LDETLFB

Extracteur de données
pour analyse dans LDVision

Décisionnel LD SYSTEME

Révision 1 – Octobre 2010 – LDETLFB V1.00 N22

Table des matières

1 - Présentation générale

1.1 - Objectifs

LDETLFB est un outil permettant d'extraire des données des bases gérées par les différents progiciels LD SYSTEME que sont LDCompta, LDPaye et LDNégoce, de mettre en forme ces données et de les charger dans une nouvelle base, dite « entrepôt de données ». Et c'est cet entrepôt de données qui sera ensuite exploité par l'outil d'analyse LDVision.

Cette étape d'extraction et de transformation des données est nécessaire pour plusieurs raisons :

- ⇨ Les données manipulées au quotidien dans les logiciels de comptabilité, paye et gestion commerciale sont optimisées pour une exploitation au travers de ces logiciels. L'organisation des données sous forme « relationnelle » n'est pas optimale pour produire facilement et rapidement des analyses pertinentes : les données sont souvent morcelées dans de très nombreuses tables distinctes, nécessitant donc de nombreuses jonctions entre celles-ci. Alors qu'à l'inverse, pour faciliter l'exploitation des données dans un logiciel d'analyse, il est préférable de disposer de vastes fichiers « à plat », où toutes les données pertinentes sont placées « en colonne » comme s'il s'agissait d'une seule et immense table.
- ⇨ Le fait de reconstituer une nouvelle base de données « décisionnelle » permet de renommer les tables et les champs dans ces tables de façon plus pertinente. La terminologie utilisée sera plus proche de l'utilisateur final, et pourra même, si on le souhaite, être adaptée à chaque entreprise.
- ⇨ Enfin, on pourra aussi « consolider » plusieurs bases distinctes en une seule, pour la comptabilité notamment. Dans LDCompta, on gère une base de données distincte par société. On pourra donc ici regrouper dans une seule base décisionnelle les données comptables de multiples sociétés, ce qui permettra ensuite de faire des analyses croisant ces sociétés.

LDETLFB se présente sous la forme d'une fenêtre principale, permettant de gérer trois entités distinctes :

Les modèles

Un modèle permet d'alimenter une table dans l'entrepôt de données, à partir d'un fichier des bases de LDCompta, LDPaye ou LDNégoce, ou d'une requête SQL basée sur un ou plusieurs de ces fichiers. Un modèle, une fois défini, peut être « exécuté » autant de fois qu'on le souhaite pour rafraîchir les données de l'entrepôt.

Les scénarios

Un scénario facilite l'exécution des modèles, en permettant d'ordonner et de regrouper l'exécution de plusieurs modèles.

Les tâches

Une tâche consiste à planifier l'exécution d'un modèle ou d'un scénario, en s'appuyant sur la gestion des tâches planifiées de Windows. On peut ainsi déclencher à heure régulière, de façon quotidienne ou hebdomadaire, les exécutions des modèles ou scénarios qui permettront d'avoir des données régulièrement tenues à jour dans l'entrepôt de données.

En sus de ces trois entités principales, LDETLFB va également exploiter d'autres entités :

Les connexions

Une connexion définit comment accéder à des données : le type de base de données (Hyper File Classic, HyperFile Client/Serveur, DB/2 via Easycom, ou Firebird), l'emplacement des données ou du serveur de base de données... On distinguera des connexions « sources », où l'on définira le type et l'emplacement des données gérées par les logiciels LD, et des connexions cibles : où se trouve

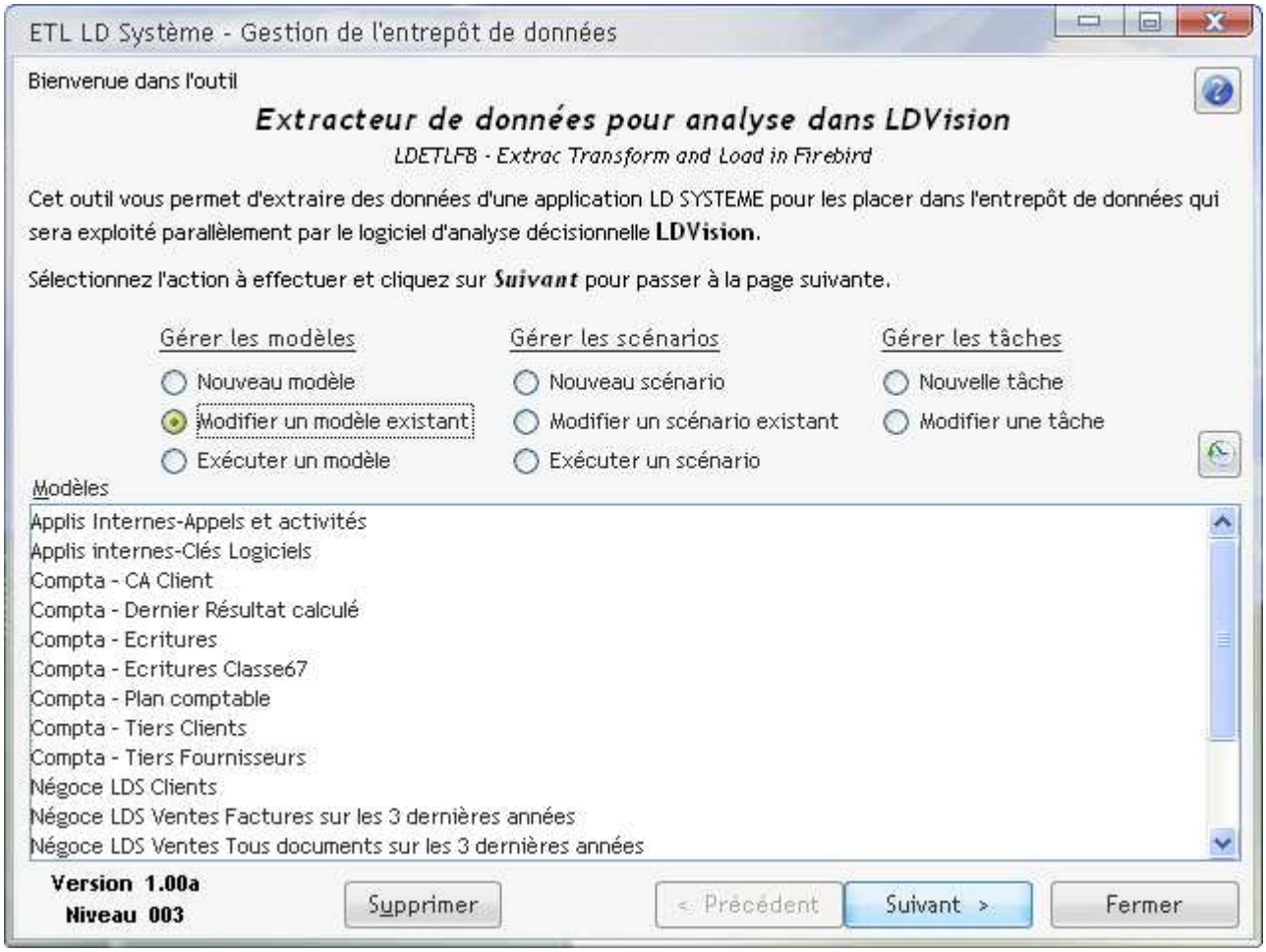
l'entrepôt de données. Les connexions sont classées en familles de connexions ; cela permet à l'exécution d'un modèle ou d'un scénario de remplacer dynamiquement la connexion définie initialement dans le modèle par une autre connexion appartenant à la même famille. Un même modèle peut ainsi être réutilisé pour plusieurs sociétés en comptabilité par exemple, ou plusieurs environnements en paye ou gestion commerciale.

Les bibliothèques de fonctions

On verra plus loin que dans la définition d'un modèle, on peut intégrer des séquences de code pour accéder à des données au delà du fichier (ou de la requête SQL) qui est la source principale du modèle. Pour faciliter la réutilisation de ces séquences de codes dans différents modèles, ou même au sein d'un même modèle pour alimenter plusieurs champs, il est préférable de regrouper ces codes dans une bibliothèque plutôt que de placer le code directement au sein du modèle.

1.2 - La fenêtre principale

La fenêtre initiale de LDETLFB présente les trois entités principales que sont les modèles, les scénarios et les tâches, avec pour chaque entité, la possibilité de créer ou modifier une entité. Pour les modèles et les scénarios, on a également la possibilité de lancer l'exécution d'une entité.



Dès lors que l'on sélectionne une option *Modifier* ou *Exécuter*, le système présente en partie basse la liste des entités du type choisi.

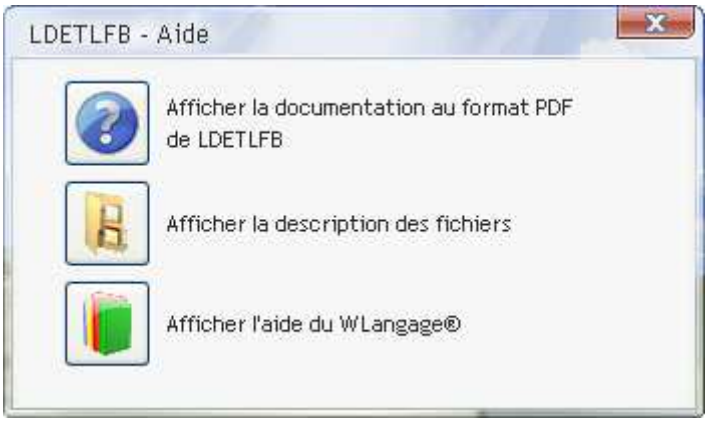
On a également la possibilité de supprimer une entité. Il faut pour cela dans un premier temps cliquer sur le choix *Modifier* pour un des trois types d'entités possibles. Une fois la liste des entités de ce type affichée, cliquez sur le nom de l'entité concernée en partie centrale, puis sur le bouton *Supprimer* en partie basse.

La création d'une entité, comme sa modification, consiste à parcourir un certain nombre de fenêtres, qui vont être présentées successivement chaque fois que l'on clique sur le bouton *Suivant*. Et l'on pourra à tout moment revenir sur le ou les écrans précédents par le bouton *Précédent*. Les différents écrans de création ou modification sont présentés en détail dans les chapitres qui suivent, pour chaque type d'entité.

Toujours dans cette fenêtre principale, on dispose d'un bouton supplémentaire : le bouton *Historique*. Situé en partie centrale, à droite, ce bouton permet d'accéder à une fenêtre présentant l'historique d'exécution des modèles et scénarios, jour par jour.

1.3 - Aide

Il est possible, à tout moment au sein de LDETLFB, d'appuyer sur la touche *F1*. La fenêtre ci-dessous s'ouvre alors, et vous propose trois types d'aide :

