**[Oracle Scheduler: Introduction et planification d'un premier job](http://braimt.blogspot.com/2012/11/oracle-scheduler-introduction-et.html)**

**1. Introduction**

Avec la complexité des systèmes actuels, il est assez courant de devoir automatiser certaines tâches. Depuis la version 10g d'Oracle, un ordonnanceur (Oracle Scheduler) est apparu pour permettre de planifier, automatiser, administrer... de manière simple, efficace et centralisée tout type de traitement à automatiser. Nous allons commencer une série d'articles par rapport à cet ordonnanceur. Tout au long de cette série, nous présenterons certaines fonctionnalités de cet ordonnanceur. Précisons tout de même que les possibilités de l’ordonnanceur sont si étendues que nous ne pourront toutes les listées et les décrire complètement...  
  
Ce premier article introduira l'ordonnanceur et planifiera un premier job de manière simple et facile. Tout au long de ce message, nous utiliserons le terme "Oracle Scheduler" pour désigner cet ordonnanceur.

**2. Présentation**

Comme nous l’avons précisé dans notre introduction, le scheduler d’Oracle permet de gérer des traitements à automatiser. Cet ordonnanceur est contrôlé au travers du package DBMS\_SCHEDULER. Ce package reprend une série de fonctions et procédures permettant de gérer des besoins d’ordonnancement allant des plus simples aux plus complexes. Ainsi, il est possible de:

* Planifier une tâche pour que celle-ci démarre à un moment donné ou en fonction d’un évènement.
* Contrôler les ressources attribuées à une tâche, définir la priorité d’une tâche en fonction de sa durée d’exécution...
* Gérer et contrôler l'exécution des tâches et leur dépendances.
* ...

Ce scheduler offre une approche modulaire. L'avantage principal de cette approche est la possibilité de réutilisé certaines composantes. Parmi les principales composantes manipulé par le scheduler, nous pouvons citer:

* Le programme (program): est une collection de méta-données qui permet de définir le type d'action à exécuter (procédure PL/SQL, procédure Java, bloc anonyme), le nom de l’action à exécuter, le nombre d’arguments...
* Les échéanciers (schedule): permettent de définir quand et combien de fois un job est à exécuter.
* Le job (job) est un programme (program) planifié à certains moments (schedule).

Nous détaillerons dans les paragraphes suivants ces différents objets.

**3. Privilèges nécéssaires**

Pour pouvoir planifier des jobs grâce au scheduler d'Oracle, un utilisateur à besoin du privilège CREATE JOB. Ce privilège conviendra dans la grande majorité des cas.

Pour des utilisations plus évoluées du scheduler, le privilège MANAGE SCHEDULER permettra, à l’utilisateur qui en dispose, de pouvoir créer des classes de travail, de définir des priorités

**4. Planifier un traitement: la méthode simple et rapide**

Comme nous l’avons vu précédemment, le scheduler d’Oracle est contrôlé au travers du package DBMS\_SCHEDULER et offre une approche modulaire. Dans l’exemple suivant, nous planifierons la création d’un premier job.

Imaginons le scénario suivant. Un certain utilisateur doit envoyer un message à tout les membres d’un groupe tout les jours de la semaine à 6 heure du matin. Ce message ne doit pas être envoyé le week-end. Comment faire pour automatiser ce traitement?

Pour des raisons de facilités, nous supposerons que la procédure send\_message du package pkg\_actions contient la logique nécessaire pour envoyer le message au bon groupe.

BEGIN

DBMS\_SCHEDULER.create\_job (

job\_name => 'mon\_premier\_job',

job\_type => 'PLSQL\_BLOCK',

job\_action => 'BEGIN pkg\_actions.send\_message; END;',

start\_date => SYSTIMESTAMP,

repeat\_interval => 'freq=daily; byday=MON,TUE,WED,THU,FRI; byhour=06; byminute=0;',

end\_date => NULL,

enabled => TRUE,

comments => 'Mon job entièrement défini dans la procédure CREATE JOB.'

);

END;

/

Le code précédant permet de planifier le job précédemment décrit. Pour créer le job, nous faisons appel à la procédure create\_job du package DBMS\_SCHEDULER. Il existe plusieurs variantes de cette procédure. Nous allons passer en revue les différents paramètres utilisés dans l'appel précédant. Mais la description est un peu réductrice:

* job\_name: Chaque job doit obligatoirement avoir un nom. Ce nom doit être unique au niveau de [l'espace de noms](http://braimt.blogspot.be/2012/11/les-espaces-de-noms.html) contenant les tables, vues....
* job\_type: Chaque job doit obligatoirement avoir un type. Nous avons utilisé le type PLSQL\_BLOCK. Ce type permet de spécifier un bloc PL/SQL anonyme comme action. Nous reviendrons sur les autres types dans des articles futurs.
* job\_action: Ce paramètre défini l'action à réaliser par le job. Il est possible de faire appel à une procédure, définir une procédure PL/SQL anonyme directement dans ce paramètre, ...
* start\_date: est la date à laquelle le job doit être démarré.
* repeat\_interval: Est la fréquence de répétition. La syntaxe est plutôt facile à comprendre. Dans notre exemple, nous voulons que notre job soit démarré de manière quotidienne (daily), uniquement les jours de la semaine (MON-FRI) à 06:00h du matin.
* end\_date: est la data à partir de laquelle le job ne doit plus être exécutée.
* enabled: Par défaut, un job est désactivé. Nous devons l'activer manuellement.
* comments permet de décrire le job.

Comme dit précédemment, la description de chacun de ces paramètres est très réductrice. Nous reviendrons dans nos articles suivants sur ces différents paramètres et sur les différentes versions de cette procédure create\_job.

**5. Pour finir**

Pour éviter de faire un article trop long et ennuyant, nous nous arrêterons ici pour cette introduction. Dans nos prochains articles, nous passerons en revue certains scénarios assez courant d'utilisation du scheduler d'Oracle avec chaque fois un peu plus de détails. N'hésitez pas à poster vos remarques, suggestions ou toutes autre question.