusieurs.

Pour le développeur, il est cependant possible « de voir » les différentes partitions qui composent une table en allant lire les métas-données, comme le font les outils clients. A titre d'exemple, sous Oracle, pour voir les partitions et/ou sous partitions, vous pouvez :

Dans l'outil client, regarder dans les propriétés de la table,

Ou exécuter une requête sur les « tables » `user_tab_partitions` (liste des tables et de leurs partitions associées) ou `user_tab_subpartitions` (liste des tables, de leurs partitions et sous partitions associées).

Cette notion s'étend également aux index qui peuvent être locaux ou globaux, respectivement propres à la partition ou à l'ensemble de la table. Pour encore plus de performances, chaque partition (voire sous-partition) pourra être stockée sur son propre tablespace.



## C. LES TYPES DE PARTITIONS

---

Le partitionnement d'une table s'effectue sur un ou plusieurs critères. En fonction de la typologie des données (type de données, occurrences, etc.), il existe plusieurs types de partitions. Sous Oracle, trois types de déterminations (primaires) de la partition existent :

- ✓ La partition de type **List** : on associe une partition à une (des) valeur(s) de la colonne (à utiliser dans le cas où l'on est en présence d'un nombre réduit de valeurs, comme une catégorie par exemple).
- ✓ La partition de type **Hash** : les partitions sont attribuées automatiquement par Oracle, en distribuant équitablement les lignes par un algorithme de hashing sur une colonne.
- ✓ La partition de type **Range** : on associe une partition à une plage de valeurs de la colonne (exemple : des pourcentages).

La partition de type **Composite** permet de combiner deux méthodes de partitionnement (sur des colonnes différentes) ; mais le partitionnement primaire doit être LIST ou RANGE, le secondaire est HASH ; on parle alors de partitions associées à des sous-partitions.

A partir de ces tables, toutes opérations classiques sur les tables restent compatibles (vue, trigger, etc.).

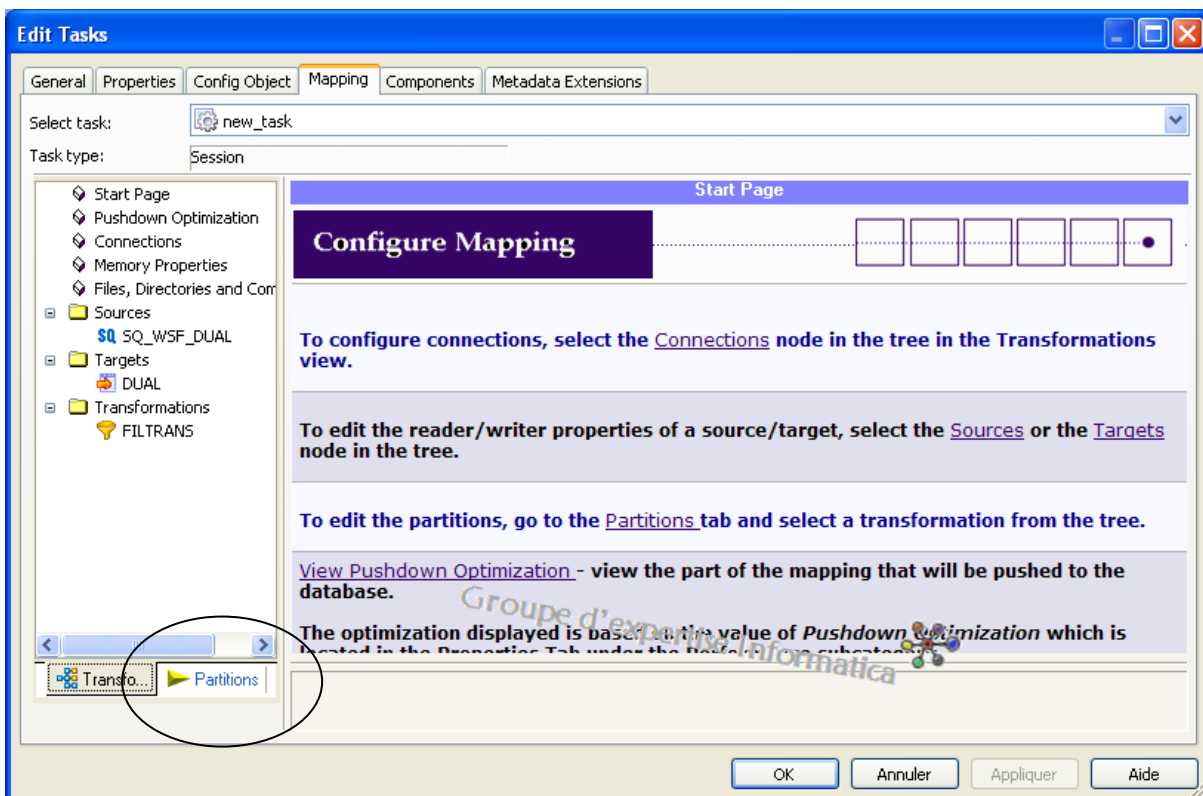


## LES PARTITIONS DANS INFORMATICA

### A. LE PRINCIPE

Tout comme pour la base de données, l'utilisation de la fonction de partitionnement dans Informatica est en option. Je ne saurais pas dire de quand date la fonctionnalité, cependant, cette option existait déjà lorsque j'utilisais la version 5 d'Informatica.