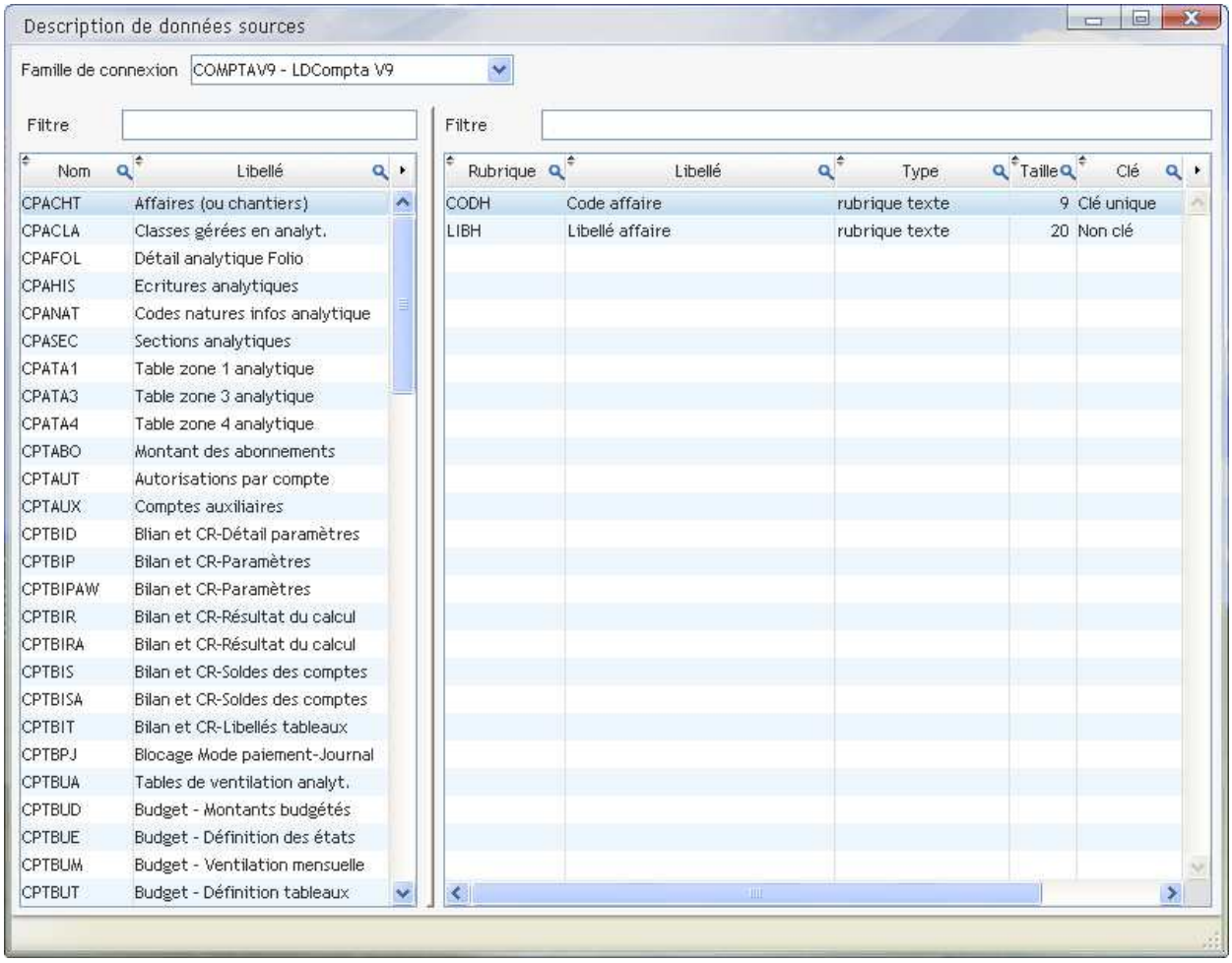


- ☑ La documentation de LDETLFB. Il s'agit de ce document, qui s'ouvre en format *PDF* ;
- ☑ La description des fichiers, décrite plus en détail ci-après
- ☑ L'aide en ligne du langage Windev 12, pour faciliter l'écriture de fonctions en langage Windev, comme on le verra plus loin.

Description des fichiers

Cette fenêtre permet d'accéder à la description de tous les fichiers définis dans les différentes analyses livrées avec LDETLFB. Lorsque vous êtes en cours de création ou de modification d'un modèle, le système affiche par défaut la description des fichiers de l'analyse courante, c'est-à-dire celle correspondant à la famille de connexion sur laquelle est basée le modèle en cours de création ou modification.

En partie haute de la fenêtre, on peut sélectionner une analyse (correspondant donc à une famille de connexion).



La partie gauche de la fenêtre présente la liste des fichiers décrits dans l'analyse sélectionnée en partie haute. Le fait de cliquer sur un des fichiers permet d'obtenir la description de toutes les rubriques du fichier en partie droite.

Une zone de filtre est disponible tant pour filtrer la liste des fichiers que pour filtrer la liste des rubriques d'un fichier. Notez que le filtre s'applique lorsqu'on sort du champ permettant le filtre.

Cette fenêtre peut être conservée affichée en parallèle de la fenêtre principale de LDETLFB, de façon à pouvoir facilement basculer de l'une à l'autre pour rechercher le nom ou la description d'une rubrique.

2 - Création-Modification d'un modèle

La définition d'un modèle se décompose en sept étapes, décrites ci-après en détail.

2.1 - Etape 1 – Connexions source et cible

La connexion source permet de définir où se trouvent les données que l'on souhaite extraire, alors que la connexion cible permet de définir où se trouve l'entrepôt de données que l'on veut « charger ».

Famille de connexion source

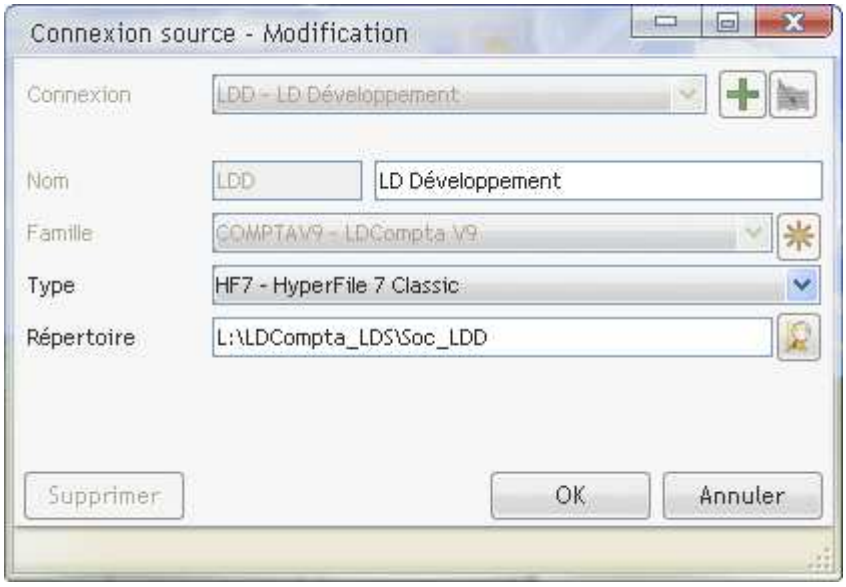
Choisissez dans un premier temps la famille de connexions source. On peut considérer qu'une famille de connexions correspond à un progiciel donné : LDCompta, LDPaye ou LDNégoce. Si la famille souhaitée n'apparaît pas dans la liste déroulante, cliquez sur le bouton situé à droite en regard de la liste déroulante. Vous accédez alors à une fenêtre permettant d'en créer une nouvelle, ou de modifier une famille existante.

Pour créer une nouvelle connexion, cliquez sur le bouton *Plus*. Renseignez ensuite :

- ⇨ Le nom de la famille,
- ⇨ un libellé adéquat
- ⇨ l'analyse correspondant à la base de données de ce logiciel. Une analyse correspond à un fichier portant l'extension *.wdd*. Ces fichiers sont livrés d'une part dans le répertoire des programmes de chaque logiciel, d'autre part dans le sous-répertoire *Analyses* du répertoire des programmes de LDETLFB. Ce sont les fichiers présents dans ce sous-répertoire *Analyses* qui sont présentés ici. Il existe en principe un fichier *.wdd* pour chaque version de chaque logiciel LD.

Connexion source

Une fois la famille de connexion choisie, sélectionnez la connexion source au sein de cette famille. Là aussi, si la connexion souhaitée n'apparaît pas dans la liste déroulante, cliquez sur le bouton situé à droite en regard de cette liste. Vous accédez alors à une fenêtre permettant d'en créer une nouvelle, ou de modifier une connexion existante.



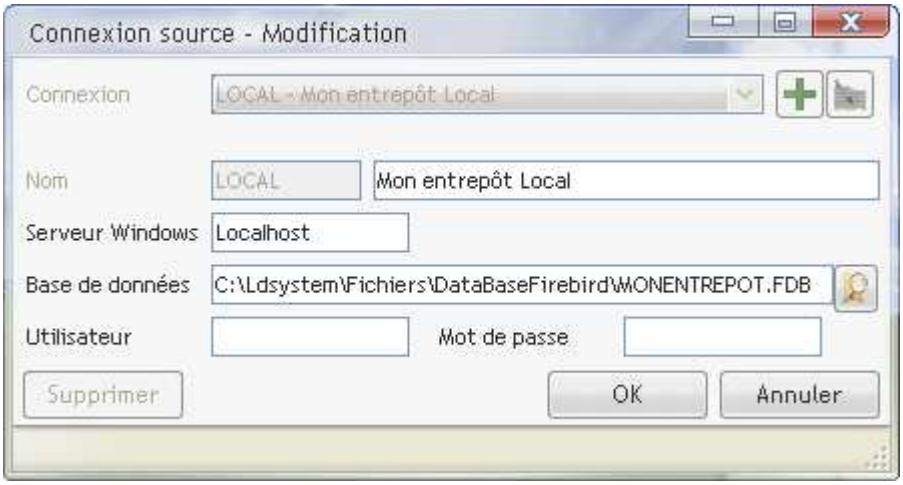
Pour créer une nouvelle connexion, cliquez sur le bouton *Plus*. Renseignez ensuite :

- ⇨ Le nom de la connexion,
- ⇨ un libellé adéquat
- ⇨ le type de connexion : Hyper File Classic, Hyper File Client Serveur, ou encore DB2 via Easycom (uniquement disponible pour LDCompta).
- ⇨ Dans le cas d'une connexion HyperFile Classic, il reste à indiquer le répertoire Windows dans lequel se trouvent les données.
- ⇨ Dans le cas d'une connexion Hyper File Client Serveur, il faut indiquer le nom du serveur Windows sur lequel est installé le serveur HyperFile (ou son adresse IP), le N° de port de ce serveur, un code utilisateur et un mot de passe ayant été définis sur ce serveur et ayant au-moins les droits de lecture sur toutes les tables auxquelles on souhaite accéder, et enfin le nom de la base de données sur ce serveur.
- ⇨ Dans le cas d'une connexion DB2 via Easycom, il faut indiquer le nom du serveur AS/400 (ou son adresse IP), un code utilisateur AS/400 et son mot de passe, cet utilisateur devant disposer des droits de lecture sur toutes les tables auxquelles on souhaite accéder, et enfin le nom de la bibliothèque de données sur ce serveur.

Notez que le plus simple, pour renseigner correctement ces paramètres, est de consulter la fenêtre *Environnement* de chaque logiciel concerné (LDCompta, LDPaye, LDNégoce) au travers de la fenêtre *?/A propos*. Vous y retrouverez toutes les informations nécessaires ici.

Connexion cible

Sélectionnez ensuite la connexion cible, qui sert à identifier votre entrepôt de données. En règle générale, on utilisera un seul entrepôt de données. Il suffira donc de le sélectionner lors de la définition du premier modèle, et celui-ci sera automatiquement sélectionné lors de la création des modèles ultérieurs.



Pour définir l'emplacement de cet entrepôt la première fois, cliquez sur le bouton situé à droite en regard de cette liste. Cliquez ensuite sur le bouton *Plus*. Renseignez ensuite :

- ⇨ Le nom de la connexion cible,
- ⇨ un libellé adéquat
- ⇨ le nom du serveur Windows sur lequel a été installé le serveur Firebird qui va héberger l'entrepôt de données
- ⇨ l'emplacement de la base de données sur ce serveur. Une base de données Firebird est enregistrée sous forme d'un fichier unique portant l'extension *.FDB*. Attention : le chemin d'accès à ce fichier doit être défini sous sa forme « locale » par rapport au serveur Windows. Ainsi, si le fichier *.FDB* est dans le répertoire *Firebird* sur le disque *C:* du serveur, il faudra indiquer *C:\Firebird\MonEntrepot.FDB*, et ce même si ce disque *C:* du serveur est accessible depuis mon poste de travail sous la lettre réseau *S:* par exemple. Ne pas indiquer comme chemin *S:\Firebird\MonEntrepot.FDB* !
- ⇨ un code utilisateur et un mot de passe permettant d'accéder au serveur de données Firebird. A défaut, le système utilisera le code utilisateur et le mot de passe définis en standard sur tout serveur Firebird, à savoir : utilisateur *SYSDBA* et mot de passe *MASTERKEY*.

2.2 - Etape 2 – Source de données

Choisissez tout d'abord le type de la source. Cela peut être un fichier ou une requête SQL.

Si vous choisissez le type *Fichier*, sélectionnez alors le fichier source souhaité, dans la liste qui vous est proposée.

Si vous choisissez le type *Requête SQL*, vous devez saisir le code source d'une requête SQL de type *SELECT*. Une petite aide vous est fournie, vous décrivant succinctement la syntaxe d'une commande SQL de type *SELECT*.

L'intérêt de passer ici une commande SQL est multiple :

- ⇨ Pouvoir effectuer une sélection sur les données d'un fichier, via une expression de sélection suivant le mot-clé *where*
- ⇨ Pouvoir construire une source de données basée sur plusieurs fichiers, au travers d'une jonction entre ceux-ci (mais on verra plus loin qu'il est possible de récupérer des données au-delà du simple fichier ou requête source formulés ici)
- ⇨ Pouvoir construire une source de données qui soit déjà un regroupement de données présentes dans un fichier donné (requête de groupage obtenue par les mots clés *group by)*.

