COGNOS®

Cognos Impromptu (R)

Création de fonctions personnalisées

Informations sur le produit

Le présent document s'applique à Impromptu (R) Version 7.1 et peut aussi s'appliquer aux versions ultérieures de ce produit. Pour vérifier s'il existe une version plus récente du document, visitez le site Web du support client de Cognos (http://support.cognos.com).

Copyright

Table des matières

Bienvenue

Contenu

Ce manuel présente en détail les informations nécessaires à la création de fonctions personnalisées pour votre produit de Cognos.

- Le chapitre 1, *Vue d'ensemble sur les fonctions personnalisées*, donne des informations générales relatives à la création de fonctions personnalisées, ainsi qu'aux systèmes d'exploitation reconnus. Ce chapitre rappelle également les procédures à suivre pour créer des fonctions personnalisées.
- Le chapitre 2, *Modification des fichiers .ini*, décrit les modifications à apporter aux fichiers .ini.
- Le chapitre 3, *Mise en œuvre de fonctions personnalisées de base de données*, décrit comment configurer les fonctions personnalisées selon vos systèmes de base de données.
- Le chapitre 4, *Mise en œuvre de fonctions personnalisées externes*, décrit comment configurer une fonction personnalisée dans une DLL (pour les environnements Windows) ou une bibliothèque partagée (pour les environnements UNIX).
- L'annexe contient des informations supplémentaires sur les conventions de nom de fichiers, les fichiers inclus dans le kit de développement logiciel (SDK ou « software development kit ») des fonctions personnalisées et l'utilitaire cogudf.exe qui vérifie la syntaxe des fichiers .sql.

Connaissances préalables

Pour créer des fonctions personnalisées, vous devez savoir :

- utiliser un éditeur de texte pour modifier les fichiers texte (.ini et .sql),
- ajouter des fonctions à votre base de données (pour les fonctions personnalisées de base de données),
- écrire, compiler et lier des programmes en langage C (pour les fonctions personnalisées externes).

Kit de développement logiciel

Vous devrez accéder au kit de développement logiciel (SDK) pour les fonctions personnalisées. Le kit de développement logiciel vous permet de créer des fonctions personnalisées de bases de données et des fonctions personnalisée externes pour Windows et UNIX.

Par défaut, le kit se trouve dans le dossier *emplacement_installation*\cer*n*\bin\UDF.

Le kit de développement logiciel contient les fichiers suivants :

- le présent manuel,
- les fichiers source C nécessaires à la mise en œuvre des fonctions personnalisées externes,
- un utilitaire permettant de vérifier rapidement la syntaxe de définition des fonctions personnalisées dans les fichiers .sql,
- des fichiers d'exemples et des fichiers texte Lisezmoi qui expliquent comment utiliser les fichiers d'exemples.

Pour en savoir davantage sur les fichiers contenus dans le kit de développement logiciel, reportez-vous à l'annexe à la (p. 39).

Les fichiers et les exemples du kit de développement logiciel ont été conçus à l'aide de Visual C++ 4.2 de Microsoft Developer Studio 97.

Conventions de ce manuel

- Les codes donnés en exemple apparaissent en police `Courier`.
- Pour éviter toute confusion entre les environnements Windows et UNIX, tous les noms de fichiers apparaissent en minuscules, à moins que l'utilisation des majuscules soit nécessaire.

Informations complémentaires

La documentation contient des guides d'utilisateurs, des guides didactiques, des manuels de référence et autres pour répondre aux besoins d'une multitude d'utilisateurs.

Toute l'information se trouve dans l'aide en ligne. L'aide en ligne est accessible à partir du bouton «?» des produits Windows.

Les renseignements du système d'aide en ligne sont accessibles en format .pdf. Toutefois, l'information d'un système d'aide donné peut être repartie à l'intérieur de plusieurs manuels. Il est possible d'imprimer les manuels en ligne et d'effectuer une recherche dans tout le document. Vous pouvez imprimer des pages précises, une section du manuel ou le tout. Cognos concède une licence non exclusive et incessible afin de permettre l'utilisation et la reproduction de la documentation protégée par les droits d'auteur, en format imprimé et électronique, et ce, pour les seules fins de fournir la formation interne et d'utiliser et de maintenir le logiciel de Cognos.

Pour les produits Windows, les manuels en ligne sont disponibles à partir du menu *Démarrer* de Windows (Cognos) et du menu *Aide* du produit (Manuels à imprimer). Tous les manuels en ligne se trouvent sur le CD de la documentation de Cognos. Vous pouvez aussi vous référez directement aux fichiers Lisezmoi et aux guides d'installation sur CD du produit de Cognos.

Seuls les guides d'installation sont disponibles en format imprimé.

Une liste annotée de la documentation complémentaire, l'*Introduction à la documentation*, est disponible à partir du menu *Démarrer* de Windows ou du menu *Aide* d'Impromptu.

Questions ou commentaires ?

Pour obtenir une réponse rapide aux questions concernant l'utilisation d'Impromptu, communiquez avec le Support client.

Pour en savoir davantage sur le support client (adresses et programmes), reportez-vous à la section relative au support client de l'aide en ligne ou visitez le site Web de support de Cognos (http://support.cognos.com).

Chapitre 1 : Vue d’ensemble sur les fonctions personnalisées

Ce chapitre aborde les thèmes suivants :

- informations générales sur les fonctions personnalisées (« user-defined functions » ou UDF),
- description des deux types de fonctions personnalisées et utilisation dans un environnement de création de rapports,
- informations sur les systèmes d'exploitation pris en charge,
- procédures pour créer des fonctions personnalisées.

Aperçu

Vous pouvez créer des fonctions pour :

- obtenir une fonction personnalisée ou de gestion qui n'est pas déjà fournie dans l'environnement de création de rapports de votre produit de Cognos,
- utiliser une fonction dans une base de données comme répertoire principal,
- accéder à des masques ou à d'autres applications qui codent et regroupent des données,
- travailler avec des types de données personnalisés trouvés dans des bases de données telles qu'Informix, DB2 et Oracle.

Types de fonctions personnalisées

Il existe deux types de fonctions personnalisées :

- les fonctions de base de données,

  Les fonctions personnalisées de bases de données sont définies de façon à pouvoir être utilisées avec un ou plusieurs systèmes de base de données. Reportez-vous à la documentation de votre SGBD relationnel pour de plus amples informations sur l'utilisation des fonctions personnalisées dans l'environnement de votre base de données.

  Remarque : Dans Architect, les fonctions de base de données sont appelées procédures stockées.
- les fonctions externes.

  Les fonctions personnalisées externes sont écrites en langage C ou dans un autre langage (la convention d'appel du langage C doit être reconnue) et sont compilées dans des DLL afin d'être utilisées sous Windows NT. Aux fins des environnements UNIX, les fonctions personnalisées externes sont compilées dans des bibliothèques partagées.

Les produits de Cognos prennent en charge les fonctions personnalisées scalaires qui renvoient une seule valeur chaque fois qu'elles sont appelées.

Recherche des noms de fonction par les produits de Cognos

Pour trouver le nom d'une fonction dans une instruction SQL, les produits de Cognos procèdent dans l'ordre indiqué ci-dessous.

1. Fonctions de bases de données (système et fonctions personnalisées) propres à une instance particulière d'un type de base de données.
2. Fonctions de base de données propres au type de base de données.
3. Fonctions personnalisées externes.

4. Fonctions incorporées dans votre produit de Cognos.

Important

- Vous devez faire une copie de sauvegarde de tous les fichiers .ini et .sql des fonctions personnalisées que vous modifiez lorsque vous travaillez avec des exemples ou lorsque vous créez vos propres fonctions personnalisées. En effet, ces fichiers seront remplacés si vous réinstallez votre produit de Cognos et vous perdrez toutes les modifications effectuées.
- Pour faciliter la migration de vos fonctions personnalisées vers des versions ultérieures du produit de Cognos, nommez les fonctions de façon à ce que ces noms risquent peu d'être attribués aux fonctions des versions futures du produit.

Systèmes d'exploitation pris en charge

Pour en savoir davantage sur les systèmes d'exploitation pris en charge et les versions, visitez le site Web de support de Cognos (http://support.cognos.com).

Procédures relatives aux fonctions personnalisées

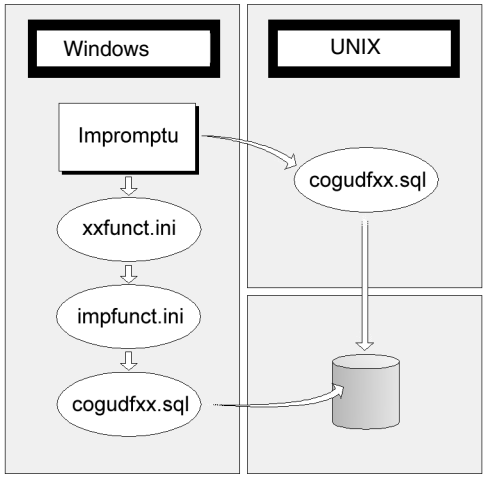
Utilisez les procédures décrites dans les sections suivantes pour mettre en œuvre les fonctions personnalisées :

- Mise en œuvre d'une fonction personnalisée de base de données
- Utilisation d'une fonction personnalisée de base de données avec plusieurs bases de données
- Mise en œuvre d'une fonction personnalisée externe

Vous trouverez en annexe une liste des abréviations des bases de données, composées de deux caractères, utilisées pour nommer les fichiers .ini et .sql qu'il est nécessaire de modifier dans ces procédures.