Celle-ci est visible dans toutes nouvelles sessions créées dans le Workflow Manager (ci-dessous une copie d'écran Informatica v9.1).



L'idée, via cette option de partitionnement, c'est de « saucissonner » une session en morceaux, et de permettre la parallélisation dans l'exécution de ces morceaux.

Pour cela, dans un premier temps, il est important de bien comprendre les notions **de pipeline, de partition, de points de partition**, puis, nous verrons dans un second temps, **les différents types** de partitions existantes sous Informatica.

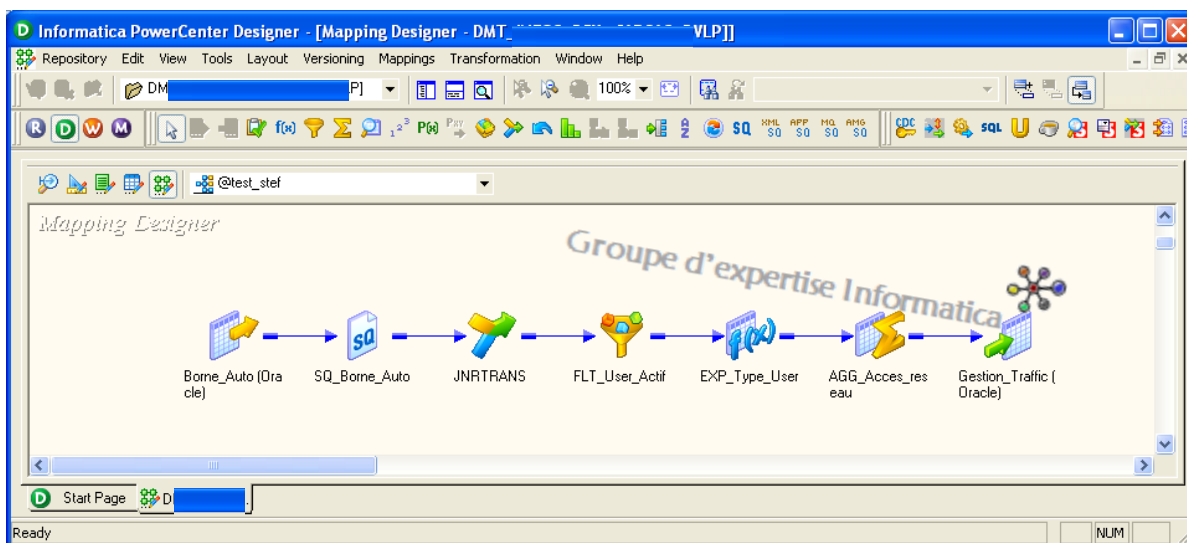


## B. LES NOTIONS

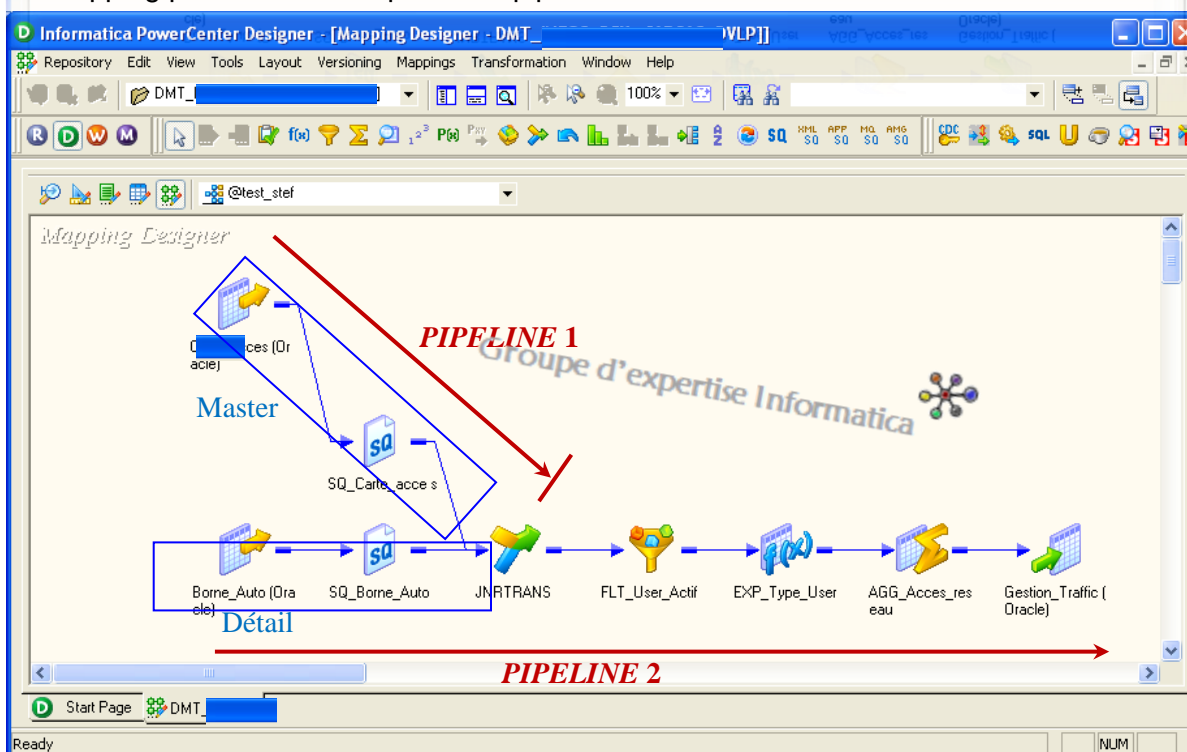
### 1. DE PIPELINE

On peut décomposer un mapping en *pipeline*, avec:

- ✓ Un source qualifier,
- ✓ Le(s) source(s) qui alimente(nt) le source qualifier,
- ✓ Toutes les transformations
- ✓ Le(s) cibles qui reçoivent les données provenant du source qualifier.