Intérêt supplémentaire : la requête SQL peut faire référence à un « paramètre » du modèle, pour comparaison par exemple entre une rubrique du fichier source sous-jacent et la valeur d'un paramètre passée au modèle lors de son exécution. La mise en œuvre des paramètres est décrite en détail en page 17.

Important : il faut comprendre qu'au final, la table cible de l'entrepôt de données sera alimentée par autant de lignes que l'on a de lignes sélectionnées dans le fichier ou la requête SQL spécifiés ici.

Remarque complémentaire : pour effectuer une sélection sur une date « glissante », vous pouvez utiliser la fonction SQL *ADD_MONTHS*, comme dans l'exemple ci-dessous qui sélectionne tous les en-têtes de bulletins dans une base de données LDPaye des 24 derniers mois par rapport au mois courant :

```
select * from CAENBU where MPAY >= ADD_MONTH(SYSDATE, -24)
```

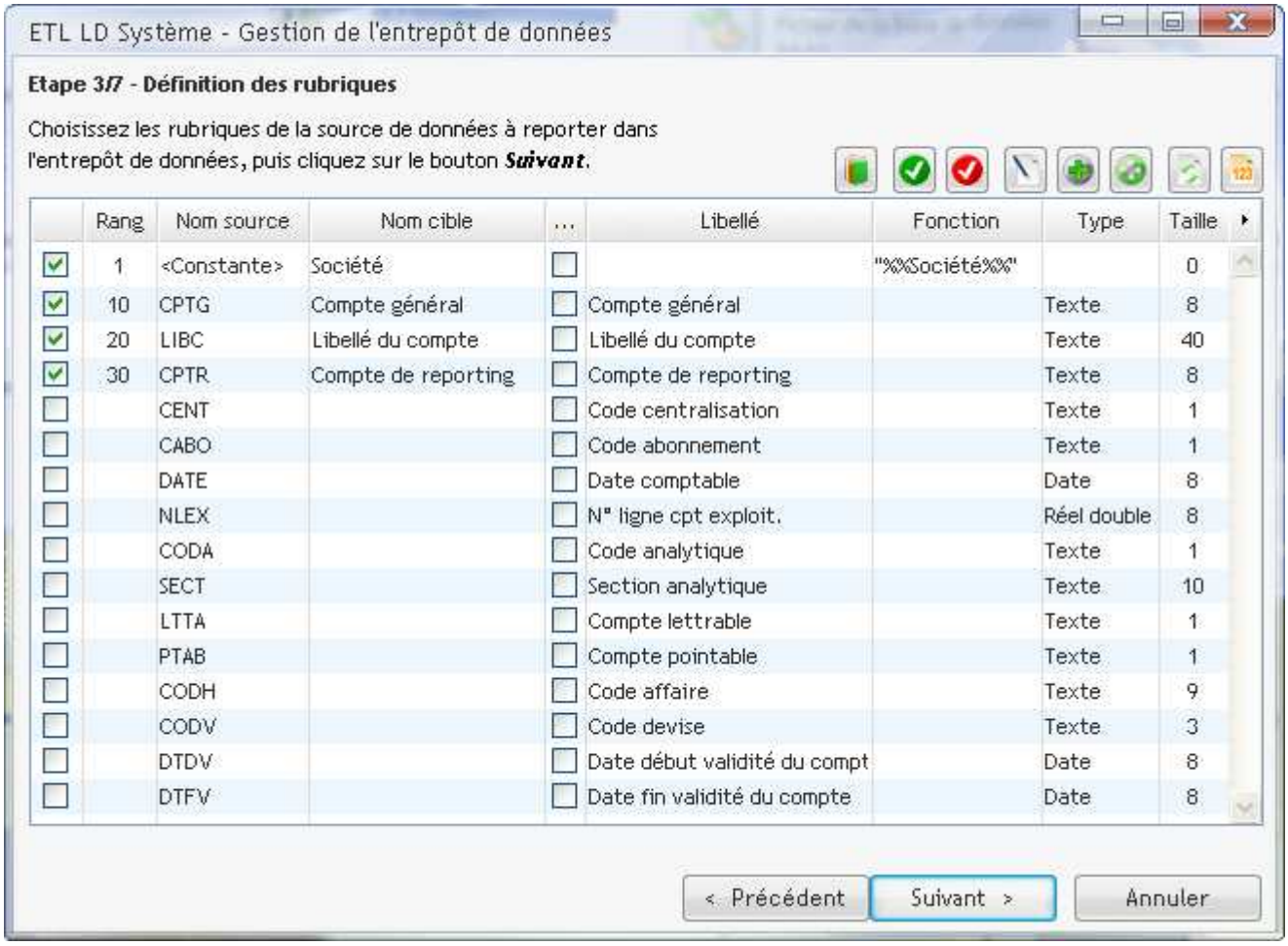
Vous pouvez également, s'agissant d'une commande SQL qui va s'exécuter sur une base HyperFile, faire appel à des fonctions de sélection du langage Windev, en préfixant les noms de fonction Windev par *wl.* :

```
select * from CPTHIS where wl.datedifférence(DATE, DATH) > 0 and  DATH<>''
```

2.3 - Etape 3 – Définition des rubriques

C'est l'étape la plus importante, celle où l'on définit quelles sont les colonnes que l'on souhaite alimenter dans la table cible de l'entrepôt de données.

L'écran se présente sous la forme d'une table, où chaque ligne correspond à une colonne du fichier ou de la requête source définies à l'étape précédente.



Sélection des colonnes

Pour qu'une colonne se retrouve dans la table cible de l'entrepôt de données, il faut la sélectionner (cocher) dans la première colonne. Au fur et à mesure que vous sélectionnez les colonnes souhaitées, celles-ci sont numérotées de 10 en 10 dans la seconde colonne *Rang*. Ce rang est représentatif de l'ordre des colonnes que l'on obtiendra dans la table cible.

Deux boutons situés dans la barre de boutons figurant au dessus de la table (les 2ème et 3ème boutons en partant de la gauche) permettent de sélectionner ou de désélectionner toutes les lignes en un seul clic.

Vous pouvez modifier si nécessaire le rang qui a été proposé par le système, en saisissant un nombre directement dans la 2ème colonne, en lieu et place de celui proposé par le système. Vous pouvez également à tout moment retrier toutes les lignes sélectionnées en fonction de leur rang (avant dernier bouton à droite, dans la barre de boutons figurant au dessus de la table). Et vous pouvez aussi demander à ce que tous les rangs soient renumérotés de 10 en 10 (sans que cela influe sur l'ordre), ce qui permet ensuite de pouvoir insérer plus facilement des lignes à un endroit précis.

Désactivation d'une colonne

En modification d'un modèle, il est possible de désactiver une colonne, plutôt que de la désélectionner. Pour ce faire, il faut décocher l'option *Activé* apparaissant dans la dernière colonne de la table. L'intérêt de cette désactivation est de pouvoir facilement la réactiver ultérieurement ; toute la définition de la colonne est conservée dans le modèle, mais la colonne n'est pas créée et alimentée dans la table cible.

On peut ainsi, à partir d'un modèle unique comportant un maximum de définition de colonnes, améliorer les performances en diminuant le nombre de colonnes réellement créées dans la table cible, mais sans pour autant perdre les définitions des colonnes initiales. Ainsi, si plus tard on souhaite rétablir une des colonnes ayant été désactivées, cela se fera de façon très simple, en cochant à nouveau l'option *Activé*.

Choix du nom des colonnes

En 3ème colonne figure le nom de la colonne dans la table source. En 4ème colonne, vous pouvez saisir le nom souhaité pour la colonne dans la table cible. Pour chaque ligne que vous avez sélectionnée, le système a pré renseigné ce nom cible à partir du libellé du champ, figurant en 6ème colonne.

Prenez beaucoup de soin pour définir ce nom. C'est celui que les utilisateurs de LDVision verront lors de toute analyse de données basée sur cette table cible. Choisissez un nom suffisamment long pour être le plus explicite possible, mais pas trop pour ne pas surcharger les écrans, où l'on est toujours limité par la taille physique de l'écran. Un nom trop long sera probablement tronqué faute de place pour le présenter en totalité, surtout si le contenu (les valeurs affichées ligne par ligne) de cette colonne est court. Evitez par exemple le nom *Code Débit Crédit* pour une colonne comportant les valeurs *D* pour Débit ou *C* pour Crédit. Préférez dans ce cas le nom *D/C*, même s'il est moins explicite. Le nom *Code débit crédit* serait très probablement tronqué dans LDVision, et on ne verrait alors que le début, c'est-à-dire *Code* qui n'est pas plus explicite que *D/C*. Préférez aussi le symbole *N°* au mot *Numéro.* Evitez de préfixer tous les noms par le mot *Code* : *Code groupe*, *Code famille*. Utilisez plutôt directement les termes *Groupe*, *Famille*…

La 5ème colonne (case à cocher) permet de définir un nom « dynamique ». Cela permet de gérer le cas où le nom souhaité pour la colonne dans la table cible doit être déterminé à partir de données elles même contenues dans un fichier de paramètres du logiciel. Cette fonctionnalité de nom dynamique permet aussi d'activer ou désactiver dynamiquement la présence d'un champ dans la table cible : si la fonction de nommage dynamique retourne une chaine vide, la colonne concernée ne sera pas créée dans la table cible (même effet que la désactivation de la colonne, présentée au paragraphe précédent). L'utilisation des noms dynamiques étant très rare dans les logiciels LD, nous ne rentrerons pas ici dans le détail de cette fonctionnalité.

Ajout de colonnes

Il est possible d'ajouter des colonnes dans la table cible qui ne correspondent pas directement à une colonne du fichier ou de la requête source. Cliquez pour cela sur le 5ème bouton en partant de la gauche, sur la barre de boutons figurant au dessus de la table, ou utilisez le raccourci *Ctrl A*.

Une fenêtre d'ajout d'un champ est présentée. Renseignez les éléments demandés :

- ⇨ Le nom cible du champ
- ⇨ Le rang de ce champ dans la table cible (ordre des colonnes)
- ⇨ Le type de champ, à choisir parmi les trois types possibles, détaillés ci-après.

Champ de type Constante

Ce type de champ permet de créer une colonne dans la table cible qui sera alimentée avec la même valeur pour toutes les lignes de la table. Il suffit alors d'indiquer la valeur de la constante, qui peut être soit une valeur numérique, soit une valeur alphanumérique. Dans ce dernier cas, la valeur doit être saisie entre guillemets (le guillemet double, celui de la touche 3).

Un champ de type *Constante* s'avère très utile quand on souhaite consolider dans une même table cible des données venant de plusieurs fichiers sources ayant tous la même structure : fichiers correspondant à des sociétés différentes, ou à des exercices différents. Pour distinguer ces données une fois toutes regroupées dans une même table, on utilisera un champ *Code société*, ou *Exercice*, qui sera alimenté par une valeur *Constante* représentative de la source de données utilisée en entrée : la société ou l'exercice lu.

Notez que la valeur de la constante peut être un « paramètre » du modèle. La mise en œuvre des paramètres est décrite en détail en page 17.

Champ lié dans un autre fichier

Ce type de champ permet d'extraire un libellé associé à un code donné, libellé figurant dans une table annexe. Prenons un exemple : nous avons une table *Ecritures comptables* dans laquelle figure un champ *N° du compte général* . Et nous souhaitons ajouter le libellé de ce compte général, sachant que ce libellé est présent dans une autre table *Plan comptable*.

Pour récupérer ce libellé, il vous suffira de renseigner :

- ⇨ Le nom du fichier relié, à choisir dans la liste déroulante (liste qui présente toutes les tables disponibles pour la famille de connexion source à laquelle est attaché ce modèle)
- ⇨ La clé d'accès dans ce fichier relié, à choisir là aussi dans la liste présentant toutes les clés d'accès au fichier sélectionné
- ⇨ Si la clé d'accès est une clé « composée », la valeur de la première composante de la clé. C'est le cas chaque fois qu'un fichier unique contient plusieurs types de données distincts. Par exemple, dans LDCompta, le fichier des paramètres divers *CPTPDI* contient la table des modes de paiement, la table des natures de pièces, la table des codes TVA … La clé de ce fichier *KPDI*, est une clé composée avec pour première composante un code identifiant la table : *P* pour les modes de paiement, *O* pour les natures de pièce, *T* pour les codes TVA… Et la seconde composante contient le code paiement, le code nature de pièce, le code TVA recherché.
  Si vous devez utiliser une clé composée, il faut donc savoir quel est le code identifiant la table à utiliser. Et une fois ce code connu, le reporter ici, entre guillemets s'il s'agit d'un code alphanumérique.
- ⇨ La valeur de la clé recherchée. Choisissez ici la colonne du fichier source contenant la clé de recherche, dans la liste déroulante qui est proposée. Dans l'exemple présenté ci-dessus d'une écriture comptable avec un N° de compte général, on sélectionnera le champ *N° de compte général* .
- ⇨ La valeur retournée. Le système propose la liste de tous les champs du fichier relié. Sélectionnez celui que vous souhaitez ajouter dans la table cible. Dans l'exemple présenté ci-dessus d'une écriture comptable avec un N° de compte général, on sélectionnera le champ *Libellé du compte*.

