INFORMATICA POWERCENTER

Mise en place de la récursivité avec l'option *Concurrent Execution*

Dans cette fiche astuce, nous allons mettre en avant une évolution d'Informatica PowerCenter v8.5, qui permet la mise en place de la notion de récursivité dans les traitements.

[Wikipedia] : En informatique et en logique, une fonction ou plus généralement un algorithme qui contient un appel à elle-même est dite **récursive**. Deux fonctions peuvent s'appeler l'une l'autre, on parle alors de **récursivité croisée**.

Nous allons vous montrer à travers cette fiche en quoi cette option vous permettra d'industrialiser vos traitements récurrents, avec un ROI quasi logique :

- gain de temps (économie d'échelle avec une rationalisation du nombre de traitements)
- meilleure maintenance
- meilleure visibilité, et meilleur compréhension fonctionnelle des traitements
- documentation facilitée

Si vous avez des questions, ou des remarques, n'hésitez pas à nous les faire parvenir. Pour accéder aux fiches : http://www.unovia.fr/Informatica/fiches. Un espace d'échange est en cours de construction, mais vous pouvez d'ores et déjà nous joindre à l'adresse suivante : expert_informatica@unovia.fr

Tout le travail accompli par ce groupe est fait de façon bénévole, n'hésitez donc pas à les remercier et à les encourager.

Fabien Duprey

Mon Profil Viadeo

Re-lecteurs: Christophe FOURNEL, Yannick TEMPORAL, Funji MATEMU, Stéphane THIA

SOMMAIRE

PROBLEMATIQUE : LES TRAITEMENTS RECURRENTS

A. CONTEXTE

Jusqu'à la version 8.5 de Informatica Power Center, il n'était pas possible de lancer de manière récursive un workflow ou une worklet sans l'aide extérieure d'un ordonnanceur (ex : Control-M) ou d'un script système (ex : bash/ksh).

Pour bien comprendre cette idée d'utilisation de la nouvelle fonctionnalité (*concurrent execution*), on peut prendre le contexte des reprises des données, qui peuvent être quotidiennes, hebdomadaires, mensuelles.

Habituellement, pour adresser ces problématiques, il existe un certain nombre de solutions techniques, dont l'efficacité et le coût dépendent de la complexité des données à reprendre et des règles fonctionnelles appliquées par l'ETL. Par exemple, on peut :

- Préparer une requête SQL qui simule le traitement sur une plage de données antérieure,
- Monter un traitement spécifique pour la reprise avec l'ETL,
- Piloter le traitement « normal » de l'ETL d'une manière différente pour la première exécution, afin qu'il traite toutes les données depuis un temps donné au lieu de ne traiter que le jour, la semaine, ou le mois en cours,
- Ou encore faire répéter le traitement « normal » en visant les données du début de la période de reprise, un nombre de fois fini correspondant au nombre d'itérations entre le début et la fin de la période.

Mais que se passe-t-il lorsque les données d'une itération sont dépendantes des résultats de l'itération précédente ?

B. EXEMPLE

Une problématique nécessitant la récursivité

Prenons un exemple : Dans un contexte assurance-vie, une application de gestion source me donne chaque mois pour l'ensemble des contrats financiers sélectionnés, une image du stock mensuel, et des mouvements quotidiens. Ces données sont transformées et insérées dans un Datawarehouse cible.

Fonctionnellement, dans le Datawarehouse cible, on doit calculer pour chaque jour ouvré l'état du stock, modifié par les mouvements, mais également par certaines règles de calcul et par des actions des gens du métier, via une application VBA pointant sur le Datawarehouse.

Autrement dit, pour chaque chargement mensuel dans le Datawarehouse, je dois travailler avec les données de mon application de gestion source du mois en cours, mais également avec les données modifiées du mois précédent sur le Datawarehouse.

Imaginons que :

- Les données opérationnelles existent à partir de janvier 2010,
- On soit au mois de mars 2011,
- Et que la livraison soit prévue pour le mois d'avril 2011.

Le jour de la mise en production, nous devrons être capable de faire une reprise de l'historique, depuis janvier 2010 jusqu'au mois d'avril 2011, mais également, de faire une alimentation mensuelle.