Un mapping peut avoir un ou plusieurs pipelines:

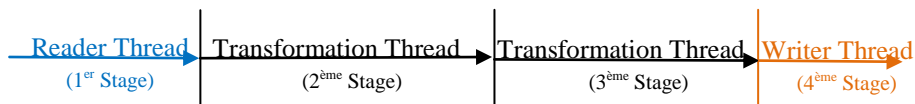
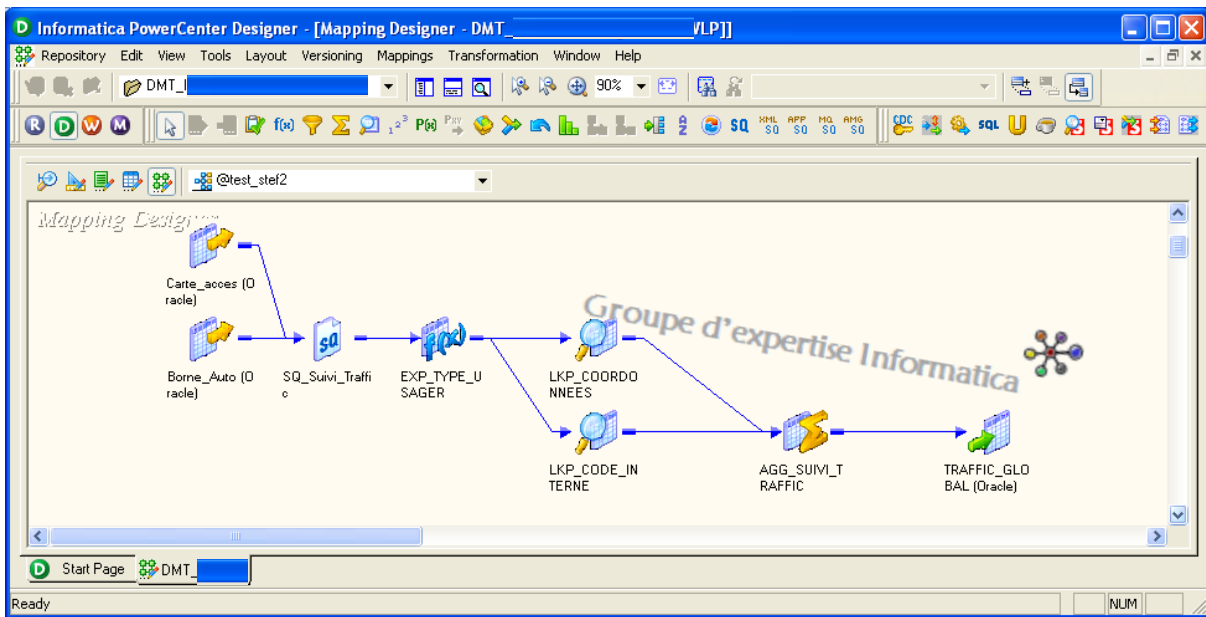




## 2. DE THREADS, PARTITIONS, ET POINTS DE PARTITIONS

### LES THREADS

- Les **Threads** servent à faire "avancer" les données du Source Qualifier vers les cibles (*Target*) existantes.
- Les données "avancent" par étapes (« *stages* »), définis par des points de partition. Les étapes peuvent être exécutées en parallèle (augmentation du nombre de partitions).
- Par défaut PowerCenter assigne un point de partition (représenté par un drapeau de couleur orange) aux objets Source Qualifier, aux transformations de type agrégation « non trié », et à la *Target*.



On retrouve ces différents types de *threads* à l'intérieur de la log de session, utile dans la recherche de goulots d'étranglement (prochaine fiche).

Severity	Timestamp	Node	Thread	Message Co...	Message
INFO	02/05/2012 11:37:17	node01_hpdv	READER_1_1_1	BLKR_16003	Initialization completed successfully.
INFO	02/05/2012 11:37:17	node01_hpdv	WRITER_1_*_1	WRT_8147	Writer: Target is database [redacted], bulk mode [OFF]
INFO	02/05/2012 11:37:17	node01_hpdv	WRITER_1_*_1	WRT_8124	Target Table FCT_Dx [redacted]:SQL INSERT statement: INSERT INTO FCT_DAT [redacted] FCT_Dx
INFO	02/05/2012 11:37:17	node01_hpdv	WRITER_1_*_1	WRT_8124	Target Tab UPDATE UPDA IN2_C WHE IN2_C = ? Al IN2_C