⇨ Fonction Windev. Vous pouvez indiquer ici une séquence de code, écrite en langage Windev. Ce code permet d'aller au-delà de la simple référence à un champ du fichier relié. Les principes à respecter dans l'écriture de ce code source sont les mêmes que ceux décrits plus loin pour un champ de type *Calculé*. Mais on peut en sus référencer ici n'importe quelle colonne du fichier relié, par la syntaxe *NomFichierRelié.NomChampFichierRelié*. Prenons un exemple assez simple : nous souhaitons retourner le libellé du compte général (champ *LIBC* dans le fichier *CPTPLA*), mais en minuscule pour faciliter la lecture (sauf la 1ère lettre en majuscule). Nous indiquerons alors :

Renvoyer Majuscule(CPTPLA.LIBC[[1]])+Minuscule(CPTPLA.LIBC[[2 à]])

Champ calculé (compilation dynamique)

Ce type de champ offre une très grande souplesse, puisque l'on va pouvoir « calculer » la valeur à porter dans ce champ par une séquence de code écrite directement en langage Windev. Mais elle suppose un minimum de connaissance de ce langage.

Il faut dans un premier temps choisir le champ « modèle » pour ce nouveau champ. C'est ce qui va permettre de définir le type du nouveau champ dans la base cible : numérique, alphanumérique, date… Notez que dans le cas d'un champ alphanumérique, le nouveau champ sera créé dans la table cible avec une longueur maximale de 200 caractères, quelle que soit la longueur du champ utilisé comme modèle.

Et on définit ensuite la fonction de calcul. Ici, tout ce qu'il est possible d'écrire en langage Windev est autorisé.

L'objectif de cette séquence de code est de calculer et de renvoyer la valeur qui sera portée dans le champ cible. La séquence de code devrait donc comporter un ou plusieurs mots-clés *Renvoyer*. Si tel n'est pas le cas, le système ajoute ce mot-clé *Renvoyer* juste avant la séquence de code écrite.

Dans cette séquence de code, on peut accéder à toutes les colonnes du fichier ou de la requête SQL source, en préfixant le nom de la colonne (celui apparaissant en 3ème colonne dans la table de définition des rubriques) par le caractère $.

Si l'on reprend l'exemple précédent du fichier *Ecritures comptables* comportant un N° de compte (colonne *CPTG*), pour lequel on souhaite ajouter le libellé du compte (colonne *LIBC* dans le fichier *CPTPLA*) (sans passer par un champ lié comme présenté au paragraphe précédent), le code serait :

HLitRecherchePremier("CPTPLA","CPTG",$CPTG)
SI HTrouve() alors Renvoyer CPTPLA.LIBC sinon Renvoyer ""

Notez que là aussi, il est possible de référencer, dans la séquence de code du champ calculé, un « paramètre » du modèle. La mise en œuvre des paramètres est décrite en détail en page 17.

Complément d'information : vous trouverez à partir de la page 25 une présentation simplifiée du langage Windev, avec les principaux éléments pouvant être mis en œuvre ici avec profit.

Modification d'un champ ajouté

Pour ces trois types de champs ajoutés, une fois l'ajout validé par le bouton *OK*, le champ est ajouté en tête de la table de définition des rubriques. Utilisez le bouton *Retrier* pour que cette table soit retriée en fonction des rangs des différents champs sélectionnés ou ajoutés.

Pour les champs de type *Constante* ou *Calculés*, la colonne *Fonction* présente la valeur de la constante ou le code source associé au champ. Il est possible d'intervenir directement en modification sur cette valeur ou ce code source. Dans le cas d'un code source constitué de plusieurs lignes, il peut s'avérer plus confortable de passer par la fenêtre de saisie du code source, pour bénéficier par exemple de la coloration syntaxique. Celle-ci peut être obtenue soit en cliquant sur le bouton *Editer le code source* (4ème bouton en

partant de la gauche, dans la barre de boutons présentée au dessus de la table), soit par un clic sur le code à modifier, suivi d'un double-clic sur ce même code.

Suppression d'un champ ajouté

Il n'existe pas de bouton *Supprimer* pour un champ ayant été ajouté par l'une des méthodes décrites ci-dessus. Il suffit de le « désélectionner », en décochant ce champ dans la 1ère colonne de la table de définition des rubriques. Ainsi, ce champ ne figurera pas dans la table cible de l'entrepôt de données.

Lors de l'enregistrement final du modèle, si le champ n'est pas sélectionné, il ne sera pas enregistré, et il ne figurera donc plus lors d'un rappel ultérieur de ce modèle en modification.

Duplication d'un champ ajouté

Un bouton *Dupliquer un champ* (6ème bouton en partant de la gauche) est proposé sur la barre de boutons affichée au dessus de la table de définition des rubriques. Il permet en un seul clic de créer un nouveau champ par duplication d'un autre champ ayant été ajouté précédemment. Il suffit ensuite de modifier ce champ : le nom « cible » du champ d'une part, et le code source permettant de calculer la valeur du champ d'autre part.

Bibliothèques de fonctions

Lorsqu'une séquence de code est nécessaire pour initialiser plusieurs champs différents, il est préférable de placer ce code dans une bibliothèque, pour qu'il soit facilement réutilisable.

LDETLFB est livré en standard avec une bibliothèque de fonctions par progiciel (LDCompta, LDPaye, LDNégoce), bibliothèques comportant des exemples de codes propres à chaque logiciel.

Il est possible soit d'enrichir ces bibliothèques, soit (ce qui est conseillé) de créer vos propres bibliothèques additionnelles.

L'accès aux bibliothèques est possible par le bouton *Editer les bibliothèques de fonctions* à l'étape 3 de définition des rubriques d'un modèle (1er bouton à gauche sur la barre de boutons). Quand on clique sur ce bouton, le système présente la fenêtre de gestion des bibliothèques. Cette fenêtre peut être conservée ouverte en parallèle de la fenêtre principale de LDETLFB, ce qui permet de consulter la liste des fonctions offertes par chaque bibliothèque, et la syntaxe d'appel de chaque fonction.

En partie haute de cette fenêtre, on peut choisir la bibliothèque à afficher. Utilisez les boutons présentés en haut à droite pour créer une nouvelle bibliothèque, ou accéder en modification à la bibliothèque dont le nom est sélectionné à gauche.

En partie centrale est présenté le code des différentes fonctions composant la bibliothèque. Chaque nouvelle fonction est repérée par le mot-clé *Fonction* suivi du nom de la fonction et de la liste des paramètres attendus (entre parenthèses), le tout figurant en caractères gras soulignés, en noir.

Pour utiliser l'une de ces fonctions dans une séquence de code source, il suffira de porter le nom de la fonction, suivi entre parenthèses des différentes valeurs des paramètres.

Exemple de fonction

La fonction ci-dessous retourne la valeur d'un cumul donné, pour un salarié et un mois donné.

```
FONCTION LireCumul(NomCumul, pCOSO,pNPPE,pMPAY,pNOBU)

HLitRecherchePremier("CACUMU","KCUMU",[pCOSO,pNPPE,NomCumul,pMPAY,pNOBU])
SI HTrouve() ALORS RENVOYER CACUMU.VACU SINON RENVOYER 0
```

Pour utiliser cette fonction au sein d'un modèle qui serait basé sur le fichier des en-têtes bulletins, dans le code d'un champ devant contenir la valeur du cumul Brut fiscal (cumul nommé *BRUFIM* dans LDPaye), voici le code à insérer :

```
RENVOYER Paye.LireCumul("BRUFIM", $COSO, $NPPE, $MPAY, $NOBU)
```

Notez que le nom de la fonction doit être préfixé par le nom de la bibliothèque. Les valeurs des paramètres passés ici sont les valeurs contenues dans l'enregistrement courant du fichier des en-têtes bulletins : code société, N° matricule du salarié, mois de paye et N° du bulletin. On y accède donc par le nom de la colonne dans le fichier source, préfixé par le caractère *$*, comme cela a été décrit plus haut.

Partage de valeurs entre différents champs (variables locales et globales)

Il peut être utile parfois de faire appel, dans une séquence de code d'un champ donné, à des éléments ayant déjà été calculés dans un champ précédent. Cela peut accélérer le traitement, en évitant de multiples accès à la base de données.

Pour ce faire, LDETLFB propose un système de variables. Avec deux types de variables :

- des variables locales, dont la valeur n'est mémorisée que le temps de traitement d'une ligne du fichier ou de la requête source. A chaque nouvelle ligne, ces variables sont effacées
- des variables globales, dont la valeur est mémorisée durant toute l'exécution du modèle. On peut ainsi calculer une variable donnée, puis réutiliser le résultat de ce calcul pour toutes les lignes qui suivent dans le modèle.

L'accès à ces variables se fait au travers d'un tableau associatif. La syntaxe à utiliser, pour une variable locale, est la suivante : `VARL["<NomVariable>"]`

Dans le cas d'une variable globale, la syntaxe sera `VARG["<NomVariable>"]`

Exemple : `VARL["BrutFiscal"]` ou *BrutFiscal* est le nom de ma variable

Si on reprend l'exemple précédent qui lisait la valeur du cumul brut fiscal via une fonction, on pourrait écrire le code suivant :

```
VARL["BrutFiscal"] = Paye.LireCumul("BRUFIM", $COSO, $NPPE, $MPAY, $NOBU)
RENVOYER VARL["BrutFiscal"]
```

Une variable, quelle soit locale ou globale, n'a pas à être déclarée. La première utilisation de la variable vaut déclaration. De même, ces variables n'ont pas de type à priori. C'est la valeur d'initialisation qui va définir le type, numérique ou alphanumérique.

2.4 - Etape 4 – Fichier cible

Cette étape permet de définir le nom de la table qui sera créée ou alimentée dans l'entrepôt de données, et son mode de remplissage.

Choisissez tout d'abord le nom de la table, qui doit lui aussi être le plus explicite possible. Seules les lettres majuscules et les chiffres sont acceptés pour ce nom ; les espaces ne sont pas autorisés.

Vous pouvez également définir des zones clés pour la table cible, en distinguant la clé principale, et de une à cinq clés supplémentaires. Attention : si vous indiquez une clé principale, prenez garde au fait qu'il doit y avoir unicité sur cette clé principale dans la table. De plus, cette clé ne doit jamais comporter de valeur « nulle » au sens SQL. Même en choisissant une colonne qui est « clé unique » dans le fichier source, comme le N° d'écriture *NECR* dans le fichier des écritures comptables *CPTHIS* de LDCompta, cela peut poser problème dans la table cible si vous consolidez dans une même table les données de plusieurs sociétés : on aura des doublons sur ce N° d'écriture entre les différentes sociétés.

Pour les clés « supplémentaires », il n'y a pas de restriction sur l'unicité et la non nullité des valeurs pour les colonnes déclarées comme « clé ».
L'intérêt de déclarer une colonne comme critère clé est d'optimiser les temps d'ouverture des univers depuis LDVision, dans deux cas de figure :

- ⇨ L'univers comporte une sélection à l'ouverture : par exemple, sélection d'une période sur une date de la table sous-jacente à l'univers. Si la colonne date en question est déclarée comme étant clé, la sélection sera plus rapide ;
- ⇨ L'univers comporte des jonctions entre plusieurs tables sous-jacentes ; là aussi, si les différentes zones utilisées pour faire la jonction entre les tables sont déclarées clés, la jonction sera plus efficace, et le temps d'ouverture de l'univers sera donc plus rapide.

Mais cela n'est sensible que sur des tables ayant un nombre d'enregistrements significatifs : plusieurs dizaines de milliers d'enregistrement. Par exemple, si on consolide dans une seule table les écritures comptables de plusieurs sociétés et plusieurs exercices, la table résultant comportant alors plusieurs centaines de milliers d'écritures, il peut être intéressant d'ajouter comme zone clé le code société, le code exercice ou la date comptable, selon le ou les critères de sélection définis pour l'univers des écritures comptables. Ainsi, l'ouverture de l'univers dans LDVision pour une société donnée ou une période donnée sera optimisée.