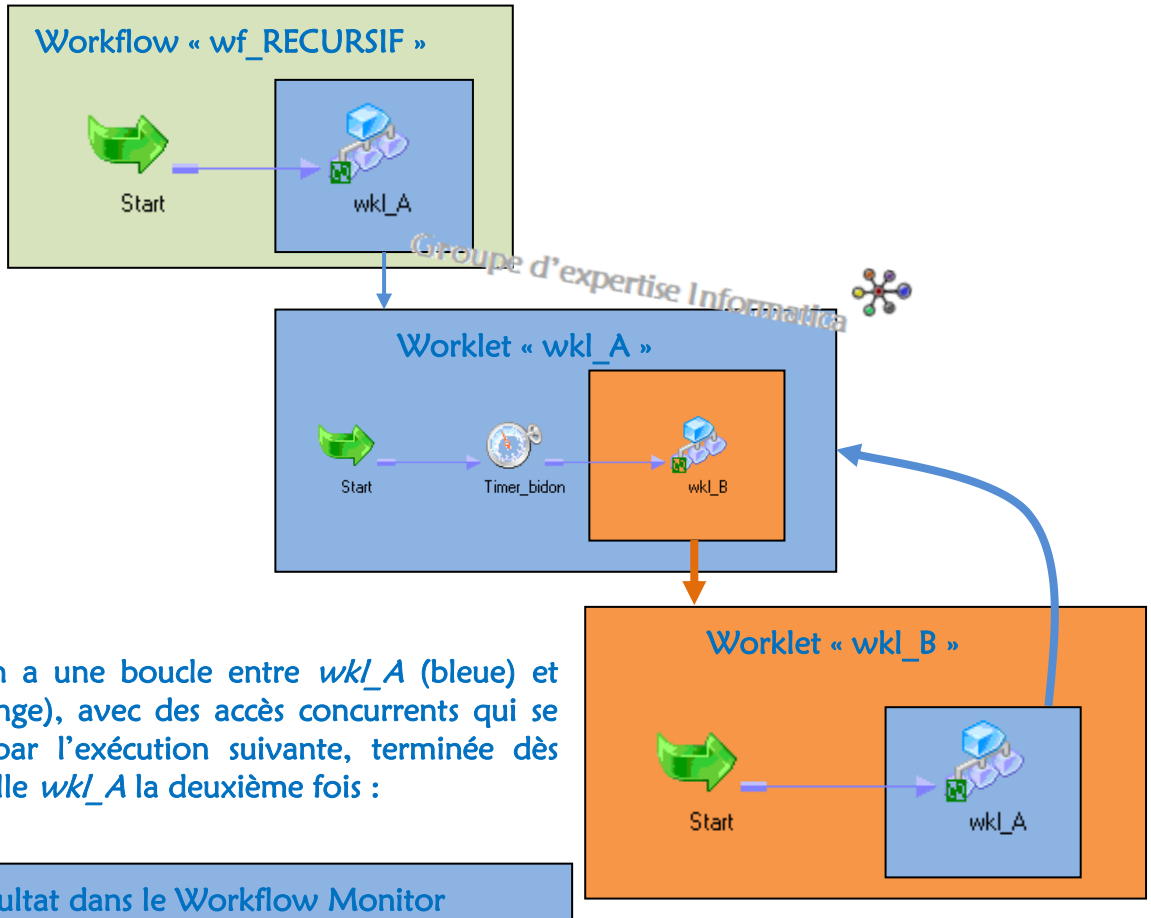
Toutes les solutions habituelles de reprise deviennent coûteuses et/ou difficiles à mettre en place.

FIN D'UN BLOCAGE TECHNIQUE : LA RECURSIVITE

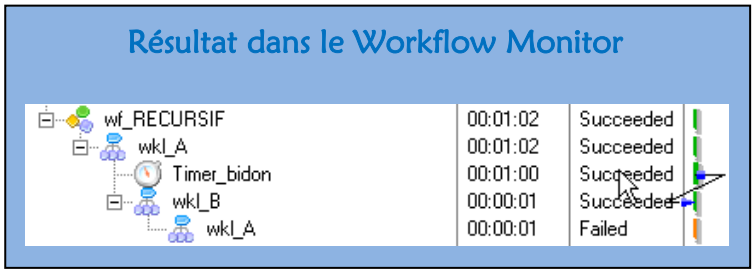
Jusqu'ici, si nous avions voulu répondre à la problématique prise en exemple via la récursivité, voici à quel blocage nous nous serions confronté avec Informatica :

Nous avons un workflow *wf_RECURSIF* qui fait appel à la worklet *wkl_A*, la worklet *wkl_A* fait elle-même appel à la worklet *wkl_B*, et enfin la worklet *wkl_B* ferme la boucle en appelant *wkl_A*.