Mise en œuvre d'une fonction personnalisée de base de données

Le diagramme affiché ci-dessous présente les fichiers utilisés par les produits de Cognos pour accéder à une fonction personnalisée de base de données.



1. Ajoutez la fonction à la base de données si la fonction n'a pas déjà été créée. (Pour en savoir davantage, consultez la documentation du SGBD relationnel.)
2. Ajoutez le nom de la fonction au fichier .ini correspondant à la base de données. Par défaut, le fichier se trouve dans le dossier *emplacement_installation*\cer*n*\bin.

   Le format des noms de fichiers est xxfunct.ini (combinaison d'un identificateur à deux caractères pour la base de données et des lettres « funct »). Par exemple, le fichier .ini pour Oracle est orfunct.ini.

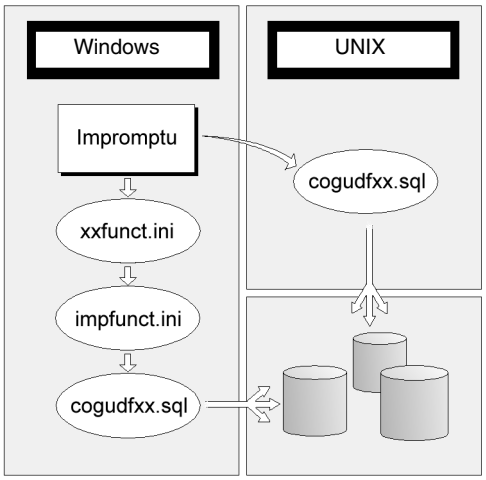
   Entrez le nom de la fonction dans la section [Database-specific Function List] du fichier .ini.
3. Ajoutez une section portant le nom de la fonction à la fin du fichier .ini. Le nom doit être entre crochets. Par exemple, [MaFonction].
4. Dans la nouvelle section, ajoutez la définition de la fonction.
5. Ajoutez la définition de la fonction au fichier .sql correspondant. Par défaut, le fichier se trouve dans le dossier *emplacement_installation*\cer*n*\bin. Le format des noms de fichiers est cogudfxx.sql (combinaison d'un identificateur à deux caractères pour la base de données et des lettres « cogudf »). S'il n'y a pas de fichier .sql pour la base de données dans le dossier Bin, créez-le.

   Vous pouvez vous servir de l'utilitaire de vérification de la syntaxe de Windows, cogudf.exe (fourni avec le kit de développement logiciel), pour vérifier ce que vous avez ajouté au fichier .sql. Pour en savoir davantage, reportez-vous à l'annexe à la (p. 39).

Si la fonction personnalisée a été correctement ajoutée à l'ensemble des fichiers, elle sera visible dans l'Éditeur d'expression lors du démarrage du produit de Cognos et de la connexion à une base de données qui utilise cette fonction.

Utilisation d'une fonction personnalisée de base de données avec plusieurs bases de données

Le diagramme affiché ci-dessous présente les fichiers utilisés par les produits de Cognos pour accéder à une fonction personnalisée de base de données créée pour plusieurs bases de données.



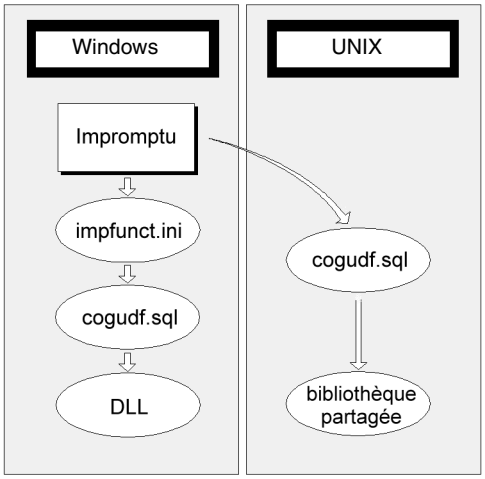
1. Ajoutez la fonction au fichier impfunct.ini dans la section [Common Database Function List].
2. Ajoutez une nouvelle section portant le nom de la fonction à la fin du fichier impfunct.ini. Le nom doit être entre crochets. Par exemple, [MaFonction].
3. Ajoutez la définition détaillée de la fonction sous le nom de la nouvelle section dans le fichier impfunct.ini.
4. Chaque fois que vous voulez utiliser cette fonction avec une base de données, ajoutez le nom de la fonction dans la section [Common Database Function List] du fichier .ini correspondant de la base de données (xxfunct.ini).
5. Pour chacune des bases de données avec laquelle sera utilisée cette fonction, ajoutez la définition de la fonction dans le fichier .sql correspondant (cogudfxx.sql), situé dans le dossier Bin. S'il n'existe aucun fichier .sql pour la base de données que vous voulez utiliser avec la fonction personnalisée, créez-le.

   L'utilitaire de vérification de la syntaxe de Windows, cogudf.exe (fourni avec le kit de développement logiciel), permet de vérifier ce que vous avez ajouté aux fichiers .sql. Pour en savoir davantage, reportez-vous à l'annexe à la (p. 39).

Si la fonction personnalisée a été correctement ajoutée à l'ensemble des fichiers, elle sera visible dans l'Éditeur d'expression lors du démarrage du produit de Cognos et de la connexion à une base de données qui utilise cette fonction.

Mise en œuvre d'une fonction personnalisée externe

Le diagramme affiché ci-dessous présente les fichiers utilisés par les produits de Cognos pour accéder à une fonction personnalisée externe.



1. Créez la DLL ou la bibliothèque partagée.
   La DLL doit être copiée vers l'un des dossiers suivants :
   - le dossier Bin.
   - l'un des répertoires dans la variable d'environnement PATH,
   - le dossier Windows,
   - le dossier système de Windows.

   Sous UNIX, utilisez l'une des variables d'environnement suivantes pour accéder à la bibliothèque partagée :
   - LD_LIBRARY_PATH (Solaris)
   - LIBPATH (AIX),
   - SHLIB_PATH (HP-UX).

   Pour en savoir davantage, reportez-vous à la section du manuel UNIX qui porte sur « ld ».
2. Ajoutez le nom de la fonction personnalisée externe dans la section [Built-in Function List] du fichier impfunct.ini.
3. Ajoutez une nouvelle section portant le nom de la fonction à la fin du fichier impfunct.ini. Le nom doit être entre crochets. Par exemple, [MaFonction].
4. Sous le nom de la nouvelle section, ajoutez la définition détaillée de la fonction.
5. Ajoutez la définition de la fonction au fichier cogudf.sql.
   L'utilitaire de vérification de la syntaxe de Windows, cogudf.exe (fourni avec le kit de développement logiciel), permet de vérifier ce que vous avez ajouté au fichier .sql. Avant de copier les fichiers .sql vers votre environnement UNIX, vous pouvez les vérifier à l'aide de cet utilitaire. Pour en savoir davantage, reportez-vous à l'annexe à la (p. 39).

Si la fonction personnalisée a été correctement ajoutée à l'ensemble des fichiers, elle sera visible dans l'Éditeur d'expression lors du démarrage de votre produit de Cognos.

Chapitre 2 : Modification des fichiers .ini

Ce chapitre aborde les thèmes suivants :

- le fichier impfunct.ini et la façon de le modifier,
- la structure des fichiers .ini spécifiques aux bases de données.

Aperçu

Tel que mentionné dans le chapitre précédent, il est nécessaire de modifier les fichiers .ini pour que votre produit de Cognos puisse utiliser vos fonctions personnalisées. Par défaut, les fichiers .ini se trouvent dans le dossier *emplacement_installation*\cer*n*\bin. Les fichiers à modifier sont les suivants :

- impfunct.ini (pour les fonctions personnalisées de bases de données qui seront utilisées avec plusieurs systèmes de base de données, ainsi que pour les fonctions personnalisées externes),
- un ou plusieurs fichiers .ini spécifiques (pour les fonctions personnalisées de bases de données seulement).

Pour en savoir davantage sur la liste des fichiers .ini spécifiques aux bases de données, consultez l'annexe à la (p. 39).

Contenu du fichier impfunct.ini

Le fichier impfunct.ini contient les définitions suivantes :

- fonctions de bases de données communes,
- fonctions personnalisées de base de données,
- fonctions personnalisées externes.

Ce fichier est divisé en plusieurs sections. Pour créer des fonctions personnalisées accessibles par plusieurs bases de données, modifiez la section [Common Database Function List].

Remarques

Si vous créez une fonction personnalisée de base de données, ne modifiez pas les entrées de la section [Built-in Function List]. Cependant, si vous créez une fonction personnalisée externe, ajoutez la nouvelle entrée à cette section.

Chaque nom de fonction se présente sur une seule ligne et est suivi du signe égal (=). Par exemple :

```
A_fonction=
B_fonction=
```

L'utilisation des majuscules et des minuscules n'a aucune importance pour les noms de fonctions : A_fonction équivaut à a_fonction.

Dans le fichier .ini, certaines sections portent le même nom que les fonctions auxquelles elles se réfèrent. Ces sections décrivent l'objet réel de ces fonctions, ainsi que la façon dont elles apparaissent dans l'Éditeur d'expression.