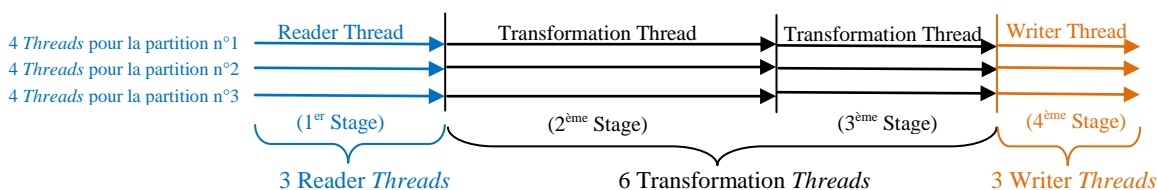
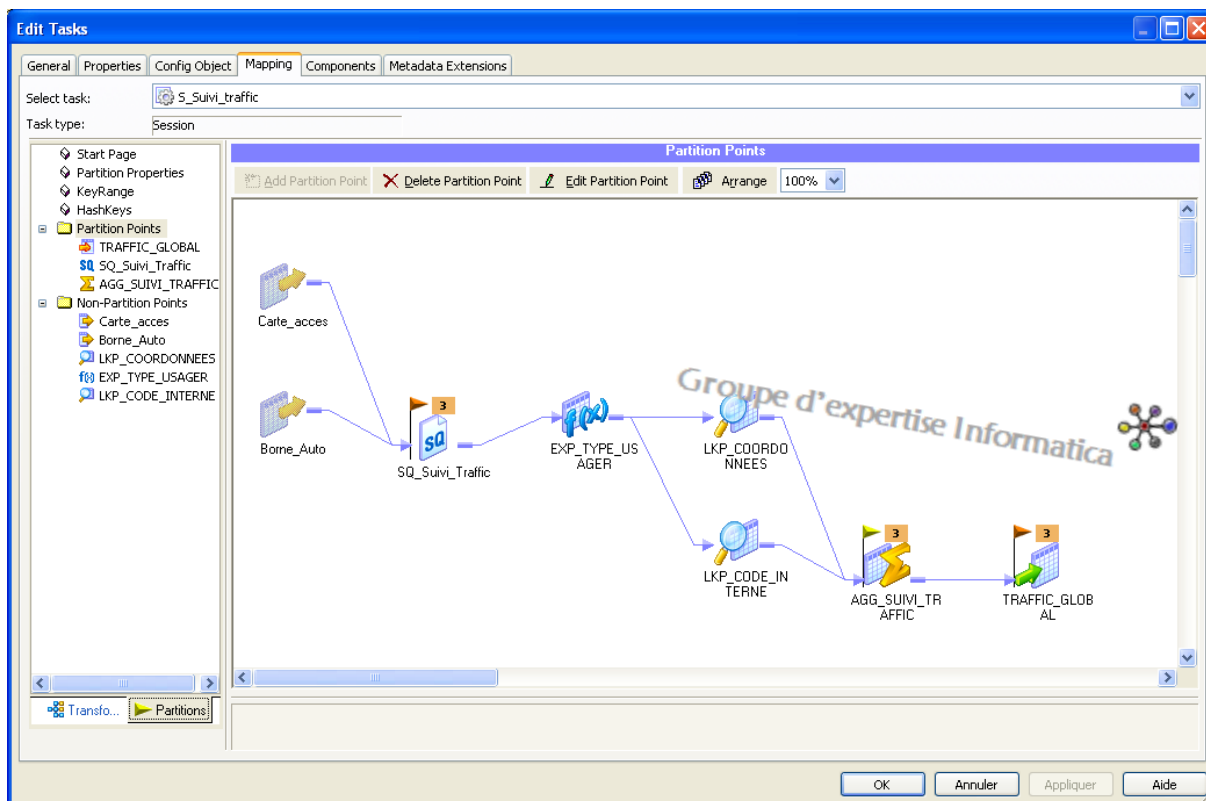
Severity: INFO  
Timestamp: 02/05/2012 11:37:16  
Node: node01\_ [redacted]  
Thread: DIRECTOR  
Process ID: 29897  
Message Code: VAR\_27028  
Message: Use override value [DMT\_ [redacted]]





## LES PARTITIONS

Si on reprend le schéma précédent, et qu'on le fait évoluer pour obtenir le même mapping mais avec 3 partitions en tout, on obtient :