Si nous lançons le workflow, Informatica arrête le traitement, dès que deux instances d'une même worklet sont en cours d'exécution, ce que le produit ne gère pas.. Ci-dessous l'illustration :



Résultat, on a une boucle entre *wkl_A* (bleue) et *wkl_B* (orange), avec des accès concurrents qui se traduisent par l'exécution suivante, terminée dès qu'on appelle *wkl_A* la deuxième fois :



Mais à partir de la version 8.5, une nouvelle option est apparue dans l'onglet « General » des propriétés des worklets et workflows :

C'est en activant cette nouvelle option, que l'IS va pouvoir gérer l'exécution de sessions concurrentes d'un même workflow ou worklet. A travers cet exemple pratique, je vais vous présenter l'un des avantages de celle-ci.

EXEMPLE DE MISE EN ŒUVRE

Pour expliquer l'exemple technique suivant, nous allons reprendre l'exemple fonctionnel décrit en début de document :

Chaque mois, pour alimenter notre table des stocks, nous avons besoin :

1) De prendre en compte les données provenant de l'application source
2) De prendre en compte et modifier les données du mois précédent sur le Datawarehouse cible

Pour cela, nous avons deux traitements :

- Un traitement que l'on nomme **Pilotage**, qui nous sert à calibrer les données nécessaires au traitement mensuel,
- Un traitement que l'on nomme **Traitement mensuel**, qui nous sert à effectuer les traitements de données applicatives.

Dans le cadre d'une mise en production future, il faut donc prévoir d'exécuter ces traitements : soit en mode Normal, soit en mode Reprise de données.

A. MISE EN ŒUVRE SANS RECURSIVITE

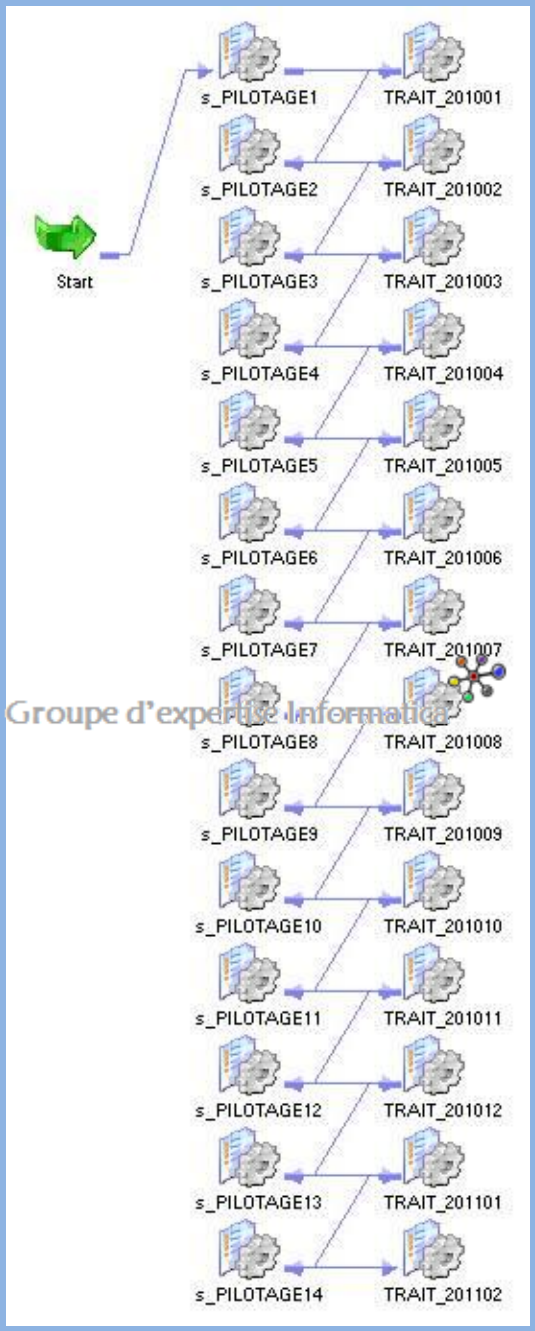
Pour répondre à ce type de problématique, sans la fonctionnalité de récursivité, traditionnellement, on peut :

- 1 - Préparer une requête SQL qui simule le traitement et fait la reprise,
- 2 - Monter un traitement spécifique pour la reprise avec l'ETL,
- 3 - Piloter le traitement « normal » de l'ETL d'une manière différente pour la première exécution, afin qu'il traite toutes les données depuis un temps donné au lieu de ne traiter que le jour, la semaine, ou le mois en cours,
- 4 - Ou encore faire répéter le traitement « normal » récurrent en visant les données du début de la période de reprise, un nombre de fois fini correspondant au nombre d'itérations entre le début et la fin de la période.