Vient ensuite le mode de gestion de la table, avec 3 choix possibles :

- ⇨ *Récréation systématique de la table* : la table sera créée si elle n'existe pas dans l'entrepôt ; elle sera effacée avant d'être récréée si elle existe déjà.
- ⇨ *Création si inexistante, Ajout des données sinon* : la table sera créée si elle n'existe pas dans l'entrepôt ; si elle existe déjà, les données issues de ce modèle seront simplement ajoutées à la table.
- ⇨ *Création si inexistante, Suppression sélective des données sinon* : ce choix est identique au précédent, à ceci près que l'on peut définir une requête SQL de suppression. Cette requête SQL sera exécutée sur la table cible, si celle-ci existe, avant de procéder à l'ajout des données. Cela permet de faire en quelque sorte une opération de type « Annule et Remplace ».
  La requête SQL de suppression est exécutée sur la table cible de l'entrepôt ; elle doit donc être écrite avec une syntaxe compatible *Firebird*. Les noms de champs apparaissant dans celle-ci doivent être portés entre guillemets doubles ; ils doivent correspondre à ceux que l'on voit dans la colonne *Nom cible* dans la table de définition des rubriques (étape précédente).
  Il est possible également d'utiliser ici des paramètres du modèle. La mise en œuvre des paramètres est décrite en détail en page 17.

Cette dernière option de suppression sélective s'avère très utile quand on consolide dans une même table cible des données venant de plusieurs sources (plusieurs sociétés ou plusieurs exercices). On pourra ainsi commencer par supprimer toutes les données correspondant à la société ou à l'exercice pour lequel on va « rafraîchir» les données.

Remarque complémentaire : pour effectuer une sélection sur des dates, il faut spécifier les dates au format *AAAAMMJJ* dans le cas d'une base HyperFile, au format *JJ.MM.AAAA* dans le cas d'une base Firebird. Et dans les deux cas, la valeur doit être portée en quottes.

Exemple HyperFile :

```
select * from CPTHIS where DATE between '20090101' and '20091231'
```

Exemple Firebird :

```
delete from ECRITURES where "DATE" between '01.01.2009' and '31.12.2009'
```

2.5 - Etape 5 – Initialisation des paramètres

On a vu précédemment qu'un modèle pouvait faire référence à des paramètres. L'intérêt des paramètres est de pouvoir exécuter un même modèle, soit directement, soit au travers d'un scénario, plusieurs fois avec des valeurs de paramètres différentes. Cela évite d'avoir à modifier le modèle avant chaque exécution.

Des paramètres peuvent être référencés :

- Dans la définition de la requête source (étape 2)
- Dans la valeur d'un champ ajouté de type *Constante* (étape 3)
- Dans le code source d'un champ ajouté de type *Calculé* (étape 3)
- Dans la requête de suppression sélective des données (étape 4)

Pour définir un paramètre, il suffit d'indiquer, dans l'un des endroits listés ci-dessus, le nom du paramètre souhaité, en l'encadrant par un double symbole %. Par exemple, *%%Exercice%%* est un paramètre valide, de même que *%%DateFinExercice%%.*

Notez que selon l'endroit où le paramètre est référencé, et si le type de valeur attendue pour ce paramètre est alphanumérique, il faudra en plus spécifier le nom du paramètre entre quottes simples (si on est au sein d'une requête SQL) ou entre guillemets doubles si on est au sein d'une séquence de code Windev.

Le tableau présenté à l'étape 5 vous donne la liste de tous les paramètres que le système a identifiés dans la définition du modèle. Et pour chacun d'entre eux, vous pouvez indiquer une valeur par défaut.

Cette valeur pourra bien entendu être remplacée à chaque demande d'exécution du modèle, qu'il s'agisse d'une exécution directe, ou d'une exécution au travers d'un scénario regroupant plusieurs modèles.

2.6 - Etape 6 – Sauvegarde du modèle

Indiquez le nom sous lequel ce modèle va être enregistré. Choisissez toujours un nom le plus explicite possible. Pensez également à classer vos modèles, en les préfixant intelligemment. Par exemple, tous les modèles relatifs à la comptabilité pourraient être préfixés par « *Compta –* » alors que ceux relatifs à la paye pourraient être préfixés par « *Paye –* ». Cela facilitera grandement l'accès à ces modèles quand leur nombre va croître.

Dans le cas de la modification d'un modèle, si vous changez le nom du modèle à cette étape, le modèle modifié sera enregistré avec ce nouveau nom. Le modèle initial sera conservé avec l'ancien nom. Il faudra ensuite le supprimer s'il n'est pas nécessaire de le conserver. Le changement de nom d'un modèle revient en fait à dupliquer un modèle existant.

Attention : si le modèle est référencé dans un scénario, le fait de modifier son nom par le procédé décrit ci-dessus n'a aucun impact direct sur le scénario. Il faudra ensuite aller modifier le contenu du scénario, pour référencer le nouveau nom du modèle si c'est ce que l'on souhaite.

Les modèles s'enregistrent dans des fichiers portant l'extension *.mdl*, dans le sous-répertoire nommé *Modèles* du répertoire des programmes de LDETLFB.

2.7 - Etape 7 – Exécution du modèle

Cette dernière étape vous permet d'enchainer directement l'exécution du modèle que vous venez de créer ou modifier. Il suffit pour cela de cocher l'option d'exécution qui est proposée.

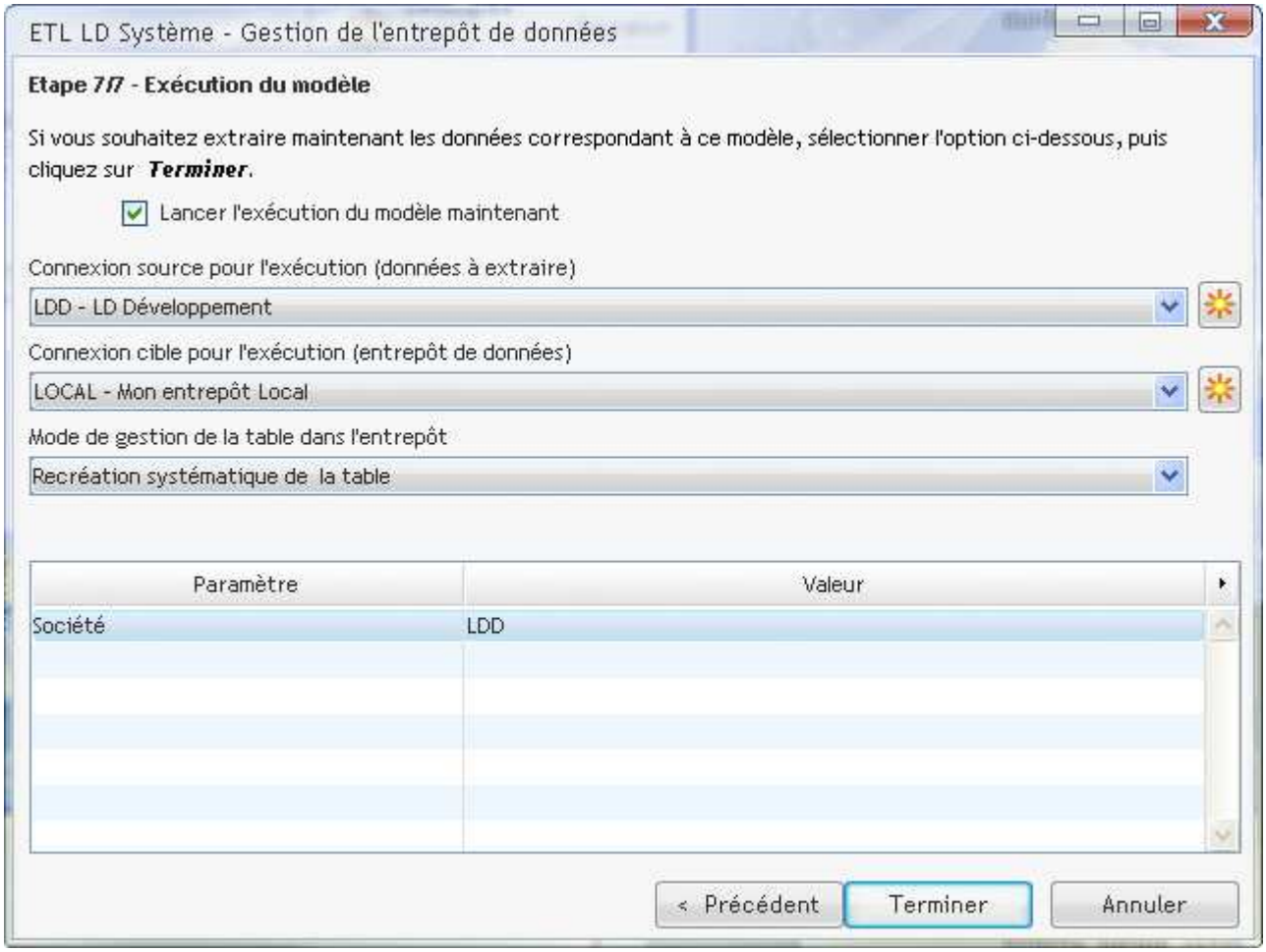
L'écran qui est proposé ici est en fait le même que celui qui est proposé lorsqu'on choisit l'option *Exécuter un modèle* depuis la fenêtre principale. Cet écran est décrit en détail au paragraphe suivant.

Suite à cette dernière étape, le système revient sur l'écran principal, qui présente la liste de tous les modèles, y compris celui que vous venez de créer le cas échéant.

3 - Exécution d'un modèle

En sélectionnant l'option *Exécuter un modèle* sur l'écran principal, on arrive directement à l'étape 7 de création ou modification d'un modèle, l'étape d'exécution.

Il est possible de corriger certaines options pour l'exécution à venir seulement. Notez bien que les valeurs par défaut inscrites dans la définition du modèle ne seront pas impactées par ces modifications.



Connexion source

Vous pouvez remplacer la connexion source ayant été définie dans le modèle, mais le choix est limité aux seules connexions source attachées à la famille de connexions ayant été spécifiée dans le modèle. Vous pouvez même créer une nouvelle connexion si nécessaire, par le bouton figurant à droite de la liste déroulante.

Connexion cible

Vous pouvez remplacer également la connexion cible, pour le cas où vous gérez plusieurs entrepôts de données. Gérer plusieurs entrepôts peut permettre par exemple de mieux répondre à des contraintes de sécurité.

Le mode de gestion de la table

Vous pouvez redéfinir le mode de gestion de la table, parmi les 3 choix possibles :

- ⇨ *Récréation systématique de la table*

- ⇨ *Création si inexistante, Ajout des données sinon*
- ⇨ *Création si inexistante, Suppression sélective des données sinon*

Les valeurs des paramètres

Si le modèle fait référence à un ou plusieurs paramètres, vous pouvez redéfinir en partie basse les valeurs de ces différents paramètres.

Cliquez ensuite sur le bouton *Terminer* pour lancer l'exécution du modèle. Une fenêtre d'exécution va se superposer à la fenêtre principale, avec une jauge d'avancement. Vous pouvez éventuellement demander l'abandon de l'exécution du modèle, en cliquant sur le bouton *Annuler* de cette fenêtre d'exécution.