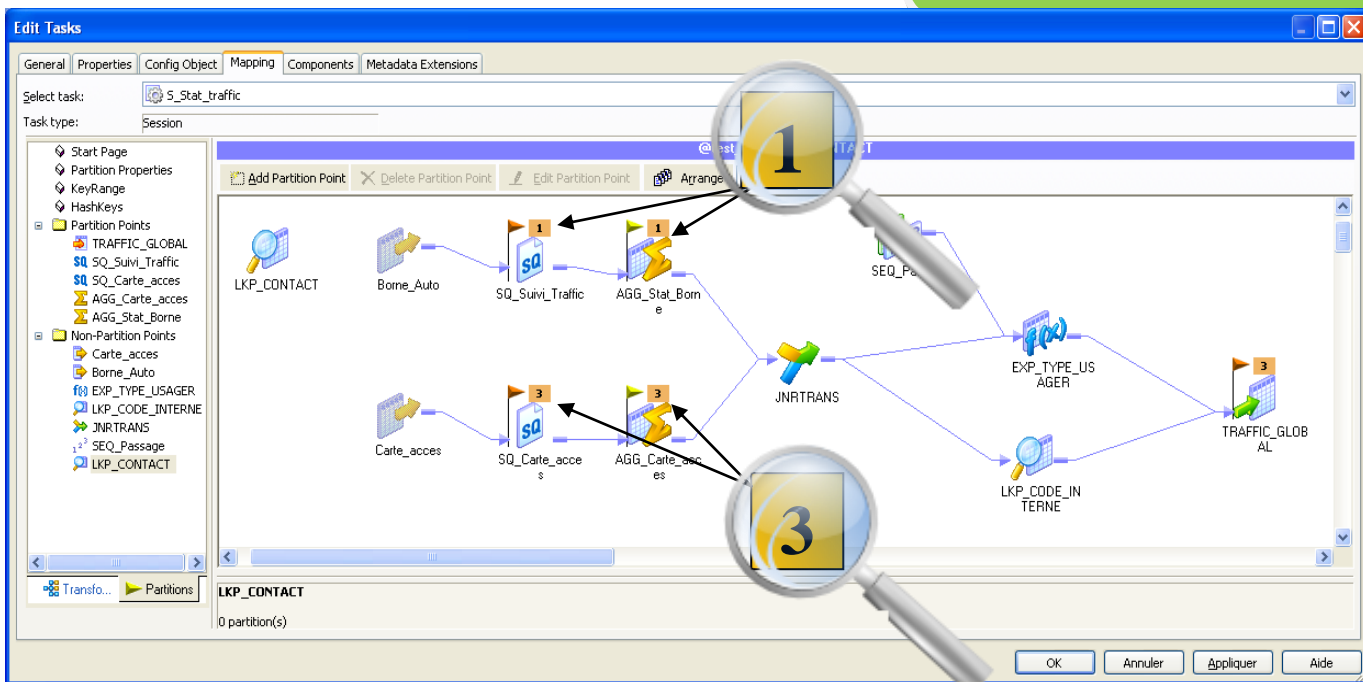
### Impact

- Ajouter des partitions augmente le nombre de *threads*
- Ajouter des points de partitions augmente le nombre de *stages* et donc de *threads*

### Remarques

- Chaque *pipeline* dispose d'au moins une partition
- Un *pipeline* avec une *target* XML ne peut avoir qu'une seule partition
- Si le *pipeline* a une source et/ou *target* de type relationnel, et que l'on a défini des n partitions, il y aura en base source n connexions parallèles et autant en base cible
- N'ajoutez des partitions et des points de partition que si vous avez suffisamment de bande passante
- A chaque point de partition, vous pouvez spécifier la façon dont seront réparties les données entre les partitions. Pour cela, on utilisera les types de partitions.
- Le numéro à côté de chaque drapeau indique le nombre de partitions
- Multi-projets: Calendrier des chargements pour éviter les goulots d'étranglement (plusieurs projets qui lanceraient des traitements gourmands dans la même fenêtre horaire) + supervision de la charge serveur.





Le nombre de partitions de chaque *pipeline* est identique.

#### Les recommandations :

- ✓ S'assurer d'avoir suffisamment de bande passante et de mémoire
- ✓ Ajouter les partitions une par une et surveiller le comportement CPU : quand ce dernier approche les 100%, n'ajoutez plus de partitions
- ✓ Penser à optimiser la taille de votre *buffer* en fonction du nombre de partitions
- ✓ Vous devez multiplier la taille des *caches* de transformation pour les objets de type Aggregators, Ranks, Joiners, & Sorters par le nombre de partitions

### LES POINTS DE PARTITIONS







- ✓ Un point de partition est associé à une transformation dans le mapping et découpe un pipeline en sections.
  - Ainsi, lorsqu'on perçoit un goulot d'étranglement dans une section du pipeline, on peut doubler le nombre de threads en décomposant la section du pipeline en deux par l'ajout d'un point de partition.
  - D'autre part, ce point de partition peut changer de type (cf chapitre suivant) pour mieux redistribuer les données parmi les différents threads de la transformation.
- ✓ Il n'est pas possible d'ajouter de points de partition sur :
  - Une transformation de type Sequence Generator
  - Une transformation déconnectée (cf. nos fiches précédentes)
  - Une transformation Source Definition
  - Un pipeline qui a été splitté puis re-concaténé, il n'est pas possible d'ajouter de point de partition entre le split et la concaténation. En effet, on aurait une section du pipeline qui irait plus vite que l'autre d'où un problème de synchronisation.
- ✓ Il n'est pas possible de supprimer le point de partition sur :
  - Une transformation de type Source Qualifier
  - Une transformation de type Normalizer pour fichier COBOL
  - Une transformation de type Target
- Les recommandations :
  - ✓ S'assurer d'avoir suffisamment de CPU *bandwidth*
  - ✓ Utiliser l'option de partition sur les transformations complexes pour lesquelles l'ajout de *threads* s'avère pertinent
  - ✓ Si vous avez plus d'une partition, ajouter les points de partitions là où les données ont besoin d'être redistribuées : Aggregator, Rank, or Sorter, où les données ont besoin d'être regroupées,



## C. LES TYPES DE PARTITIONS

### 1. DEFINITION

Si vous avez plus d'une partition, chaque point de partition spécifie la façon dont la donnée sera répartie entre les partitions. Les différents types de partitions peuvent visuellement être dissociées car elles sont représentées par des flags de couleurs différentes

-  • Pass through (orange)
-  • Key range (cyan)
-  • Round robin (vert)
-  • Hash auto keys (jaune)
-  • Hash user keys (bleu)
-  • Database (pourpre)

#### ***PARTITION TYPES – PASS THROUGH***

- ✓ Les données sont traitées sans qu'il y ait de répartition entre les partitions
- ✓ Utile lorsque vous souhaitez ajouter un thread supplémentaire pour une transformation complexe, mais que vous n'avez pas besoin de redistribuer les données (ou que vous avez une seule partition)
- ✓ Disponible pour n'importe quel point de partition