Dans notre contexte, la solution la moins coûteuse est sûrement la 4ème : répéter le traitement X fois en visant les plages de données correctes par table de pilotage, du début à la fin de la période de reprise.

Illustration du traitement mode Normal

Illustration du traitement en mode Reprise



Mode normal : Ce mode correspond à notre traitement mensuel. Ce dernier consisterait à alimenter la table dit de *Pilotage*, puis à lancer le *Traitement mensuel* des données.

Mode reprise de données : Ce mode consiste à ajuster la table de *Pilotage* pour le premier mois, à lancer le *Traitement mensuel*, puis ajuster la table de *Pilotage* pour le deuxième mois, à relancer le *Traitement mensuel*, et ainsi de suite autant de fois qu'il y a de mois entre le début de la reprise et la date prévue de mise en production.

Remarque : Pour optimiser cet enchainement de traitements (Illustration en mode Reprise), la session s_Pilotage et s_Traitement n'ont été développées qu'une seule fois : elles ont été configurées en reusable (en cas de modification d'un des deux mappings, on ne le fera donc qu'une seule fois).

Malgré l'utilisation du reusable, et du gain de temps que cela peut procurer, il reste que :

- Si une option propre à chaque mois chargé doit être changée, elle doit être changée sur l'ensemble de la chaîne, session par session (-> Maintenance risquée).
- On s'impose un nombre de mois à charger : Si le déploiement en production prend du retard (ou de l'avance, mais c'est moins courant…), ou doit être relancé à cause d'une correction des données sources, nous sommes obligés de rajouter un (ou des) maillon(s) à la chaîne et relivrer/revalider chaque étape dans chaque environnement ! (->Evolutions coûteuses)

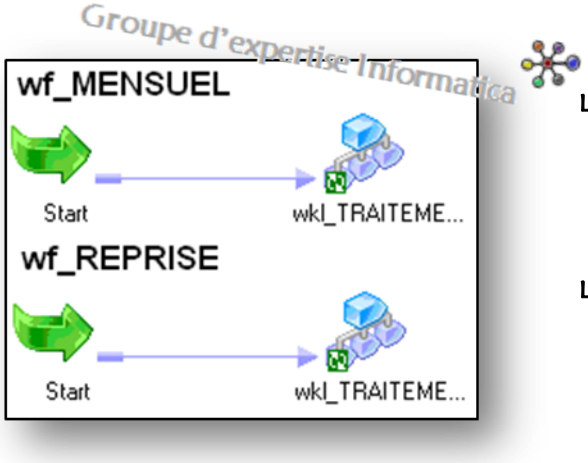
Ces problèmes n'existent plus si on utilise la nouvelle option, qui nous permet d'implémenter la récursivité, comme nous allons vous l'illustrer :

B. Mise en œuvre avec recursivite

Le but de l'exemple suivant est de créer un système rationnalisé, qui soit indépendant de la date de déploiement, et qui évite la répétition du code du pilotage ainsi que du traitement des données.

Cette fois encore, nous avons deux modes d'alimentation (Mode Normal, et mode Reprise), qui se traduisent par la mise en place de deux workflows : Le workflow mensuel, (Wf_MENSUEL) et le workflow de reprise (Wf_REPRISE).

Bien qu'ils se ressemblent, ce qui les différencie, c'est l'utilisation d'une variable de worklet/workflow, que l'on nommera dans notre exemple **$$CHARGEMENTS** (de type integer) et dont la valeur sera initialisée différemment selon qu'on soit en mode Normal ou en mode REPRISE. C'est cette variable qui va nous permettre de contrôler la récursivité.