Propriétés de définition de fonctions

Chaque définition de fonction possède un certain nombre de propriétés. Chaque propriété se présente sur une seule ligne. Toutefois, vous remarquerez dans l'exemple ci-dessous relatif à la fonction *val-absolue*, la ligne commençant par *tip=* est affichée sur plusieurs lignes pour des raisons pratiques, mais elle devra apparaître sur une seule ligne dans le fichier .ini.

```
[absolute]
label=val-absolue
param=1
return=NM
1=NM;exp_numérique
exp=absolute ( 1^ )
tip=val_absolue (exp_numérique) \nDonne la
valeur absolue de exp_numérique. Une valeur
négative devient positive. Exemples : val-absolue (5)
donne 5 ; val-absolue (-5) donne 5.
tip1=Expression numérique
```

La propriété du libellé est facultative, mais les autres sont obligatoires. Chaque entrée de propriété est limitée à 4 096 caractères, excepté pour le paramètre tip. Le nombre maximal de caractères est 512.

Les propriétés de fonctions sont décrites ci-après.

Propriétés de fonctions

libellé

Facultatif. Nom de la fonction tel qu'il apparaît dans la section *Function List* de l'Éditeur d'expression. Le nom de la fonction apparaît si aucun libellé n'est spécifié.

param

Indique le nombre de paramètres de la fonction. Ce nombre doit être inférieur à 32 767.

return

Indique le type de valeur que donne la fonction. La valeur est représentée par l'un de ces codes à deux lettres :

NM Numérique

CH Caractère

DA Date

DT Date-heure

IN Intervalle

TM Heure

Pour les expressions complexes (celles dont les valeurs renvoyées diffèrent comme la fonction « decode » d'Oracle), la valeur est représentée par un seul caractère de A à Z. Elle comprend également une classe de paramètres (voir ci-dessous). Le type de la valeur renvoyée sera le même que le type de la classe de paramètres. Par exemple :

return=**B**

1=DA,DT,NM,CH,IN,TM:A;expression

2=DA,DT,NM,CH,IN,TM:A;recherche1

3=DA,DT,NM,CH,IN,TM:**B**;résultat1

4=DA,DT,NM,CH,IN,TM:A;recherche2

5=DA,DT,NM,CH,IN,TM:**B**;résultat2

6=DA,DT,NM,CH,IN,TM:**B**;défaut

Les paramètres de classe A (paramètres 1, 2 et 4) seront de type identique, de même que les paramètres de classe B (paramètres 3, 5 et 6). Dans cet exemple, le type de la valeur renvoyée sera le même que celui de la classe de paramètres B. La fonction « decode » d'Oracle (dont voici un extrait) peut être appelée comme dans les exemples suivants (NM? indiquant une valeur numérique, CH? une chaîne de caractères et DA? une valeur date) :

decode (NM1, NM2, CH3, NM4, CH5, NM6, CH7, CH8) renvoie une chaîne de caractères

decode (DA1, DA2, NM3) donne une valeur numérique

<description_paramètre>

Chaque paramètre est composé de différents types de valeurs possibles séparés les uns des autres par une virgule, suivis d'un point virgule (;) et d'un commentaire.

Il y a une seule propriété de description par paramètre. Ces propriétés ont le libellé 1, 2, 3 et ainsi de suite.

Les types sont :

NM Numérique

CH Caractère

DA Date

DT Date-heure

IN Intervalle

TM Heure

Certaines fonctions peuvent avoir des paramètres de types différents. Pour ceci, listez les différents types dans la description du nombre de paramètres en prenant soin de les séparer par une virgule. Par exemple, la fonction *ajout-jours()* accepte aussi bien une valeur date qu'une valeur date-heure comme premier paramètre.

1=DA,DT;exp_date

Pour les expressions complexes, la liste des différents types de chaque description de paramètre peut être suivie de deux points (:). Un seul caractère indique la classe de paramètres. Tous les paramètres d'une même classe doivent avoir la même liste de types dans leurs attributs de description de paramètres.

Tous les paramètres réels de la même classe doivent être de même type.

exp

La fonction « expression ». Le langage SQL de l'attribut « exp » doit être correct. Un ou plusieurs espaces ou caractères de tabulation séparent chaque unité syntaxique.

Les paramètres sont substitués dans l'instance SQL générée. Les marqueurs de paramètre sont remplacés par l'instance SQL générée pour chaque paramètre. Un marqueur de paramètre est un caractère caret (^) suivi d'un nombre. Par exemple, ^1.

Les expressions complexes peuvent être indiquées (pour avoir un exemple, reportez-vous à l'entrée « date-en-chaîne » du fichier impfunct.ini), mais la plupart des définitions se composent du nom de la fonction suivi de la liste des marqueurs de paramètres séparés par une virgule.

Pour référencer une fonction personnalisée externe, le nom de la fonction doit apparaître entre crochets ([…]) :

```
exp=[mafonct] ( ^1 , ^2 )
```

Les crochets sont également utilisés pour indiquer les séquences facultatives. De plus, les crochets suivis d'une astérisque (*) indiquent la répétition d'une séquence. Par exemple, la définition de la fonction « decode » comprend la définition d'expression suivante avec des séquences répétées et facultatives.

```
exp=decode ( ^1 , ^2 , ^3 [ , ^4 , ^5 ]* [ , ^6 ] )
```

[, ^4, ^5]* est répété mais [, ^6] ne ne l'est pas. Vous pouvez avoir une séquence répétée et une séquence facultative dans la même définition de fonction. Cependant, vous ne pouvez pas avoir plus d'une séquence de chaque dans la même définition de fonction. Si les deux types de séquence sont présents, la séquence répétée doit précéder la séquence facultative.

Pour une séquence répétée, les produits de Cognos traitent systématiquement tous les paramètres proposés.

Voici un autre exemple du fonctionnement d'une séquence répétée. Si vous définissez :

```
exp=FooFunc( ^1 [, ^2 , ^3]*)
```

Vous pouvez appeler la fonction comme suit :

```
FooFunc(A, B, C)
FooFunc(A, B, C, D, E)
FooFunc(A, B, C, D, E, F, G)
```

tip

Indique le contexte affiché pour la fonction dans l'Éditeur d'expression. Le nombre maximal de caractères de la chaîne est 512.

parameter tip

Indique le contexte des paramètres. Chaque paramètre doit avoir une fonction « tip ». Les fonctions tip des paramètres sont libellés tip1, tip2, tip3 et ainsi de suite.

type

Indique que la définition de la fonction dispose d'un paramètre répété, que la valeur renvoyée est dynamique, ou les deux. Sa valeur doit être une chaîne de caractères comprenant les caractères D ou R, ou DR :

- D : valeur de type dynamique (varie selon les types de paramètres),
- R : les paramètres peuvent être répétés,
- DR : les paramètres peuvent être répétés et la valeur renvoyée est de type dynamique.

fichiers .ini spécifiques aux bases de données

Le format des noms pour les fichiers .ini spécifiques aux bases de données est xxfunct.ini. Il s'agit de la combinaison d'un identificateur à deux caractères et des lettres « funct ». Reportez-vous à l'annexe pour consulter la liste des identificateurs à deux caractères.

Les fichiers .ini spécifiques aux bases de données sont traités lors de la connexion à la base de données.

[Built-in Function List]

Cette section a la même syntaxe que le fichier impfunct.ini. Si une fonction apparaît dans cette section, elle est utilisée dans la base de données comme langage SQL natif. Notez que la définition de la fonction personnalisée externe doit être ajoutée à la section [Built-in Function List] du fichier impfunct.ini, et non à celles des fichiers .ini spécifiques aux bases de données.

[Common Database Function List]

Cette section répertorie les fonctions de bases de données communes qui sont réellement utilisées par la base de données. Si une fonction répertoriée dans la section [Common Database Function List] du fichier impfunct.ini n'apparaît pas ici, elle n'est donc pas utilisée par la base de données et n'apparaîtra pas dans la liste des fonctions de l'Éditeur d'expression.

[Database-specific Function List]

Cette section répertorie les fonctions réellement utilisées par la base de données correspondante.

Définitions des fonctions individuelles

Le format de cette section est le même que celui du fichier impfunct.ini. Si vous souhaitez annuler les attributs de la description des paramètres et de exp dans le fichier impfunct.ini, redéfinissez-les dans cette section. Par exemple, vous voudrez peut-être annuler ces attributs pour transformer exp en une syntaxe spécifique d'une base de données ou afin de modifier les types de paramètres que la fonction utilise. Vous ne pouvez annuler aucun des autres attributs.

Les attributs des fonctions répertoriées dans la section [Database-specific Function List] doivent être indiqués ici.

Si une fonction est définie deux fois dans cette section, c'est la dernière définition qui est utilisée.

Chapitre 3 : Mise en œuvre de fonctions personnalisées de bases de données

Ce chapitre aborde les thèmes suivants :

- la modification des fichiers .sql appropriés,
- l'utilisation de la syntaxe de définition de fonctions personnalisées de bases de données,
- la définition des fonctions personnalisées de bases de données surchargées.

Fichiers SQL de bases de données (cogudfxx.sql)

Pour définir une fonction personnalisée de base de données, vous devez :

- ajouter la fonction à la base de données (s'il y a lieu),
  Pour en savoir davantage, reportez-vous à la documentation du SGBDR.
- modifier le fichier .ini pour indiquer le type de base de données,
- modifier le fichier impfunct.ini si la fonction personnalisée doit être utilisée avec plusieurs systèmes de bases de données,
- ajouter la définition de la fonction au fichier .sql correspondant au type de base de données.

Un fichier .sql de fonctions personnalisées de base de données contient des définitions de fonctions personnalisées et peut également contenir des commentaires. Pour créer des commentaires, utilisez deux tirets (- -). Tout texte se trouvant sur la même ligne que les tirets est ignoré.