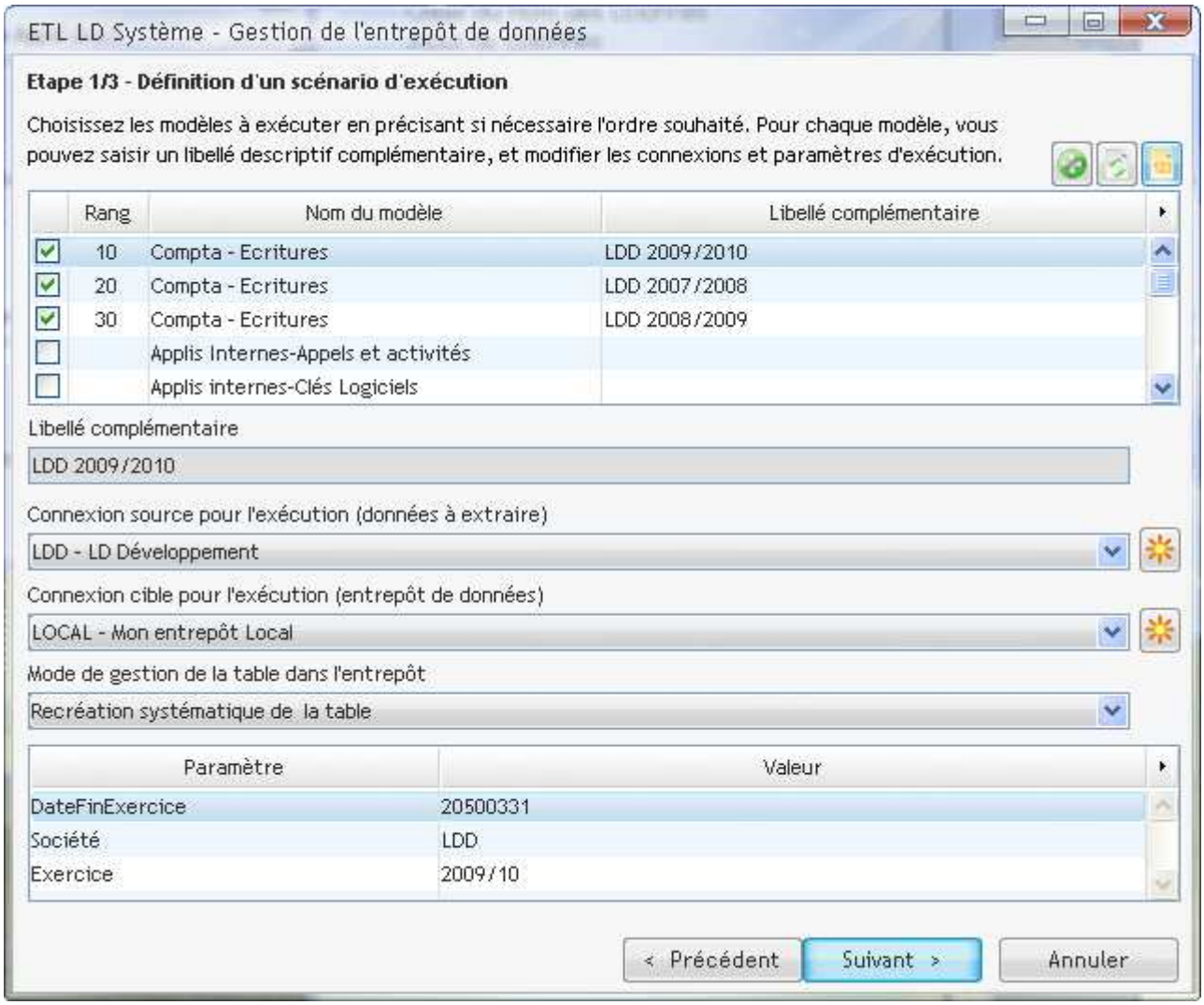
En fin d'exécution, le système revient sur l'écran principal.

Chaque exécution est « tracée » dans un historique d'exécution, historique qui peut être consulté depuis la fenêtre principale, par le bouton *Historique* situé en partie centrale, à droite.

4 - Gérer les scénarios

4.1 - Choix des modèles

La première étape de définition d'un scénario consiste à sélectionner un ou plusieurs modèles à exécuter, dans la liste de tous les modèles qui est proposée.



Pour choisir un modèle, il suffit de le sélectionner en cliquant dans la 1ère colonne. Au fur et à mesure de vos choix, les modèles sélectionnés sont numérotés de 10 en 10 dans la seconde colonne *Rang*. Ce rang est représentatif de l'ordre dans lequel seront exécutés les modèles au sein du scénario. Ce rang a une grande importance lorsque plusieurs modèles mettent à jour une même table cible, l'un des modèles (en principe, le premier à s'exécuter) provoquant la recréation systématique de la table cible, les autres modèles venant ensuite ajouter des données dans la table cible.

Deux boutons situés dans la barre de boutons figurant au dessus de la table (les 2ème et 3ème boutons en partant de la gauche) permettent de sélectionner ou de désélectionner toutes les lignes en un seul clic.

Vous pouvez modifier si nécessaire le rang qui a été proposé par le système, en saisissant un nombre directement dans la 2ème colonne, en lieu et place de celui proposé par le système. Vous pouvez également à tout moment retrier tous les modèles sélectionnés en fonction de leur rang (avant dernier bouton à droite, dans la barre de boutons figurant au dessus de la table). Et vous pouvez aussi demander à ce que tous les

rangs soient renumérotés de 10 en 10 (sans que cela influe sur l'ordre), ce qui permet ensuite de pouvoir intercaler plus facilement des modèles à un endroit précis.

Pour chaque modèle sélectionné, vous pouvez jouer avec les options d'exécution, comme on peut le faire lors de l'exécution directe d'un modèle : choix des connexions sources et cible choix du mode de gestion de la table cible, choix des valeurs des éventuels paramètres du modèle.

Vous pouvez également saisir un libellé complémentaire, libellé représentatif à la fois du modèle sélectionné et des choix d'exécution qui ont été opérés : connexion source, valeurs des paramètres… Ce libellé complémentaire figure dans la table présentée en partie haute, et peut faciliter la maintenance des scénarios, surtout lorsqu'un scénario regroupe un grand nombre de modèles.

4.2 - Réutilisation d'un modèle

Il est possible, au sein d'un scénario, d'exécuter plusieurs fois un même modèle, avec des options d'exécutions différentes : changement de connexion source, ou valeurs de paramètres différentes.

Pour réutiliser un modèle, placez-vous sur le modèle à réutiliser dans la table des modèles en partie haute (modèle qui doit être sélectionné, c'est-à-dire coché dans la 1ère colonne, sans quoi on ne peut parler de réutilisation), puis cliquez sur le bouton *Réutiliser le modèle* (3ème bouton en partant de la gauche). Une nouvelle ligne est ajoutée en tête de la table, ligne totalement identique à celle sur laquelle on était placé au moment du clic sur le bouton.

Il ne reste plus qu'à modifier le rang d'exécution de cette nouvelle ligne, et à adapter les options d'exécution correspondant à cette nouvelle ligne.

Cette opération peut être répétée, si le modèle doit être réutilisé plus d'une fois.

4.3 - Sauvegarde du scénario

Une fois la définition du scénario achevée, cliquez sur le bouton *Suivant*.

Indiquez le nom sous lequel ce scénario va être enregistré. Choisissez toujours un nom le plus explicite possible.

Dans le cas de la modification d'un scénario, si vous changez le nom du scénario à cette étape, le scénario modifié sera enregistré avec ce nouveau nom. Le scénario initial sera conservé avec l'ancien nom. Il faudra ensuite le supprimer s'il n'est pas nécessaire de le conserver. Le changement de nom d'un scénario revient donc à dupliquer un scénario existant.

Les scénarios s'enregistrent dans des fichiers portant l'extension *.scl*, dans le sous-répertoire nommé *Modèles* du répertoire des programmes de LDETLFB.

4.4 - Exécution d'un scénario

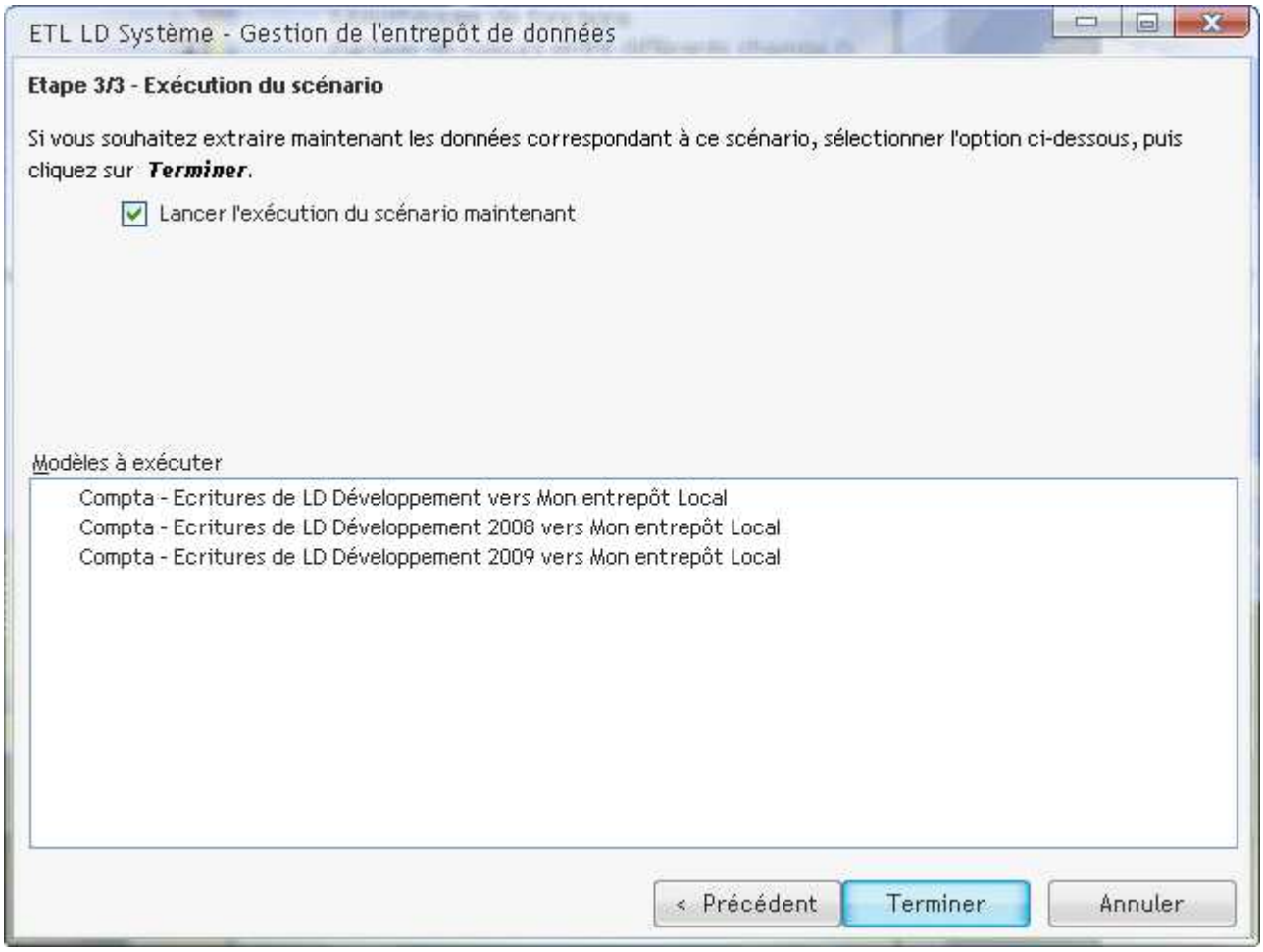
Il y a deux façons d'exécuter un scénario (si l'on omet l'exécution au travers d'une tâche, décrite au chapitre suivant) :

- ⇨ Soit en sélectionnant l'option *Lancer l'exécution du scénario maintenant* proposé sur l'écran qui suit l'enregistrement du scénario en cours de création ou de modification
- ⇨ Soit en sélectionnant l'option *Exécuter un scénario* directement depuis la fenêtre principale.

Dans les deux cas de figure, le système affiche la liste des modèles qui composent ce scénario, dans l'ordre où ils vont être exécutés.

Durant l'exécution du scénario, le système présente successivement la fenêtre d'exécution pour chaque modèle exécuté, avec une jauge d'avancement. Vous pouvez là aussi demander l'abandon de l'exécution du

modèle, en cliquant sur le bouton *Annuler* de cette fenêtre d'exécution. En cas d'abandon de l'exécution d'un modèle, le scénario entier est abandonné.



Chaque exécution d'un scénario est « tracée » dans l'historique d'exécution, avec le détail de chaque modèle exécuté au sein du scénario.

5 - Gérer les tâches

Rappelons qu'une tâche permet de planifier l'exécution d'un modèle ou d'un scénario à une heure précise, de façon quotidienne ou hebdomadaire par exemple.