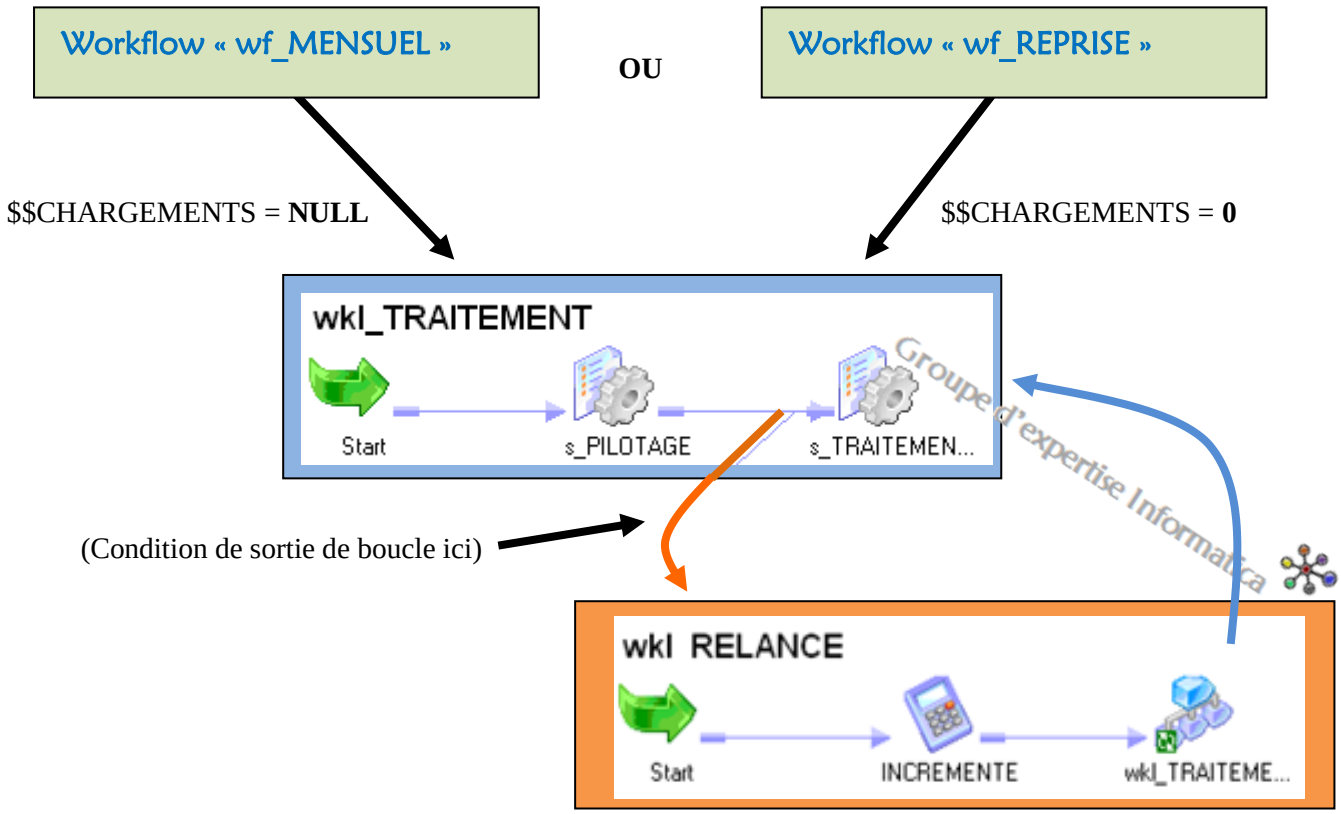


Le workflow wf_MENSUEL (Alimentation en **mode Normal**)

- il initialise la variable $$CHARGEMENTS à NULL,
- puis lance la worklet wkl_TRAITEMENT en lui passant la variable.

Le workflow wf_REPRISE (Alimentation en **mode REPRISE**)

- il initialise la variable $$CHARGEMENTS à 0,
- puis lance la worklet wkl_TRAITEMENT en lui passant la variable.