Remarque : Les fichiers cogudfxx.sql ne peuvent pas contenir d'instructions DML (« Data Manipulation Language » ou langage de manipulation de données) SQL de Cognos, telles que SELECT ou INSERT.

Syntaxe de définition d'une fonction personnalisée de base de données

Les éléments indiqués en gras sont obligatoires. Les éléments indiqués entre crochets [ ] sont facultatifs. Les ensembles d'éléments entre accolades { } peuvent être présents. Le format et la casse des noms de la base de données doivent correspondre aux noms utilisés pour définir les fonctions personnalisées dans la base de données.

```
<déclaration_fonction_base_de_données> ::=
   DECLARE [ DATABASE] [ <type_fonction> ] FUNCTION
  <nom_fonction> [<liste_paramètres_formels>]
  RETURNS  <type_données>
  FUNCTION NAME  <nom_fonction_base_de_données>;

<type_fonction> ::=
      SCALAR

<nom_fonction> ::=
      [< nom_base_de_données_logique>.]<identificateur>

<liste_paramètres_formels> ::=
      ([<  type_données>] [{,<type_données>} … ] )

<renvoie_type_données> ::=
      <type de données>

<nom_fonction_base_de_données> ::=
    : [<catalogue>.][<schéma>.]<nom_fonction>

<nom_base_de_données_logique> ::=
  : <identificateur>

<catalogue> ::=
  : <identificateur>

<schéma> ::=
  : <identificateur>

<nom_fonction> ::=
  : <identificateur>

<identificateur> ::=
  : texte
  | "<texte>"

<type_données> ::=
     STRING
   | BOOLEAN
   | NUMBER
   | BINARY
   | DATE
   | TIME
   | TIMESTAMP
   | INTERVAL
   | BLOB
   | TEXT
   | <valeur littérale>

<valeur_littérale> ::=
  : '<texte>'
```

Description de la syntaxe

- Un paramètre peut être de type BINARY (binaire) mais la valeur de renvoi d'une fonction ne peut pas être BINARY.
- Le paramètre<nom_fonction> initial est le nom utilisé par votre produit de Cognos pour identifier la fonction personnalisée.
- Le type *STRING* (chaîne) représente des valeurs de caractères, soit fixes, soit de longueur variable.
- Le type *NUMBER* (nombre) représente les types suivants de valeurs numériques :
  - SMALLINT
  - INTEGER
  - DECIMAL
  - NUMERIC
  - REAL
  - FLOAT
  - DOUBLE PRECISION
- Pour indiquer qu'une fonction de base de données n'a pas de paramètres et ne requiert pas de parenthèses, il suffit de ne pas mettre la liste des paramètres entre parenthèses dans la définition de la fonction.
- Le paramètre <nom_fonction_base_de_données> est identique à celui utilisé par la base de données sous-jacente pour identifier la fonction. Ceci permet à des fonctions de bases de données définies dans un autre schéma ou catalogue d'être référencées en SQL à l'aide d'un nom simple et personnalisé. Ce nom est utilisé exactement tel que vous l'entrez ; si des caractères spéciaux ou noms délimités sont requis, le nom complet de la fonction doit être délimité par des guillemets doubles. Les guillemets doubles incorporés sont indiqués par deux guillemets doubles successifs.
- Un paramètre peut avoir une valeur fixe indiquée par une chaîne de texte délimitée par des guillemets simples. Les guillemets simples incorporés sont indiqués par deux guillemets simples successifs. Un paramètre comportant une valeur fixe spécifiée risque de ne pas pouvoir être combiné avec un autre type de données de paramètre.
- La valeur de renvoi d'une fonction de base de données peut ne pas être une valeur littérale.

Exemples de définitions de fonctions personnalisées de base de données

Les fichiers .sql contiennent des définitions telles que :

```
DECLARE FUNCTION substitute( STRING, STRING )
RETURNS STRING
FUNCTION NAME lion.tiger.subst;

DECLARE DATABASE FUNCTION localDB.timeOfDay()
RETURNS TIME
FUNCTION NAME sysfunc.time_of_day;

Declare Database Function "Ma Fonction Personnalisée"
( Number, 'Valeur fixe ici' )
Returns Number
Function Name "Mon catalogue"."Mon schéma"."Fonction personnalisée";
```

Fonctions de bases de données surchargées

Les produits de Cognos peuvent accéder à des fonctions personnalisées de base de données quelle que soit la façon dont elles sont mises en œuvre dans la base de données sous-jacente. Mais les fonctions doivent être accessibles à l'intérieur d'une instruction SQL.

Certaines bases de données, telles que DB2 et Informix, reconnaissent la surcharge de fonctions. La surcharge signifie que deux ou plusieurs fonctions portant le même nom peuvent exister. Les fonctions surchargées se différencient par le nombre et les types de paramètres que chaque fonction s'attribue.

Actuellement, les produits de Cognos ne prennent pas en charge la surcharge de fonctions. Chaque fonction doit posséder un nom unique. Pour pouvoir accéder à différentes versions d'une fonction surchargée, les produits de Cognos requièrent que chaque instance de la fonction soit identifiée de façon unique dans le fichier .sql correspondant de la base de données.

La syntaxe d'identification de la fonction demeure la même, que la fonction personnalisée de base de données soit écrite en langage C, en Java ou dans un langage procédural de base de données propriétaire tel qu'Oracle PL/SQL.

Exemple

La fonction MAFONC est définie deux fois dans DB2. La première définition est :

```
CREATE FUNCTION mafonc (SMALLINT)
    RETURNS SMALLINT
    EXTERNAL NAME mabib!mafonc
    LANGUAGE C …
```

La deuxième définition est :

```
CREATE FUNCTION mafonc (FLOAT, FLOAT)
    RETURNS FLOAT
    EXTERNAL NAME mabib!mafoncflot
    LANGUAGE C …
```

Pour utiliser ces fonctions personnalisées depuis une instruction SQL de Cognos, vous devez définir les deux fonctions comme indiqué ci-dessous dans le fichier cogudfd2.sql :

```
DECLARE DATABASE FUNCTION maFoncEnt (NUMBER)
RETURNS NUMBER
FUNCTION NAME mafonc;

DECLARE DATABASE FUNCTION maFoncvirgflot (NUMBER, NUMBER)
RETURNS NUMBER
FUNCTION NAME mafonc;
```

Fonctions d'Oracle

L'exemple ci-dessous est en langage C mais il est possible d'utiliser le langage PL/SQL pour définir une fonction personnalisée dans une base de données Oracle. Pour en savoir davantage, reportez-vous à la documentation d'Oracle.

Exemple

```
CREATE FUNCTION obtain_city
( longitude IN FLOAT,
  latitude IN FLOAT )
RETURN FLOAT AS EXTERNAL
LIBRARY mabibliothèque
NAME "obtain"
LANGUAGE C;
```

La définition de cette fonction dans le fichier cogudfor.sql est :

```
DECLARE DATABASE FUNCTION obtainCity (NUMBER, NUMBER)
RETURNS STRING
FUNCTION NAME obtain_city;
```

Fonctions DB2

Comme indiqué ci-dessus, DB2 prend en charge la surcharge de fonctions. Un nom unique doit être utilisé pour identifier chaque version de la fonction surchargée dans votre produit de Cognos.

Exemple

```
CREATE FUNCTION obtain_city (FLOAT, FLOAT)
RETURNS VARCHAR(50)
EXTERNAL NAME "mabibliothèque!obtain"
LANGUAGE C
PARAMETER STYLE DB2SQL
DETERMINISTIC
NO SQL
NO EXTERNAL ACTION
```

La présentation de la fonction dans le fichier cogudfd2.sql est la même que pour l'exemple d'Oracle.

Fonctions d'Informix

Les produits de Cognos ne prennent pas en charge les variables locales d'instructions d'Informix (SLV ou « statement local variables »). Une variable locale d'instruction est une variable se trouvant dans une procédure enregistrée à laquelle vous pouvez faire référence dans l'instruction SQL d'appel. Pour de plus amples informations sur les SLV, reportez-vous à la documentation d'Informix.

Informix prend en charge la surcharge de fonctions. Ici encore, un nom unique doit être utilisé pour identifier chaque version de la fonction surchargée dans votre produit de Cognos.

Les fonctions personnalisées Universal Server d'Informix peuvent être écrites hors de la base de données. Les fonctions personnalisées pour Informix Online sont écrites sous forme de procédures. L'exemple suivant est pour Informix Universal Server 9.12.

Exemple

```
CREATE FUNCTION obtain_city (longitude FLOAT, latitude FLOAT)
RETURNING VARCHAR(50)
EXTERNAL NAME "/usr/lib/local/obtain.so"
LANGUAGE C
END FUNCTION;
```

La déclaration de la fonction dans le fichier cogudfif.sql est la même que pour l'exemple Oracle.

Chapitre 4 : Mise en œuvre de fonctions personnalisées externes

Ce chapitre aborde les thèmes suivants :

- l'ajout de fonctions personnalisées externes au fichier cogudf.sql,
- le codage de fonctions personnalisées externes,
- les exigences et les points à garder à l'esprit lors de la conception d'une bibliothèque de fonctions personnalisées externes.

Aperçu

Pour mettre en œuvre une fonction personnalisée externe, créez et déployez une DLL ou bibliothèque partagée, puis :

- ajoutez les définitions des fonctions personnalisées externes au fichier cogudf.sql,
- ajoutez la définition et le nom de la fonction au fichier impfunct.ini.

Ce chapitre décrit comment ajouter une entrée au fichier cogudf.sql et comment concevoir une bibliothèque de fonctions personnalisées externes.