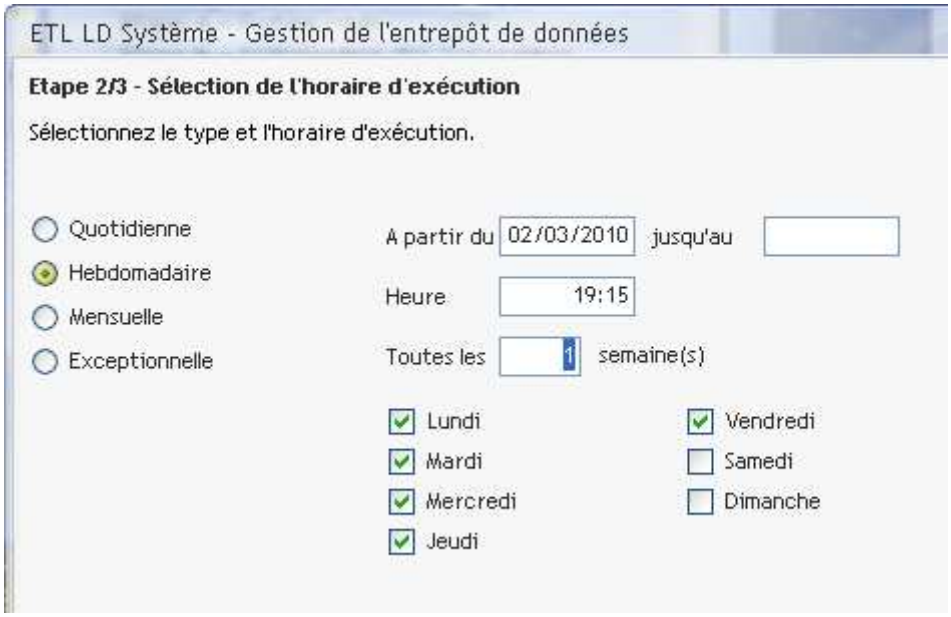
Les tâches, contrairement aux modèles et scénarios, ne s'enregistrent pas sous forme de fichier texte dans le sous répertoire *Modèles*. Le système exploite uniquement la gestion des tâches planifiées de Windows. La liste des tâches qui est présentée dans la fenêtre principale, lorsqu'on sélectionne l'option *Modifier une tâche*, est donc le reflet de ce que l'on peut voir dans la gestion des tâches planifiées de Windows du poste de travail sur lequel LDETLFB a été lancé. Notez que toutes les tâches planifiées ne sont pas présentées ici ; le système ne présente que les tâches qui ont été créées par LDETLFB, qui sont repérées par une mention *<LDETLFB>* dans le commentaire associé à la tâche.

5.1 - Création/Modification d'une tâche

Sur le premier écran de création d'une tâche, vous devez sélectionner le modèle, ou plus souvent le scénario, dont vous souhaitez planifier l'exécution.

Sur le deuxième écran, vous allez définir le planning d'exécution. De nombreux choix de planification sont offerts. Le choix le plus fréquent sera le choix *Hebdomadaire*, pour lequel vous pourrez spécifier :

- L'heure d'exécution
- Le rythme (en principe, toutes les semaines)
- Le ou les jours de la semaine (en principe, tous les jours du lundi au vendredi)



Pour cette planification, respectez quelques règles :

- Essayer de planifier l'exécution à des moments où les bases de données de production (les données « source ») ne font plus l'objet de mise à jour, et où le serveur (ou le poste de travail si vous ne disposez pas d'un serveur) sur lequel ces bases sont hébergées est peu sollicité ; en fin de journée, entre 12Het 14H, ou dans la nuit.
- Attention à choisir une heure d'exécution où le serveur est effectivement en fonction (pour le cas où le serveur est arrêté la nuit par exemple) !

- Evitez dans la mesure du possible les plages durant lesquelles les sauvegardes des données sont en cours d'exécution (souvent aux alentours de 20H le soir).

Sachez que la durée d'exécution des scénarios est directement liée aux volumes de vos données. Si vous gérez des centaines de milliers de lignes de facturation par exemple, et que vous souhaitez rafraîchir chaque nuit la table contenant ces lignes de facturation, cela peut nécessiter des heures de traitement. Il peut s'avérer plus pertinent de ne recharger la totalité de la table qu'une fois par semaine (le dimanche par exemple, quand le serveur n'a rien d'autre à faire), et de ne procéder au quotidien qu'au rafraichissement des données des 3 ou 6 derniers mois (qui seules sont censées faire l'objet de modification en temps normal). Cela peut se faire assez facilement avec un seul modèle disposant d'un paramètre pour la sélection des données à extraire (Nombre de mois), et qu'on lance une fois le dimanche avec 36 mois par exemple et remplacement systématique de la table cible, et une fois chaque jour du lundi au vendredi, avec extraction des données des 6 derniers mois seulement, et une requête de suppression sélective dans la table cible de ces 6 derniers mois.

IMPORTANT : pour que les tâches planifiées le soient sur un serveur Windows, il faut absolument que l'application LDETLFB s'exécute sur ce serveur au moment où l'on définit ces tâches. Il ne suffit pas de lancer cette application depuis un poste de travail accédant à l'application LDETLFB au travers d'un disque réseau. Il faut réellement que l'application soit lancée sur le serveur, que ce soit en mode *Console* (on a ouvert une session directement sur la console du serveur) ou en mode *TSE* (on a ouvert une session sur le serveur par une connexion Bureau à distance).

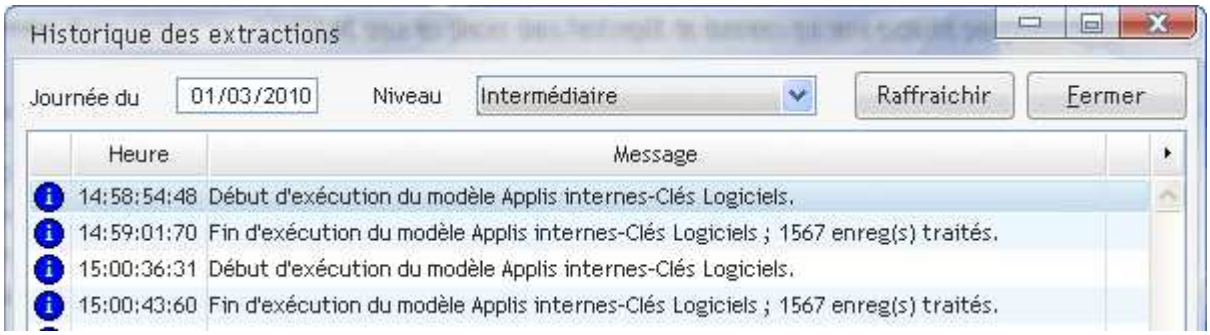
5.2 - Contrôle de l'exécution des tâches

Il y a deux façons de contrôler l'exécution régulière des tâches planifiées :

- Observer depuis LDVision ou FlameRobin le contenu de la table *ENTREPOT*. Cette table est automatiquement mise à jour à chaque exécution de modèle. Elle contient une ligne par table mise à jour dans l'entrepôt, avec pour chaque ligne :
  - Le nom de la table
  - Le nombre d'enregistrements présents dans la table suite à la dernière mise à jour
  - La date et heure de dernière création et de dernière mise à jour de cette table
  - Le nom du modèle utilisé lors de la dernière mise à jour de la table.

  Vous pouvez ainsi facilement vérifier que les mises à jour des tables sont faites régulièrement.
- Observer l'historique d'exécution de LDETLFB, où l'on trouve, jour par jour, une trace des différents scénarios et modèles exécutés, avec le cas échéant les messages d'erreur rencontrés. Pour chaque exécution, on trouve un message de début et fin d'exécution, le message de fin comportant notamment le nombre d'enregistrements ajoutés dans la table cible.

En tout état de cause, il convient de vérifier régulièrement la bonne marche de ces tâches planifiées. En effet, c'est une garantie de disposer de données exactes dans l'entrepôt de données. Si la ou les tâches de LDETLFB ne se sont pas exécutées, les utilisateurs de LDVision vont travailler avec des données qui ne seront pas à jour.



6 - Présentation du langage Windev

L'objet de ce chapitre est de présenter de façon très succincte la syntaxe du langage Windev, et les fonctions les plus utiles pour une mise en œuvre au sein de l'application LDETLFB.

Pour consulter l'aide en ligne, vous pouvez à tout moment appuyer sur la touche *F1*. Une fenêtre s'ouvre alors, vous proposant plusieurs choix, dont l'accès à l'aide du WLangage. Si l'aide Windev n'est pas installée sur le poste de travail, le système vous propose de la télécharger automatiquement.

Vous pouvez lancer l'installation à partir du lien ci-dessous :

ftp://framework.pcsoft.fr/HLP/12.0/FR/InstallHlpMCU.EXE

6.1 - Syntaxe générale du langage

Tout code Windev est constitué d'une suite d'instructions. En règle générale, on saisit une instruction par ligne, mais on peut aussi :

- saisir plusieurs instructions sur une même ligne, en les séparant par un point virgule « ; »
- saisir une instruction en plusieurs lignes ; la ligne doit pour cela se terminer par trois points accolés les uns aux autres « … »

On peut toujours commenter le code que l'on saisit, et cela est même vivement recommandé ; tout ce qui suit une séquence de deux caractères « // » est du commentaire, et sera donc ignoré par le système.

Pour référencer une valeur constante, indiquer directement la valeur s'il s'agit d'une valeur numérique ; placez cette valeur entre guillemets « " » s'il s'agit d'une chaine de caractères.

Le langage Windev est insensible à la casse ; vous pouvez donc indifféremment saisir votre code en majuscule ou en minuscule.

Enfin, les espaces, dès lors qu'ils se trouvent en dehors d'une valeur de type chaine de caractères (toujours frappée entre guillemets `"comme ceci"`) ne sont pas significatifs. Vous pouvez en placer à votre guise.

6.2 - Les variables

Si vous avez besoin d'une variable pour enregistrer des résultats intermédiaires de calcul, il faut la déclarer. Une variable se définit par son nom et son type. Le type de la variable détermine les valeurs que la variable peut prendre, et les opérations qui sont possibles. La déclaration d'une variable est toujours de la forme :

NomVariable est un(e) TypeVariable

ou

NomVariable1,NomVariable2 sont des TypeVariables

Remarque : les mots-clés *un*, *une* et *des* ne sont pas obligatoires ; ce sont des mots d'agrément. De même, le type de la variable peut être défini au singulier ou au pluriel ; l'orthographe n'est pas vérifiée par le système ; le respect des règles d'orthographe a simplement pour but de faciliter la relecture du code par une autre personne.

Quelques exemples :

```
Indice est un entier
NomSalarié est une chaine
Montant est un réel
```

```
I, J, K sont des entiers
```

Les principaux types sont :

- *Entier*, pour manipuler des valeurs entières uniquement
- *Réel*, pour manipuler des valeurs réelles (avec des décimales)
- *Chaine*, pour manipuler des chaines de caractères

On peut aussi déclarer et initialiser une variable en une seule instruction :

NomVariable est un(e) TypeVariable = Valeur

Exemple :

```
Compteur est un entier = 15
MoisPaye est une chaine = "200703"
```

6.3 - Les tableaux

Windev sait manipuler un grand nombre de tableaux. : simples, dynamiques, à une ou plusieurs dimensions… Contentons nous de la forme la plus simple :

NomTableau est un(e) tableau de Dimension TypeVariable

Exemple :

```
BrutMensuel est un tableau de 12 réels
```

Pour faire référence à un élément du tableau :

```
NomTableau[Indice]
```

Pour connaître le nombre d'élément d'un tableau :

```
Dimension(NomTableau)
```

Exemple (calcul de la somme des éléments d'un tableau):

```
BrutMensuel est un tableau de 12 réels
TotalAnnuel est un réel
i est un entier
Pour i = 1 à Dimension(BrutMensuel)
   TotalAnnuel = TotalAnnuel + BrutMensuel[i]
Fin
```

6.4 - Les opérateurs

Les opérateurs logiques

Les opérateurs logiques sont *ET*, *OU*, *PAS*. Ces opérateurs logiques permettent de construire des conditions.

Exemple :

```
// Tester qu'une valeur est dans un intervalle donné
SI BrutMensuel < 1000 ET BrutMensuel < 1500 ALORS
```

Les règles de priorité des opérateurs sont classiques ; dans le doute, vous pouvez utiliser des parenthèses pour forcer l'ordre d'évaluation des conditions :

```
SI (Brut < 1000 ET Brut < 1500) OU Catégorie="10" ALORS
```

Les opérateurs arithmétiques

En plus des 4 opérateurs classiques :

- \+ Addition
- \- Soustraction
- * Multiplication

/ Division

on trouve d'autres opérateurs forts pratiques :

++ Incrémentation : `I++` est équivalent à `I=I+1`
-- Décrémentation : `I--` est équivalent à `I=I-1`
+= Ajout d'une valeur `I+=5` est équivalent à `I=I+5`

Les règles de priorité des opérateurs sont là aussi classiques ; utilisez les parenthèses lorsque cela est nécessaire :

```
I = 5 ; J = 6
I = I + J * 3        // I vaut maintenant 23
I = (I + J) * 3      // I vaut maintenant 33
```

Les opérateurs de comparaison

Ce sont les opérateurs classiques :

= Egal
<> Différent
<= Inférieur ou égal
>= Supérieur ou égal
< Inférieur strictement
> Supérieur strictement

Ces opérateurs peuvent être utilisés tant avec des variables ou valeurs numériques qu'avec des chaines de caractères.