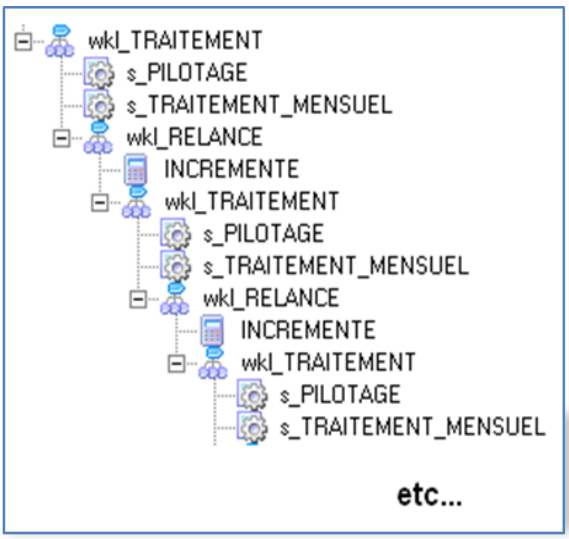
Cette fois, le système de récursivité peut fonctionner, car les worklets ont l'option « Configure Concurrent Execution » cochée !

La worklet wkl_TRAITEMENT

- reçoit la variable $$CHARGEMENTS,
- lance les sessions de pilotage et traitement des données,
- si la variable $$CHARGEMENTS n'est pas NULL, ou n'est pas égale au nombre de mois entre le premier mois de reprise et la date d'aujourd'hui, lance la worklet RELANCE. (ainsi, nous avons une condition de sortie de boucle)

La worklet wkl_RELANCE

- reçoit la variable $$CHARGEMENTS,
- incrémente $$CHARGEMENTS de 1,
- et relance la worklet TRAITEMENT.



Les avantages d'une telle solution sont nombreux :

- Le mapping de pilotage et de traitement n'existent que dans une seule session dans une seule worklet,
- Pas de copier-coller des sessions, une configuration unique,
- Le nombre de lancements du traitement mensuel lors de la reprise ne dépend plus que de la date à laquelle on la lance ! Pas besoin de re-livrer si le déploiement est en retard ou si on veut refaire une reprise,
- Libre à vous de rajouter des chemins, par exemple vous pouvez vouloir lancer un mapping spécial à la fin de chaque trimestre ? utilisez la variable, avec par exemple une condition sur une division entière par trois ?
- L'immense plaisir de casser la mise en page du Workflow Monitor chez tous vos voisins branchés sur le même environnement…

Testez vous-même !

CONCLUSION

Nous pouvons implémenter une forme de récursivité sous Informatica :

- ✓ sans dépendre d'un ordonnanceur externe ou de scripts système,
- ✓ sans devoir fixer un nombre de boucles,
- ✓ sans copier X fois le code et risquer d'autant plus les erreurs de configuration des sessions.

L'exploration des avantages de l'option « Configure Concurrent Execution », selon les situations techniques et/ou fonctionnelles, peut s'avérer très gagnante en matière de coûts de développement, de livraison et de maintenance des traitements.

Elle a également un potentiel certain dans la résolution de problématiques complexes spécifiques, limitant par exemple d'éventuelles acrobaties dans les mappings, ou l'introduction de procédures stockées et autres overrides de sources/cibles.

DOCUMENT DE REFERENCE

PowerCenter Workflow Administration Guide 8.5

- Chapter 20: Running concurrent Workflows

GLOSSAIRE

- **Récursivité** [Wikipédia] : En informatique et en logique, une fonction ou plus généralement un algorithme qui contient un appel à elle-même est dite récursive. Deux fonctions peuvent s'appeler l'une l'autre, on parle alors de récursivité croisée.

CONTACTS ET FICHES

Je fais partie d'une communauté regroupant aujourd'hui une soixantaine d'indépendants spécialisés dans le décisionnel.

L'objectif de cette communauté est de capitaliser sur les compétences d'experts du métier et de les diffuser vers l'extérieur, afin d'y faire connaître notre expertise.

Cette reconnaissance, nous souhaitons l'obtenir :

- par des formations (nous sommes nombreux à donner des formations pour le compte de grands cabinets des formations),
- par la diffusion de fiches :
  - de type fiche produit,
  - de type best practices (une fiche n'est diffusée qu'après validation d'une groupe d'experts composé de 4 à 5 personnes),
  - de type astuce de développement,
- par du service (prestations d'expertise, de conseil, d'architecture, d'audit) soit directement pour les clients, soit pour le compte des éditeurs à travers leur pôle service.

Si vous avez des questions, ou des remarques, n'hésitez pas à nous les faire parvenir. Pour accéder à toutes nos fiches (best practices, astuces, …) vous pouvez vous rendre à l'url suivante : http://www.unovia.fr/Informatica/fiches.

Un espace d'échange est en cours de construction, mais vous pouvez d'ores et déjà nous joindre à l'adresse suivante : expert_informatica@unovia.fr