Exigences et restrictions

Les exigences et restrictions suivantes s'appliquent aux fonctions personnalisées externes.

- Il est possible d'utiliser un autre langage que le langage C pour construire une fonction personnalisée externe mais cette fonction doit nécessairement prendre en charge la convention d'appel C.
- Le nom attribué à une fonction personnalisée externe doit être unique.
- Une fonction personnalisée externe peut utiliser 16 paramètres au maximum.
- Les paramètres ne peuvent pas comporter plusieurs types de données.
- Le mode binaire, le texte et les textes BLOB ne sont pas pris en charge en tant que paramètres de fonctions personnalisées ou de valeurs renvoyées de ces fonctions.
- Toutes les données élémentaires fournies et renvoyées par une fonction personnalisée externe sont alignées en fonction du type de données.
- Toutes les fonctions personnalisées externes doivent renvoyer une valeur vide.

Fichier SQL des fonctions personnalisées externes (cogudf.sql)

Par défaut, le fichier cogudf.sql se trouve dans le dossier *emplacement_installation*\cer*n*\bin. Les produits de Cognos lisent ce fichier lorsqu'ils détectent une fonction qu'ils ne reconnaissent pas comme une fonction de base de données. Le fichier cogudf.sql indique à votre produit de Cognos :

- où trouver la fonction personnalisée,
- le nom de la fonction personnalisée en langage C,
- le type de chaque paramètre,
- le type de renvoi de la fonction personnalisée.

Chemins de recherche pour les fichiers .sql

Dans les environnements Windows, les fichiers .sql sont recherchés dans :

1. le dossier de travail courant,
2. le dossier spécifié dans la section `servicesL` du fichier Cognos.ini,

3. le dossier Bin.

Dans les environnements UNIX, les fichiers .sql sont recherchés dans :

1. le répertoire de travail courant,
2. le répertoire indiqué par la variable d'environnement `COGUDFSQL`,
3. le répertoire indiqué par la variable d'environnement `DMDBIMI`.

Contenu

Le fichier cogudf.sql contient des instructions en DDL (« Data Definition Language » ou langage de définition des donnée) SQL de Cognos. Il ne peut pas contenir d'instructions DML (« Data Manipulation Language ») SQL de Cognos, telles que SELECT ou INSERT. Il ne peut pas non plus contenir de définitions de fonctions personnalisées de bases de données.

Tout comme les autres fichiers .sql, cogudf.sql peut également contenir des commentaires. Pour créer des commentaires, utilisez deux tirets consécutifs (- -).

Syntaxe d'une fonction personnalisée externe

Dans la syntaxe ci-dessous, les éléments indiqués entre crochets ([ ]) sont facultatifs. Il peut y avoir zéro ou plusieurs occurrences d'éléments entre accolades { }.

```
<déclaration_fonction_externe> ::=
  DECLARE EXTERNAL  [ <type_fonction> ]
  FUNCTION  <nom_fonction> (<liste_paramètres_formels>)
  RETURNS <type_données> [ <distribution_résultats> ]
  FUNCTION NAME  <nom_fonction_base_de_données> ;

<type_fonction> ::=
      SCALAR

<nom_fonction> ::=
      <identificateur>

<liste_paramètres_formels> ::=
      [<type_données>] [ { , <type_données> } … ]

<distribution_résultats> ::=
      CAST AS  <type_données>

<nom_fonction_externe> ::=
      [ <nom_module> . ] <nom_fonction_externe>

<nom_module> ::=
      <texte>

<nom_fonction_externe> ::=
      <texte>

<identificateur> ::=
  : <texte>
  | "texte"
<type_données> ::=
      CHARACTER  <longueur_optionnelle>
    | CHAR <longueur_optionnelle>
    | BINARY <longueur_optionnelle>
    | SMALLINT  <échelle_optionnelle>
    | INTEGER   <échelle_optionnelle>
    | QUADWORD   <échelle_optionnelle>
    | DECIMAL <précision_et_échelle>
    | NUMERIC  <précision_et_échelle>
    | REAL
    | DOUBLE PRECISION
    | DATE
    | TIME
    | TIMESTAMP
    | INTERVAL

<longueur_optionnelle> ::=
   ( <longueur> )

<échelle_optionnelle> ::=
   ( <échelle> )

<précision_et_échelle> ::=
   ( <précision> [, <échelle> ] )

<longueur> ::=
   <entier_absolu>

<précision> ::=
   <entier_absolu>

<échelle> ::=
   <entier_relatif>
```

Description de la syntaxe

- <nom_fonction> est le nom grâce auquel votre produit de Cognos identifie la fonction externe. Comme indiqué, les noms des fonctions doivent être uniques. Si deux fonctions portent le même nom, seule la première fonction est reconnue.
- CHARACTER et CHAR sont des synonymes.

- La taille d'un paramètre de type *CHARACTER* ne doit pas être spécifiée. La taille de tous les paramètres de caractères est déterminée par la valeur d'entrée attribuée à la fonction avant son exécution. Si le type de renvoi d'une fonction est *CHARACTER*, une longueur doit être spécifiée. Il s'agit de la longueur maximale pouvant être gérée par la fonction personnalisée externe.
- Un paramètre peut être de type BINARY (binaire) mais la valeur de distribution ou de renvoi d'une fonction ne peut pas être BINARY.
- Le <nom_module> du <nom_fonction_externe> est le nom de la bibliothèque de liaison dynamique (DLL) ou de la bibliothèque partagée contenant la fonction. Le <nom _fonction> du <nom_fonction_externe> est le nom de la fonction à l'intérieur de la bibliothèque. Par exemple, maDLL.maFonction.
- L'échelle implicite des valeurs SMALLINT, INTEGER et QUADWORD est zéro. Une échelle de valeurs supérieure à zéro indique le nombre de chiffres qui doivent apparaître après la virgule décimale. Une échelle de valeurs inférieure à zéro représente le nombre de chiffres supplémentaires qui doivent apparaître avant la virgule décimale. Par exemple, si une valeur INTEGER de 123 a une échelle de 2, sa valeur réelle est 1,23. Si l'échelle est -2, sa valeur réelle est 12 300.
- L'échelle implicite des valeurs *DECIMAL* et *NUMERIC* est zéro. Les valeurs d'échelle de *DECIMAL* et *NUMERIC* sont zéro ou supérieures à zéro.
- Les noms des fonctions de Cognos ne tiennent pas compte de la distinction entre majuscules et minuscules. Les noms des fonctions externes tiennent compte de la distinction entre majuscules et minuscules.

Exemples de définitions de fonctions personnalisées externes

Le fichier .sql contient des définitions telles que :

```
DECLARE EXTERNAL FUNCTION substitute( CHAR, CHAR )
SCALAR RETURNS CHAR(10) CAST AS CHAR(50)
FUNCTION NAME rxxfunc.subst;

DECLARE EXTERNAL FUNCTION timeOfDay()
RETURNS TIME
FUNCTION NAME sysfunc.time_of_day;
```

En langage C, ces deux fonctions se présentent comme suit :

```
void subst( CogChar *, CogChar *, CogChar ** )
{    /* Code ici. */
    return;

} /* subst */

void time_of_day( CogTime ** )
{    /* Code ici. */
    return;

} /* time_of_day */
```

Les exemples ci-dessus supposent que la bibliothèque peut être trouvée par le programme de chargement dynamique sans l'indication du chemin d'accès complet. Le nom de la DLL ou de la bibliothèque partagée peut être un nom simple ou peut contenir un chemin ou une extension de fichier. Lorsque le nom contient un chemin ou une extension de fichier, le nom de la bibliothèque doit être indiqué entre guillemets. Par exemple, l'instruction DECLARE FUNCTION ci-dessous est une instruction correcte :

```
DECLARE EXTERNAL FUNCTION maFonction( INTEGER )
RETURNS INTEGER
FUNCTION NAME "/usr/local/lib/mylib.so.1".ma_fonction;
```

Création de la bibliothèque de fonctions personnalisées externes

Considérations pour les environnements Windows

Toutes les fonctions personnalisées doivent être exportées de la DLL dans laquelle elles se trouvent afin d'être accessibles dans les produits de Cognos. Votre produit de Cognos peut accéder simultanément à plusieurs DLL qui contiennent des fonctions personnalisées.

Les DLL contenant des fonctions personnalisées doivent être situées dans :

- le répertoire courant de l'application à exécuter,
- l'un des répertoires indiqués dans la variable d'environnement PATH,
- le répertoire Windows,
- le répertoire système de Windows.

Toutes les DLL doivent être en 32 bits et compilées avec un alignement de 8 octets.

Considérations pour les environnements UNIX

Il existe des fonctions personnalisées dans de nombreuses bibliothèques partagées. La bibliothèque partagée de fonctions personnalisées doit être située dans un répertoire accessible à l'éditeur de liens d'exécution. Vous pouvez accéder à une bibliothèque partagée de fonctions personnalisées à l'aide des variables d'environnement UNIX déjà mentionnées :

- LD_LIBRARY_PATH (Solaris)
- LIBPATH (AIX),
- SHLIB_PATH (HP-UX).

Fichier cogudf.h

Le fichier cogudf.h doit être inclus à tout fichier contenant des fonctions personnalisées. Ce fichier permet d'accéder à toutes les macros nécessaires, aux définitions de types et de fonctions. Le fichier cogudf.h contient cogudfty.h. Vous n'avez donc pas besoin de l'inclure de façon explicite.