On dispose également d'opérateurs spécifiques aux chaines de caractères :

~= Egalité souple : ne tient pas compte ni des caractères majuscules ou minuscules,
ni des voyelles accentuées,
ni des espaces situés avant ou après la chaine de caractères à tester
[= Commence par

Exemples:

```
"Dupont" = "DUPONT"       // Retourne Faux
"Dupont" ~= "DUPONT"      // Retourne Vrai
```

Les instructions conditionnelles

L'instruction la plus utile est *SI, SINON, FIN*.

Quelques exemples :

```
SI Indice < 0 ALORS Indice=0    // La forme la plus simple

SI Indice < 0 ALORS             // Sur plusieurs lignes, avec Sinon
   Indice = 0
SINON
   Indice=Indice+1
FIN

SI Indice < 0 ALORS             // Plusieurs conditions en cascade
   Libellé = "Erreur"
SINON SI Indice = 0
   Libellé = "Pas encore traité"
SINON
   Libellé = "c'est OK"
FIN

SI I < 10 ALORS                 // Plusieurs conditions imbriquées
```

```
   SI J < 10 ALORS
         J = J+1
   SINON
         J=0 ; I ++
   FIN
SINON
   Libellé = "c'est fini"
FIN
```

Les instructions de boucle

Les instructions de boucle les utiles sont :

- *POUR FIN* pour un nombre d'itérations déterminé
- *TANTQUE FIN* pour réaliser un traitement tant qu'une condition n'est pas vérifiée.

Quelques exemples :

```
Indice est un entier
POUR Indice = 1 à 100
   // Traitement ici réalisé 100 fois
FIN

Indice est un entier
Indice=1
TANTQUE Indice <= 100
   Indice ++
   // Traitement ici réalisé aussi 100 fois
FIN
```

6.5 - Les fonctions de base sur les variables numériques, chaînes, et dates

Windev offre plusieurs centaines de fonctions intégrées prêtes à l'emploi. Nous allons en décrire ici seulement quelques unes, d'un usage très fréquent.

Manipulation de chaînes de caractères

Pour concaténer deux chaînes de caractères, utiliser l'opérateur +.

```
NomPrénom = PEPERS.NMAG + " " + PEPERS.PREN
```

Pour extraire une partie d'une chaine de caractères, plusieurs possibilités :

```
Chaine= "ABCDEF"
NouvelleChaine=Chaine[[2 à 3]]        // NouvelleChaine vaut "BC"
NouvelleChaine=Gauche(Chaine, 2)      // NouvelleChaine vaut "AB"
NouvelleChaine=Droite(Chaine, 2)      // NouvelleChaine vaut "EF"
NouvelleChaine=Milieu(Chaine, 2, 3)   // NouvelleChaine vaut "BCD"
```

La fonction *Taille* permet de connaître la taille d'une chaine de caractères.

```
Info( Taille(Chaine))     // Affiche 5
```

La fonction *Extraitchaine* permet d'extraire une partie d'une chaine composée d'une suite de plusieurs éléments séparés par un caractère bien identifié.

```
MaListe est une chaine = "01;02;003;004"
Chaine=ExtraitChaine(MaListe,3, ";")  // Chaine vaut "003"
```

Une chaine peut être transformée en majuscule ou en minuscule :

```
Chaine=Majuscule("Abcdef")      // Chaine vaut "ABCDEF"»
Chaine=Minuscule("Abcdef")      // Chaine vaut "abcdef"
```

Une chaine peut être recherchée dans une autre chaine avec la fonction *Position* :

```
MaChaine est une chaine = "ABCDEFGH"
Pos est un entier
```

```
Pos=Position(MaChaine,"DE",1)   // Pos vaut 5
// le 1er paramètre définit la chaine dans laquelle on recherche
// le 2ème paramètre définit la chaine recherchée
// le 3ème paramètre définit la position à partir de la quelle on recherche
// On peut également spécifier un 4ème paramètre SansCasse pour
//   que la recherche s'effectue sans tenir compte
//   des majuscules et minuscules
```

Une chaine peut être complétée à une longueur donnée :

```
MaChaine est une chaine = "ABCDEFGH"
MaChaine=Complete(MaChaine,10)  // MaChaine a été complété avec 2 espaces
à droite
```

Autres fonctions pouvant s'avérer utiles :

```
MaChaine=Répète("0", 10)  // Contient 10 fois le caractères 0
MaChaine=SansEspace(MaChaine)   // Elimine les espaces situés à gauche
                                //    et à droite
```

Manipulation des valeurs ou variables numériques

Utilisez un point décimal pour les valeurs comportant une partie décimale

```
Salaire est un réel
Salaire = 1345.56         // et non pas 1345,56
```

Tous les opérateurs mathématiques décrits plus haut peuvent être utilisés, avec les règles de priorité classiques. On peut utiliser des parenthèses pour forcer un ordre de calcul particulier :

```
Salaire est un réel = 1234.56
I est un entier = 2
Résultat est un réel
Résultat = Salaire * (I+1) / 10       // Résultat = 370.368
```

Il est possible de concaténer une chaîne et un numérique avec l'opérateur +. Le résultat est une chaîne de caractères :

```
Info("Le résultat est "+Résultat)     // Pour afficher le résultat
```

Autres fonctions utiles (qui se passent d'explications) :

```
R est un réel = -1234.56
P est un réel
P=Abs(R)                  // P vaut 1234.56
P=PartieEntière(P)        // P vaut 1234
P=PartieDécimale(R)       // P vaut -0.56
P=Arrondi(R,1)            // P vaut -1234,6 // 0.56 arrondi à 0.60
```

Pour transformer une valeur numérique en chaîne de caractères, utilisez la fonction *NumériqueVersChaine* :

```
R est un réel = 1234.56
Info(NumériqueVersChaine(R, "10,3f")) // Affiche 1234,560
Info(NumériqueVersChaine(R, "010.3f")) // Affiche 001234.560
I est un entier = 1234
Info(NumériqueVersChaine(I)) // Affiche 1234
Info(NumériqueVersChaine(I, "05d")) // Affiche 01234
```

A l'inverse, pour transformer une chaîne en valeur numérique, utilisez la fonction *Val* :

```
MaChaine est une chaine = "1234.56"
R est un réel
R=Val(MaChaine) * 2       // R vaut 2469.12
```

Manipulation des dates

En Windev, une date est enregistrée la plupart du temps sous forme d'une chaîne de caractères, au format *AAAAMMJJ*.

Pour convertir une date du format interne *AAAAMMJJ* au format traditionnel *JJ/MM/AAAA*, utiliser la fonction *DateVersChaine* :

```
MaDate est une chaine = "20070312"
Info(DateVersChaine(Madate, "JJ/MM/AAAA")) // Affiche 12/03/2007
```

La fonction *ChaineVersDate* a l'effet inverse.

Pour effectuer des calculs sur les dates, le mieux est de convertir la date en un entier.

```
MaDate est une chaine = "20070331"
D est un entier
D=DateVersEntier(Madate)
D++
Madate=EntierVersDate(D)
Info(DateVersChaine(Madate, "JJ/MM/AA")) // Affiche 01/04/07
```

On aurait pu aussi écrire de façon plus concise (mais peut être moins lisible) :

```
MaDate est une chaine = « 20070331 »
Info(DateVersChaine(EntierVersdate(DateVersEntier(Madate)+1),"JJ/MM/AA"))
// Affiche là aussi 01/04/07
```

Pour connaître la date du jour, utilisez la fonction *DateDuJour* ou *DateSys* :

```
MaDate est une chaine = DateDuJour()  // Date au format AAAAMMJJ
```

Notez dans l'exemple ci-dessus que tout appel à une fonction Windev doit être suivi de parenthèses juste après le nom, même si la fonction n'attend aucun paramètre.

Pour connaître le nombre de jours écoulés entre deux dates, utilisez la fonction *DateDifférence* :

```
MaDate est une chaine = "20070301"
Info(DateDifférence(Madate,DateSys()) // Affiche 27 si on est le 28/03/07
```

Autres fonctions utiles sur les dates :

```
MaDate est une chaine = "20070331"
Info(DateVersJour(madate))            // Donne 5 (1=Lundi, 2=mardi…)
Info(DateVersJourEnLettre(madate))    // Jeudi
Info(DateVersMoisEnLettre(madate))    // Mars
```

6.6 - Gestion des fichiers

Seules les fonctions de lecture de la base de données sont abordées ici, et seulement pour les fonctions les plus simples.

Pour accéder, dans une séquence de code, à une rubrique d'un fichier, la syntaxe est *NomFichier.NomRubrique*.

Cette syntaxe peut être utilisée dès lors qu'un enregistrement du fichier a été lu auparavant.

Pour lire un enregistrement, la fonction la plus courante est la fonction *HLitRecherchePremier*. Cette fonction attend 3 paramètres, le nom du fichier, le nom de la clé d'accès sur ce fichier, et la valeur de la clé recherchée. Lors de l'utilisation de cette fonction au sein d'une séquence de code dans LDETLFB, le nom du fichier et le nom de la clé doivent être stipulés entre guillemets doubles.

Exemple de lecture d'un enregistrement du plan comptable (fichier CPTPLA), dans une séquence de code de LDETLFB, correspondant au compte dont le N° est dans la colonne CPTG du fichier cource du modèle :

```
HLitRecherchePremier("CPTPLA", "CPTG", $CPTG)
```

Suite à l'appel de cette fonction, on peut utiliser la fonction *HTrouve* pour savoir si un enregistrement corrrespondant à la clé de recherche a été lu ou pas.

```
SI HTrouve() ALORS
   // Code exécuté si on a trouvé l'enregistrement recherché
SINON
   // Code exécuté si on n'a pas trouvé l'enregistrement recherché
```

```
FIN
```

On peut aussi tester directement la valeur retournée par la fonction de lecture, comme ceci :

```
SI HLitRechercheDernier("CPTPLA", "CPTG", $CPTG) ALORS
   // Code exécuté si on a trouvé l'enregistrement recherché
SINON
   // Code exécuté si on n'a pas trouvé l'enregistrement recherché
FIN
```

Dans le cas d'une clé composée, les valeurs correspondant aux différentes composantes de la clé seront portées entre crochets, séparées par une virgule :

```
HLitRechercheDernier("CPTPDI", "KPDI", ["P", $MOPM])
```

Les exemples ci-dessus présentent la recherche d'un enregistrement précis. Mais on peut aussi parcourir tous les enregistrements dont la clé est égale, ou commence par une valeur donnée. L'exemple ci-dessous permet de parcourir toutes les lignes de bulletins correspondant à un en-tête de bulletin donné (identifié ici par le code société *$COSO*, le N° matricule *$NPPE*, le mois de paye *$MPAY* et le N° du bulletin *$NOBU*) pour un N° de rubrique donnée (ici, la rubrique *5900*) :

```
Total est un réel = 0
HLitRechercheDernier("CALIBU","KLIBU5",[pCOSO,pNPPE,"5900",pMPAY,pNOBU],
hGenerique)
TANTQUE HTrouve()
   Total+=CALIBU.MONT
   HLitSuivant()
FIN
```

Notez l'utilisation du paramètre supplémentaire *hGenerique* qui permet de faire une recherche sur toutes les clés commençant par la valeur spécifiée.

Complément d'information :

- ⇨ La fonction *HLitRecherche* est équivalente à la fonction *HLitRechercheDernier* avec l'option *hGenerique.* A l'inverse, la fonction *HLitRechercheDernier* est équivalente à la fonction *HLitRecherche* avec l'option *hIdentique.*
- ⇨ Les fonctions *HTrouve* et *HlitSuivant* s'appliquent, lorsqu'on ne spécifie aucun paramètre, au dernier fichier accédé. Si on imbrique des codes accédant à plusieurs fichiers, il est préférable de spécifier le fichier concerné : *HTrouve("CALIBU")* et *HLitSuivant("CALIBU","KLIBU5")* dans l'exemple ci-dessus.

6.7 - Conclusion

Tout ce qui a été décrit dans ce chapitre ne représente, vous l'aurez compris, qu'une toute petite part de ce qu'il est possible de réaliser avec le langage de Windev.

Cette documentation n'a pas prétention à être exhaustive ; elle a juste pour but de vous montrer que l'écriture de code Windev est à la portée d'un grand nombre d'utilisateurs. Elle s'avère d'ailleurs souvent plus facile que l'écriture de macros Excel !