#### ***PARTITION TYPES – KEY RANGE***

- ✓ Le Service d'Intégration transmet les données à chaque partition en fonction des limites spécifiées (plage de données)
- ✓ PowerCenter rejette les lignes qui n'entrent pas dans les plages spécifiées
- ✓ Vous pouvez utiliser plusieurs ports pour former une clé de partition composée
- ✓ Attention à ne pas faire chevaucher les plages de données lorsque vous les définissez. Vous risquez d'avoir alors des duplications de données dans les partitions concernées
- ✓ Ce type de partition n'est pas utilisable pour les sources de données de type fichier plat, xml ou encore pour les objets de type Aggregator, Joiner, ou Rank

#### ***PARTITION TYPES – ROUND ROBIN***

- ✓ Le Service d'intégration distribue des lignes de données uniformément à toutes les partitions
- ✓ Type de partition à utiliser quand:
  - il n'est pas nécessaire de regrouper les données entre les partitions
  - on a des sources de données issues de fichiers plats de tailles différentes
  - quand les données ont été partitionnées de façon inégale en amont et nécessitent d'être modifiées avant d'arriver à la cible
- ✓ Ce type de partition n'est pas utilisable pour les objets de type Sources Qualifier, Aggregator, Joiner, Rank, et Sorter

#### ***PARTITION TYPES – HASH AUTO KEYS***

- ✓ La fonction d'intégration applique une fonction de hachage à une clé de partition pour regrouper des données entre les partitions
- ✓ Utiliser ce type de partitionnement pour que des groupes de lignes soient traités dans la même partition
- ✓ Le Service d'Intégration utilisera alors les ports de group by et de tri en fonction de la clé de partition qu'il aura définie.
- ✓ Utilisable sur les objets de type Aggregator (et option de tri non cochée), Joiner, Lookup, Rank, et Sorter



### ***PARTITION TYPES – HASH USER KEYS***

- ✓ La fonction d'intégration applique une fonction de hachage à une clé de partition pour regrouper des données entre les partitions
- ✓ Utiliser ce type de partitionnement pour que des groupes de lignes soient traités dans la même partition
- ✓ Contrairement au type de partition hash auto keys, c'est à vous de spécifier les ports qui formeront la clé de partition.
- ✓ Non utilisable pour les objets de type Source Qualifier, Aggregator, Joiner, Rank, et Sorter

### ***PARTITION TYPES – DATABASE***

- ✓ Ce type de partition n'est utilisable que pour les bases de données DB2 et Oracle dans leur mode multi-nœuds permettant la parallélisations des exécutions de requêtes.
  - Tables sources: Oracle and DB2
  - Tables cibles: Oracle and DB2
- ✓ Requis lorsque l'on utilise plus d'une partition ou dans l'utilisation du DB2 bulk loading (que je n'ai jamais testé pour ma part).
- ✓ Le nombre de partitions n'a pas à être égal au nombre de nœuds de la base de données. Mais la rentabilité reste la meilleure quand le nombre de nœuds de la base est égale au nombre de partitions.

### ***FOCUS SUR LES SOURCES ET CIBLES DES SESSIONS***

#### *Partition Types – Sur les données sources*

- ✓ **Sources relationnelles (base de données)**
  - PowerCenter crée des requêtes sql différentes pour chacune des partitions. La base de données ouvrira donc autant de sessions de connexion que de partitions Informatica
  - La base de données doit pouvoir supporter les connexions parallèles
  - Il est possible de faire de l'overwrite pour chacune des requêtes sql.
- ✓ **Fichiers plats**
  - Dans le cas de multiples fichiers plats en entrée
    - Chaque partition lit un fichier différent
    - PowerCenter lira les fichiers en parallèle
    - Si les fichiers sont de tailles significativement différentes, il est conseillé de répartir les données selon le type de partition « Round-Robin ».
  - Dans le cas où l'on a un seul fichier plat en entrée
    - PowerCenter effectuera des connexions multiples en parallèle. Le nombre de connexions sera basé sur le nombre de partitions spécifiées.
    - PowerCenter distribuera les données de façon aléatoire dans les partitions
    - Cette option est disponible uniquement à partir de PowerCenter version 7.11