Si une bibliothèque de fonctions personnalisées utilise certaines fonctions définies dans le fichier cogudf.h, cette bibliothèque doit être liée à la bibliothèque de fonctions personnalisées (udflib.lib sous Windows, udfLib.a sous UNIX).

Une fonction personnalisée externe peut utiliser les fonctions de nombreuses bibliothèques (statiques ou partagées) ou DLL. Cependant, une fonction personnalisée doit libérer toute la mémoire allouée avant de terminer sa tâche.

Vérification du paramètre NULL

Tous les paramètres d'une fonction personnalisée sont transmis par référence à l'interrogation. Une valeur NULL pour un paramètre particulier indique que la valeur associée dans l'interrogation SQL partiellement accomplie est elle-même NULL.

La suppression de la référence à l'interrogation d'un paramètre NULL entraîne l'arrêt anormal de votre produit de Cognos. Par conséquent, les fonctions personnalisées doivent vérifier si des valeurs de paramètres sont égales à NULL avant de poursuivre le traitement.

Si la fonction personnalisée rencontre une valeur de paramètre NULL, elle doit renvoyer une valeur NULL.

Le code suivant indique comment vérifier les valeurs NULL :

```
Void maNouvFonction( CogInt32 *  param1,
                     CogInt32 *  param2,
                     CogInt32 ** résultat );
{
    if( param1 == NULL || param2 == NULL )
    {
        *résultat = NULL;
    }
    else
    {
        **résultat = *param1 - *param2;
    }

    return;

} /* maNouvFonction */
```

La valeur renvoyée de toutes les fonctions personnalisées est le paramètre final de la fonction. L'accès se fait toujours par une indirection unique pour définir sa valeur sur NULL et par une indirection double pour lui attribuer une valeur (comme indiqué ci-dessus).

Une fonction personnalisée peut accepter jusqu'à 16 paramètres d'entrée, sans compter le paramètre supplémentaire de la valeur de renvoi.

Gestion des erreurs

Il n'existe actuellement pas de système de gestion des erreurs pour les fonctions personnalisées. Si une erreur survient, une fonction personnalisée doit pouvoir se rétablir et renvoyer une valeur correcte.

Types de données

L'ensemble de types de données fixe suivant est disponible pour les fonctions personnalisées.

Type de données	Description
CogChar *	Pointeur vers une chaîne de caractères terminée par NULL.
CogInt16	Entier 16 bits relatif.
CogInt32	Entier 32 bits relatif.
CogInt64	Entier 64 bits relatif.
CogFloat32	Valeur décimale à virgule flottante 32 bits.
CogFloat64	Valeur décimale à virgule flottante 64 bits.
Décimal condensé (CogUInt8)	Longueur fixée, valeur numérique exacte (décrite ci-dessous).
CogDate	Représentation binaire d'une valeur année/mois/jour.
CogTime	Représentation binaire d'une valeur heure:minute:seconde:milliseconde.
CogTimestamp	Représentation binaire d'une valeur date/heure.

Type de données	Description
CogInterval	Représentation binaire d'une valeur heure:minute:seconde:milliseconde d'un jour.
CogBinary	Donnée binaire de longueur fixe.

Ces types de données (à l'exception de décimal condensé) sont définies en langage C dans l'en-tête du fichier cogudfty.h.

Les types de données CogChar, CogInt16, CogInt32, CogFloat32 et CogFloat64 sont représentés par des types de données en langage C sous-jacents. À l'exception de CogInt64, les types de données sont représentés sous forme de valeurs propriétaires. Le type de données CogInt64 est pris en charge sous forme de type de données entier 64 bits natif dans tous les systèmes d'exploitation qui sont compatibles avec les fonctions personnalisées.

Types CogInt64 natifs	
Windows	LONGLONG
Solaris	long long
HP-UX	long long
AIX	long long

Chaque plate-forme UNIX requiert une option de compilateur spécifique pour activer la prise en charge des entiers 64 bits. Consultez la documentation UNIX pour de plus amples informations.

Type de données décimales condensées

Les valeurs décimales condensées sont représentées à l'aide d'un quartet unique (4 bits) pour chaque chiffre d'un nombre. Toutes les valeurs sont complétées par des zéros de début et de fin afin d'assurer qu'un nombre fixe de chiffres est toujours représenté par une valeur décimale condensée. Pour indiquer si le quartet final (le plus à droite) d'une décimale condensée est positif ou négatif, une des valeurs suivantes lui est attribuée :

- `COG_DECIMAL_POSITIVE_1`,
- `COG_DECIMAL_POSITIVE_2`,
- `COG_DECIMAL_NEGATIVE`.

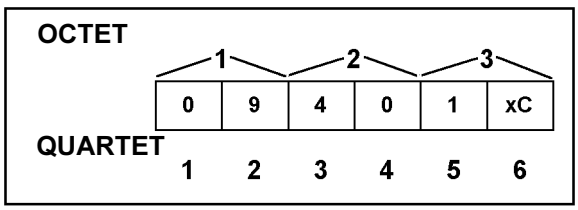
Ces valeurs sont définies dans le fichier cogudfty.h. Gardez à l'esprit qu'il est plus simple pour vos fonctions personnalisées de vérifier l'indicateur négatif que les indicateurs positifs.

La précision maximale d'une valeur décimale condensée est 77. L'échelle d'une valeur décimale condensée doit être supérieure ou égale à zéro et inférieure ou égale à la précision.

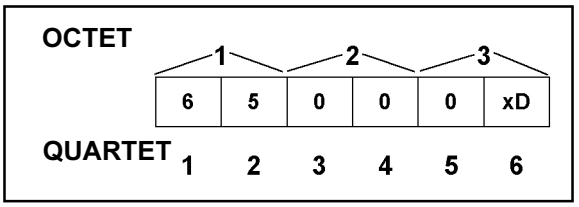
Le nombre d'octets requis pour représenter une valeur numérique dépend de la précision du type de données et si cette précision est paire ou impaire. Le calcul du nombre d'octets requis pour représenter une valeur décimale condensée est :

```
nombre d'octets = ( nombre de chiffres / 2 ) + 1
```

Par exemple, si un paramètre est défini avec une précision de 4 et une échelle de 2, la valeur 94.01 est représentée comme suit :



Si un paramètre est défini avec une précision de 5 et une échelle de 3, la valeur 65 est représentée comme suit :



Dans l'exemple ci-dessus, notez que la valeur -650 ne peut pas être représentée dans un paramètre décimal avec une précision de 5 et une échelle de 3. Lorsque vous définissez des fonctions personnalisées afin qu'elles acceptent ou renvoient des valeurs (numériques) décimales, assurez-vous que la fonction ne recevra que des valeurs acceptables ou que la gamme des valeurs acceptées est importante (grande précision). Si l'application ne nécessite pas plus de 19 chiffres de précision, vous pouvez également utiliser les valeurs CogInt64.

Les fonctions personnalisées manipulent les valeurs décimales condensées sous forme de tableau d'octets sans signe, dont le nombre est déterminé par le nombre de chiffres de la valeur décimale condensée (comme décrit ci-dessus).

Les deux fonctions suivantes sont fournies avec le logiciel de développement des fonctions personnalisées afin de convertir les valeurs décimales condensées en des chaînes de caractères ou depuis des chaînes de caractères. Les chaînes de caractères doivent être allouées assez longtemps avant l'appel de la fonction cogDecimalToString pour conserver la valeur obtenue.

```
CogChar * cogDecimalToString(
    CogUInt8 * pDecimal,
    CogChar  * pString
    CogInt32   précision,
    CogInt32   échelle,
);

CogUInt8 * cogStringToDecimal(
    CogChar  * pString,
    CogUInt8 * pDecimal
    CogInt32   précision,
    CogInt32   échelle
```

);

Vous trouverez ci-dessous la définition d'une fonction personnalisée acceptant une valeur décimale condensée en tant que paramètre d'entrée et renvoyant une valeur décimale à virgule flottante 32 bits :

```
void maFonctionDC( CogUInt8 * param1, CogFloat32 ** résultat )
{
    /*
     * La précision et l'échelle sont prédéfinies pour une fonction personnalisée.  Pour
     * cet exemple, supposons que la précision est de cinq et l'échelle
     * de deux.
     */

     if( param1 == NULL )
     {
        *résultat = NULL;
     }
     else
     {
        CogInt16   i;

        for( i = 0, **résultat = 0.0; i < 5; i++ )
        {
            **résultat *= 10;

            if( i & 1 )
            {
                **résultat += ( param1[ i / 2 ] ) & 0xF;
            }
            else
            {
                **résultat += ( param1[ i / 2 ] >> 4 ) & 0xF;
            }
        }

        /*
         * Prise en compte de l'échelle.
         */

        **résultat /= 100;

        if( ( param1[ 2 ] & 0x0F ) == COG_DECIMAL_NEGATIVE ){
            **résultat = -**résultat;
        }
     }

     return;

} /* maFonctionDC */
```

Type de données de date

Les valeurs de dates sont représentées par la structure CogDate. Les fonctions suivantes sont fournies en tant que partie du logiciel de développement des fonctions personnalisées afin de manipuler des valeurs de dates :

```
CogDate * cogConstructDate(
    CogDate * pCogDate,
    Int32     année,
    Int32     mois,
    Int32     jour
);
CogInt32 cogGetDatePart(
    CogDate * pCogDate,
    UInt32    datePart
);
```

Les valeurs possibles pour le paramètre datePart sont :

- COG_DATE_YEAR,
- COG_DATE_MONTH,
- COG_DATE_DAY.