#### *Partition Types – Sur les données cibles*

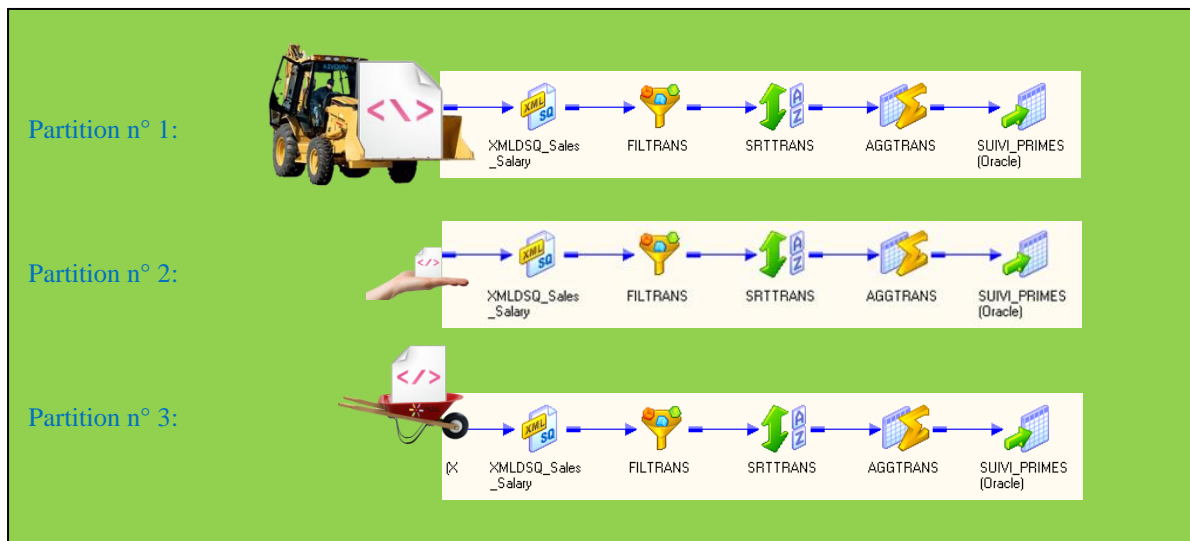
- ✓ **Relational Targets**
  - PowerCenter créera autant de connexions à la base de données cible, qu'il existe de partitions.
  - PowerCenter charge les données simultanément
- ✓ **File Targets**
  - PowerCenter écrit les données de chaque partition dans des fichiers séparés.
  - PowerCenter peut merger les fichiers cibles si tous on une connexion locale à la machine de l'Intégration Service
  - PowerCenter charge les données simultanément



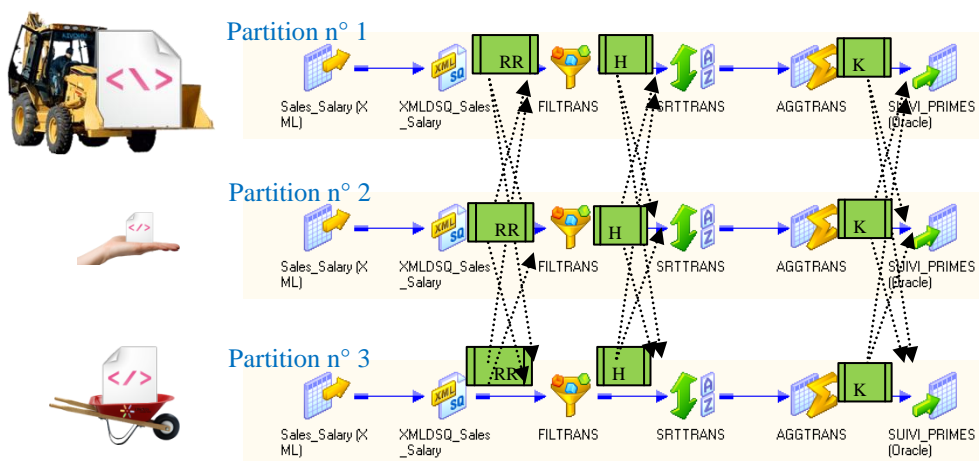
## 2. EXEMPLE D'UTILISATION DES TYPES DE PARTITION

On prendra comme scenario :

- ✓ Le traitement de données d'étudiants
- ✓ Source XML et cible Oracle
- ✓ Source XML éclatée en 3 fichiers de tailles différentes.
- ✓ On définit une partition pour chaque fichier source (donc 3 partitions en tout).



Type de partition proposée:



### Légende

**RR:** Utiliser le type de partitionnement Round Robin pour équilibrer la charge

**H:** Utiliser le type de partitionnement Hash auto keys sur l'objet Rank pour grouper les lignes de façon appropriée.

**K:** Utiliser le type de partitionnement Key Range pour optimiser l'écriture des données dans la table cible.



## D. RETOUR D'EXPERIENCE

### 1. DUREE DE TRAITEMENT

A titre informatif, vous trouverez ci joint un exemple de temps de chargement d'un traitement avant et après optimisation:

#### État 1: début 2008 -----

Lorsque je suis arrivé chez mon client actuel en janvier 2008, l'équipe projet avait dans ses différents projets Informatica, un projet qui avait une problématique de temps de chargement: Dans un projet particulier, le projet contenait une 50aine de mappings. L'un d'entre eux prenait à lui seul environ **7 heures de temps** pour une volumétrie **d'environ 1,5 Millions de lignes**.

Ce même traitement, en fin d'année 2008, prenait environ **11 heures de temps** pour une volumétrie **d'environ 2 Millions de lignes**.

Ces temps étaient pénalisant car:

- ✓ Le chargement se terminait alors vers 8heures du matin,
- ✓ Lorsque l'on voulait tester un chargement global dans l'environnement de DEV ou de REC, il fallait souvent attendre le lendemain pour que la MOA puisse avoir ses données

Tables agrégées en 2008: **Environ 11h pour environ 2 Millions de lignes**

#### État 2: début 2009-----

Nous avons effectué un chantier d'optimisation pour améliorer les temps de traitement.

Parmi les actions entreprises, nous avons notamment :

- ✓ Partitionné les tables sources,
- ✓ Partitionné et sous partitionné les tables cibles,
- ✓ Supprimé les index avant le chargement des tables agrégées,
- ✓ Recréé les index après chargement.

**Pour environ 2 Millions de lignes**

- **Environ 55 minutes pour charger les tables**
- **Environ 25 minutes pour recréer les index**

Tables agrégées en 2009: **Environ 1h20 pour environ 4,2 Millions de lignes**

#### État 3: Milieu 2012-----

Depuis 2009 à 2012,

- ✓ Il y a eu environ 6 millions de nouvelle lignes par an,
- ✓ De plus, il a été fait une reprise d'historique d'une nouvelle application, dont les données sont venues compléter ces même tables agrégées pour une volumétrie supplémentaire d'environ **63 millions de lignes**.

**Pour environ 87 628 429 de lignes**

- **Environ 1h05 minutes pour charger les tables (datas)**
- **Environ 3h13 minutes pour recréer les index**

Tables agrégées à mi 2012: **Environ 4h18 pour environ 87 628 429 Millions de lignes**



## 2. BUG RENCONTRE

Lors de la mise en place:

- ✓ des partitions sous Informatica,
- ✓ en mode de chargement BULK sous Informatica,
- ✓ combiné aux sous partitions et sous-partition d'Oracle
- ✓ en mode **Enable parallel Mode activé**

Sur un environnement de développement, Funji MATEMU (l'un de nos experts, relecteur de cette fiche) avait observé une augmentation extrêmement importante du tablespace Oracle. Il a pu reproduire le problème, ci dessous mis en avant:

### CONDITION DE LANCEMENT

- 1 – Pas d'index
- 2 – Pas de contrainte
- 3 – allocation mémoire pour la table cible à vide d'environ 3 Go
- 4 – Taille du fichier source d'environ 270 Mo
- 5 – Nombre d'enregistrements : 3 426 721
- 6 – Très peu de transformation de données sous Informatica

### TESTS

#### TEST 1

- ✓ Mode normal
- ✓ Enable parallel Mode activé
- ✓ Commit interval de 10 000

aucune augmentation sur la table cible

#### TEST 2

- ✓ Mode bulk
- ✓ Enable parallel Mode activé
- ✓ Commit interval de 10 000

augmentation d'environ **10 Go** sur la table cible

#### TEST 3

- ✓ Mode bulk
- ✓ Enable parallel Mode désactivé
- ✓ Commit interval de 10 000

aucune augmentation sur la table cible

#### TEST 4

- ✓ Mode bulk
- ✓ Enable parallel Mode activé
- ✓ Commit interval de 3,5 millions

augmentation d'environ **1,5 Go** sur la table cible

### OBSERVATION

D'après le dba expert Oracle,

- ✓ en moyenne sous la table cible chaque enregistrement pèse 80 octets.
- ✓ Soit une taille totale estimée après chargement de 80 octets \* 3 426 721 enregistrements = 274 137 680 octets, soit environ 275 Mo.

Comme la table à vide disposait de 3Go de disque alloué, elle n'aurait jamais dû grossir dans les Tests 2 et 4 ci dessus.





Il n'observe pas non plus la parallélisation des tâches sous Oracle.

Notez que le Test 4 permet de ne réaliser qu'un commit sur la base et qu'il a diminué l'augmentation de l'allocation du disque (division par 6) par rapport au Test 2.

D'autre part, les deux modes donnent la même performance en termes de temps d'exécution.

### 3. ERREUR RECURRENTE

---

Il est important de comprendre que partitionner est une des solutions possibles lorsque l'on veut optimiser. Cependant, cela s'adresse à des développeurs confirmés et ne doit être utilisé qu'à la fin du processus d'optimisation des traitements Informatica.

Globalement, l'optimisation des traitements Informatica doit passer:

- 1 - Par le respect des règles de nommage,
- 2 - L'utilisation des bests practices de développement (Cf. fiche Optimisation 4.1 - les bests practices de développement),
- 3 - L'analyse des sessions d'exécution pour éradiquer les goulots d'étranglement,
- 4 - Et enfin utiliser les partitions quand cela est nécessaire.

---

## CONCLUSION

---

- ✓ On peut avoir mis en place une super architecture, des serveurs de folies, suivre les normes de développements et avoir optimisé la base de données, et être pénalisé par le débit du réseau,
- ✓ On peut avoir une bonne architecture, des serveurs bien taillés, des bases optimisées, mais ne pas avoir planifié le démarrage des nombreux projets qui démarrent tous en même temps,
- ✓ On peut avoir une bonne architecture, les bonnes ressources, tout bien planifier, et avoir des requêtes mal écrites dans les SQ Override,
- ✓ etc.

Ce qui est passionnant dans ce genre d'audit, c'est qu'on doit s'intéresser à une pléthore de paramètres illustrés par la diversité de la population concernée. On peut aller discuter avec un développeur sql ou avec un dba, en passant par l'architecte, le responsable réseau ou encore le DSI pour des problématiques.

Les problématiques varient d'un client à l'autre, le plus important, au début d'un audit, est de ne pas se disperser. Il faut avant tout cartographier l'environnement et saucissonner l'architecture, puis identifier les goulots d'étranglements pour affiner au fur et à mesure.

J'espère que cette fiche vous servira et qu'il sera un bon support pour démarrer vos premiers audits !





## DOCUMENTS DE REFERENCE

---

### **Fiche de la communauté**

- ✓ [Best practice - Fiche 1 - Organisation et droits](#)
- ✓ [Best practice - Fiche 2 - Normalisation et codification](#)
- ✓ [Best Practice - Fiche 3 - Méthodologie VELOCITY](#)
- ✓ [Best Practice - Fiche 4 - Audit ETL - Optimisation Informatica](#)

### **PowerCenter Designer Guide 8.5.1**

- ✓ [Workflow administration guide](#)