L'exemple suivant indique une fonction personnalisée qui ajoute un an à la date fournie :

```
void maFonctionDate( CogDate * param1, CogDate ** résultat )
{
    CogInt32 année;
    CogInt32 mois;
    CogInt32 jour;

    if( param1 == NULL )
    {
        *résultat = NULL;
    }
    else
    {
        année  = cogGetDatePart( param1, COG_DATE_YEAR );
        mois = cogGetDatePart( param1, COG_DATE_MONTH );
        jour   = cogGetDatePart( param1, COG_DATE_DAY );

        année++;

        cogConstructDate( *résultat, année, mois, jour );
    }

    return;

} /* maFonctionDate */
```

Type de données d'heure

Les valeurs d'heures sont représentées par la structure CogUdfTime. Les fonctions suivantes sont fournies en tant que partie du logiciel de développement des fonctions personnalisées afin de manipuler des valeurs d'heures :

```
CogTime * cogConstructTime(
    CogTime * pCogTime,
    Int32         heure,
    Int32     minutes,
    Int32         secondes,
    In32          millisecondes
);
CogInt32 cogGetTimePart(
    CogTime * pCogTime,
    UInt32    timePart
);
```

Les valeurs possibles pour le paramètre timePart sont :

- COG_TIME_HOUR,
- COG_TIME_MINUTE,
- COG_TIME_SECOND,
- COG_TIME_MILLISECOND.

L'exemple ci-dessous indique une fonction personnalisée qui arrondit une valeur d'heure à la minute la plus proche :

```
void maFonctionHeure( CogTime * param1, CogTime ** résultat )
{
    CogInt32 heure;
    CogInt32 minute;
    CogInt32 secondes;
    CogInt32 fsecondes;

    if( param1 == NULL )
    {
        *résultat = NULL;
    }
    else
    {
        heure     = cogGetTimePart( param1, COG_TIME_HOUR );
        minute    = cogGetTimePart( param1, COG_TIME_MINUTE );
        secondes  = cogGetTimePart( param1, COG_TIME_SECOND );
        fsecondes = cogGetTimePart( param1,
                                    COG_TIME_MILLISECOND );

        /*
         * Arrondit vers le haut ou le bas à la minute la plus proche.
         */

        if( fsecondes >= 500 )
        {
            secondes++;
        }

        if( secondes >= 30 )
        {
            minute++;
        }

        if( minute == 60 )
        {
            minute = 0;
            heure++;
        }

        if( heure == 24 )
        {
            heures = 0;
        }

        fsecondes = 0;
        secondes  = 0;

        cogConstructTime( *résultat,
                          heure,
                          minute,
                          secondes,
                          fsecondes );
    }

    return;

} /* maFonctionHeure */
```

Type de données d'horodatage

Les valeurs d'horodatage sont représentées par la structure CogTimestamp.

```
typedef struct {
    CogDate date;
    CogTime heure;
} CogTimestamp;
```

Les différentes fonctions CogDate et CogTime peuvent être utilisées pour construire les champs d'une structure CogTimestamp ou pour en extraire des parties.

Type de données d'intervalle

Les valeurs d'intervalles sont représentées par la structure CogInterval. Les fonctions suivantes sont fournies en tant que partie du logiciel de développement des fonctions personnalisées afin de manipuler des valeurs d'intervalles :

```
CogInterval * cogConstructInterval(
    CogInterval * pCogInterval,
    Int32         jours,
    Int32         heure,
    Int32     minutes,
    Int32         secondes,
    In32          millisecondes
);
CogInt32 cogGetIntervalPart(
    CogInterval * pCogInterval,
    UInt32        intervalPart
);
```

Les valeurs possibles pour le paramètre intervalPart sont :

- COG_INTERVAL_DAYS,
- COG_INTERVAL_HOUR,
- COG_INTERVAL_MINUTE,
- COG_INTERVAL_SECOND,
- COG_INTERVAL_MILLISECOND.

Un intervalle indique un nombre de jours et une valeur d'heure. Un intervalle n'indique pas une date ou une heure spécifique mais un intervalle de temps. Par exemple, un intervalle mesure le temps entre deux valeurs d'horodatage spécifiques.

Type de données binaires

Les valeurs binaires sont représentées en langage C avec un descripteur :

```
typedef struct
{
    Int32      longueur;
    CogUInt8 * données;
}
CogBinary;
```

Le champ de la longueur représente le nombre d'octets de données binaires indiqué par le champ des données.

Une fonction qui renvoie une valeur binaire a une taille maximale prédéfinie, indiquée dans le fichier de définition des fonctions personnalisées. Une fonction personnalisée doit attribuer au champ de longueur le nombre d'octets qui contiennent des données correctes (si la valeur n'est pas égale à NULL).

Points essentiels sur les types de données

Lors de la conception d'une fonction personnalisée, vous devez prendre en considération l'environnement dans lequel cette fonction sera utilisée. Voici les questions importantes à vous poser :

- Si la fonction personnalisée renvoie une chaîne de caractères, quelle est la taille maximale prise en charge ?
- Quel éventail de valeurs la fonction personnalisée doit-elle prendre en charge ?
- La fonction personnalisée doit-elle prendre en charge de nombreux types de données (nécessitant des distributions de types implicites ou explicites) ?

Si vous ne répondez pas à ces questions, la fonction personnalisée risque de ne pas fonctionner ou d'entraîner l'instabilité du système. Elle peut également être exécutée mais renvoyer des données incorrectes.

Type de données de caractères

Tous les paramètres de chaînes de caractères sont fournis à la fonction personnalisée sous forme de chaînes terminées par NULL. Il n'existe aucune restriction quant à la longueur d'un paramètre d'entrée pour une fonction personnalisée. Cependant, cela n'est pas le cas pour la valeur de renvoi. La longueur maximale (à l'exclusion des terminateurs NULL) doit être prédéfinie dans le fichier cogudf.sql. Les produits de Cognos proposent la fonction personnalisée avec un tampon de la taille spécifiée (plus un octet pour le terminateur NULL). Le résultat ne peut pas être supérieur à la taille indiquée.

La longueur maximale d'une chaîne de caractères est 2^{31} - 1 octets.

Gardez également à l'esprit que pour les paramètres non-anglais, le nombre de caractères d'une chaîne n'est pas nécessairement égal au nombre d'octets. Un caractère peut requérir de 1 à 4 octets. La taille indiquée dans le fichier cogudf.sql représente le nombre maximum d'octets, pas de caractères.

Types de données d'entier court, d'entier et d'entier 64 bits

Les types de données numériques exactes représentent une gamme de valeurs croissante. Si les valeurs à gérer par un paramètre ou une valeur de renvoi sont toujours dans une gamme spécifique, il vaut mieux choisir le type applicable le plus court.

Les types d'entiers sont efficaces lorsqu'il s'agit d'effectuer des calculs arithmétiques et de fournir des valeurs exactes. Cependant, leur gamme de valeurs peut être limitative. Bien que les entiers 64 bits offrent une précision de 19 chiffres, ils ne sont pas pris en charge par tous les équipements.

Notez également qu'une échelle optionnelle peut être définie dans le fichier cogudf.sql pour un paramètre d'entier définissant le nombre de chiffres qui apparaissent après la virgule décimale. Cette option peut être utile pour les calculs monétaires.

Type de données décimales condensées

Comme indiqué précédemment, toutes les valeurs décimales condensées, qu'il s'agisse de paramètres ou de valeurs de renvoi, doivent comporter une échelle et une précision prédéfinies. Pour cette raison, leur utilité risque d'être limitée pour une fonction personnalisée. L'une des solutions consiste à utiliser la précision maximale pour tous les paramètres décimaux condensés et toutes les valeurs de renvoi.

Les valeurs décimales condensées entraînent un autre problème : le langage C ne les prend pas en charge. Toutes ces valeurs doivent être converties en une autre représentation de données avant d'être manipulées.

Types de données à double précision et à virgule flottante

Les valeurs numériques à double précision et à virgule flottante prennent en charge une gamme de valeurs plus large que les types de données numériques exactes, mais ne fournissent pour la virgule flottante et la double précision que 7 et 15 chiffres de précision. Si vous pensez que vous n'avez pas besoin d'une grande précision, ces données numériques approximatives sont probablement la meilleure option pour les paramètres numériques et les types de renvois.

Notez que les calculs à virgule flottante peuvent être plus longs à traiter que les opérations équivalentes effectuées sur des types numériques exacts, même si l'ordinateur en question utilise un calculateur à virgule flottante.

Type de données binaires

Lorsque vous traitez des données binaires, la fonction personnalisée est supposée connaître toutes les données binaires fournies en tant que paramètre tout comme une application est supposée connaître le contenu des données binaires renvoyées par une fonction personnalisée.

Distributions implicites

Les distributions de types permettent à une fonction d'être appelée avec des valeurs ne correspondant pas exactement aux types de données des paramètres de la fonction. Plusieurs distributions de type implicite sont possibles. Si une distribution ne peut pas être effectuée, une erreur survient lors de la préparation d'une interrogation SQL.

Bien que la conversion de types numériques exacts en types de données approximatifs soit prise en charge, il est possible que des chiffres essentiels soient perdus lors de la conversion. Des débordements peuvent se produire lors de conversions vers des valeurs décimales condensées ou depuis des valeurs décimales condensées, en fonction de la précision de cette valeur.

Le tableau suivant attribue des codes aux types de données afin de simplifier la présentation dans le deuxième tableau. Vous trouverez au-dessous la légende du deuxième tableau.

Type de données	Code	Type de données	Code
CogUInt16	UI16	CogChar	CH
CogInt16	I16	CogDate	DT
CogUInt32	UI32	CogTime	TM
CogInt32	I32	CogTimestamp	TS
CogInt64	I64	CogInterval	IN
CogFloat32	F32	CogBinary	BN
CogFloat64	F64	Décimales condensées	PD

Afin de / Depuis	I16	UI16	I32	UI32	I64	PD	F32	F64	CH	DT	TM	TS	IN	BN
I16	+	+	+	+	+	+	+	+	+					
UI16	●	+	+	+	+	+	+	+	+					
I32	●	●	+	+	+	+	+	+	+					
UI32	●	●	●	+	+	+	+	+	+					
I64	●	●	●	●	+	+	+	+	+					
PD	●	●	●	●	●	+	+	+	+					
F32	●	●	●	●	●	●	+	+	+					
F64	●	●	●	●	●	●	●	+	+					
CH	+	+	+	+	+	+	+	+	+	+	+	+	+	
DT									+	+		+		
TM									+		+	+		
TS									+	+	+	+		
IN									+				+	
BN														+

Symbole	Légende
+	La distribution est prise en charge.

Symbole	Légende
Vide	La distribution n'est pas prise en charge.
●	Quoique la distribution soit prise en charge, un débordement peut se produire.

Annexe

Cette annexe présente :

- un tableau contenant les identificateurs de Cognos pour les systèmes de bases de données pris en charge et les noms de fichiers .ini et .sql spécifiques à la base de données,
- des conseils de dépannage,
- des informations sur l'utilitaire de vérification de la syntaxe cogudf.exe,
- une liste des fichiers compris dans le kit de développement logiciel des fonctions personnalisées.

Identificateurs de bases de données

Base de données	Identificateur	Fichier .ini	Fichier SQL
DB2	D2	d2funct.ini	cogudfd2.sql
Informix	IF	iffunct.ini	cogudfif.sql
SQL Server de Microsoft	MS	msfunct.ini	cogudfms.sql
ODBC	OD	odfunct.ini	cogudfod.sql
Oracle	OR	orfunct.ini	cogudfor.sql
Sybase CT-Library	CT	ctfunct.ini	cogudfct.sql

Dépannage

Cette section décrit les actions à entreprendre si les messages d'erreur ou les problèmes répertoriés ci-dessous surviennent.

Erreur

```
La fonction <nom de la fonction> n'est pas disponible en tant que fonction intégrée,
fonction de base de données ou fonction externe.
```

Action

Assurez-vous que le fichier .sql se trouve dans l'un des emplacements suivants :

- le répertoire de travail courant,
- sous Windows, dans le dossier spécifié dans la section `ServicesL` du fichier cognos.ini ou dans le répertoire Bin,
- sous UNIX, dans le répertoire spécifié par les variables d'environnement `COGUDFSQL` ou `DMDBINI`.

Si le fichier est situé à un emplacement correct, assurez-vous que la définition de la fonction ne contient pas d'erreur de syntaxe. Vérifiez la syntaxe à l'aide de l'utilitaire cogudf.

Erreur

```
La fonction externe <nom de la fonction> requiert 2 paramètres mais 3 sont fournis.
```

Action

Un nombre incorrect de paramètres a été attribué à une fonction. Assurez-vous que :

- la définition de la fonction est correcte,
- la fonction est correctement mentionnée dans l'instruction SQL,
- il n'y a qu'une seule définition de la fonction dans le fichier .sql approprié, car seule la première définition d'une fonction est utilisée.

Erreur

```
La fonction externe <nom de la fonction> n'est pas accessible.
```

Action

Il est impossible de trouver la fonction. Il existe plusieurs explications possibles :

- le nom de la bibliothèque indiquée dans le fichier .sql ne correspond pas au nom de la bibliothèque partagée contenant la fonction externe,
- le nom de la fonction dans le module C et le nom de la fonction C définie dans le fichier .sql ne sont pas les mêmes,
- sous UNIX, aucune valeur n'a été attribuée à la variable d'environnement du chemin de la bibliothèque spécifique à la plate-forme,
- sous UNIX, la bibliothèque partagée ne se trouve dans aucun des répertoires indiqués dans la variable d'environnement du chemin de la bibliothèque,
- sous Windows, la bibliothèque partagée ne se trouve dans aucun des répertoires indiqués dans la variable d'environnement PATH,
- la fonction n'a pas été exportée depuis la bibliothèque partagée ou la DLL,
- le nom de la fonction tel qu'il est indiqué dans la liste d'exportation ne correspond pas au nom de la fonction dans le module C.

Données incorrectes ou arrêt inattendu

Sous Windows, si vous recevez des données incorrectes d'une fonction ou si la fonction entraîne la fermeture de l'application, assurez-vous que vous avez compilé la DLL avec un alignement de 8 octets.

Erreurs de la base de données

Si vous recevez un message d'erreur de base de données lors de l'appel d'une fonction personnalisée de base de données dans une instruction SQL, assurez-vous que :

- le nom de la fonction personnalisée a été correctement entré dans le fichier .sql correspondant,
- la fonction personnalisée de la base de données possède les qualifications nécessaires.

  Par exemple, si la fonction personnalisée est installée dans un schéma autre que le schéma implicite, elle nécessitera probablement une qualification de schéma dans la définition de fonction du fichier .sql. En général, il vaut mieux qualifier entièrement les noms de fonctions personnalisées de bases de données afin de supprimer toute ambiguïté et d'éviter que des erreurs de bases de données se produisent si la base de données ne réussit pas à trouver les fonctions personnalisées.

Utilitaire de vérification de la syntaxe cogudf

Pour rechercher toute erreur de syntaxe dans les fichiers .sql, servez-vous de l'utilitaire de Windows.

Si vous développez des fonctions personnalisées pour une plate-forme UNIX, vous pouvez utiliser cogudf pour déboguer les fichiers .sql dans un environnement Windows avant de déployer les fonctions personnalisées sur des systèmes UNIX.

Pour utiliser cogudf, il suffit de l'exécuter. Aucune option de ligne de commande n'est nécessaire. L'utilitaire cogudf effectue une recherche tout d'abord dans le dossier courant, puis dans le dossier indiqué dans la section *ServicesL* du fichier Cognos.ini.

L'utilitaire cogudf signale toute erreur rencontrée dans les fichiers .sql. Dans les environnements Windows, les résultats sont envoyés au fichier journal indiqué à l'entrée *UDF Log File* de la section *[ServicesL]* du fichier Cer*n*.ini :

```
[ServicesL]
UDF Log File=d:\temp\udf.log
```

Si cette entrée n'existe pas ou qu'aucune valeur ne lui est attribuée, aucune information n'est inscrite dans le fichier journal. Si l'entrée existe, le fichier est créé ou les informations sont ajoutées au fichier lorsqu'il existe.

Votre produit de Cognos inscrit également des informations dans le fichier journal lorsqu'il détecte une fonction personnalisée dans une expression SQL.

Sous UNIX, la variable d'environnement `COGUDFLOG` permet d'accéder au fichier journal. Si vous précisez cette variable, des informations sont transmises au fichier journal uniquement lorsque le serveur de requêtes trouve une fonction personnalisée.

Fichiers du kit de développement logiciel des fonctions personnalisées

Les fichiers et les exemples du kit de développement logiciel ont été conçus à l'aide de Visual C++ 4.2 de Microsoft Developer Studio 97.

Fichier	Description
cogudfty.h,	Pour UNIX et Windows. Fichier d'en-tête définissant les types de données pris en charge par les fonctions personnalisées. Figure dans le fichier cogudf.h.
cogudf.h	Pour UNIX et Windows. Fichier d'en-tête définissant les macros, les types et les fonctions nécessaires pour les fonctions personnalisées.
udfLib.a	Pour UNIX. Bibliothèque contenant les fonctions définies dans cogudf.h. Doit être liée à toute DLL contenant des fonctions personnalisées si l'une des fonctions de support des fonctions personnalisées de Cognos est utilisée.
udflib.lib.	Pour Windows. Bibliothèque contenant les fonctions définies dans cogudf.h. Doit être liée à toute DLL contenant des fonctions personnalisées si l'une des fonctions de support des fonctions personnalisées de Cognos est utilisée.
cogudf.exe	Pour Windows. Utilitaire recherchant des erreurs de syntaxe ou sémantiques dans tous les fichiers .sql disponibles contenant des définitions de fonctions externes et de bases de données.
cogudf.sql	Pour UNIX et Windows. Fichier .sql contenant des définitions de fonctions personnalisées.

Fichier	Description
fonction personnalisée bdd.txt	Fichier lisez-moi pour la fonction personnalisée de base de données. Explique comment utiliser les fichiers suivants : fonction personnalisée oracle.txt mise à jour de orfunct.txt fonction personnalisée informix.txt cogudfxx.sql
encrypt - lisez-moi.txt	Fichier lisez-moi pour les fonctions personnalisées d'encodage externe. Explique comment utiliser les fichiers suivants : mise à jour de impfunct.txt Encrypt.c Encrypt.def Encrypt.dll mise à jour de cogudf (encrypt).sql mise à jour de impfunct (encrypt).txt
html - lisez-moi.txt	Fichiers pour les fonctions personnalisées HTML externes. Pubhtml.c Pubhtml.c Pubhtml.def Pubhtml.dll mise à jour de cogudf (html).sql mise à jour de impfunct (html).txt
Udf.mdp Udf.mak Udf.ncb	Fichiers de projet de Microsoft Developer Studio pour les exemples.

Index

U

V

W