ModÃ¨le de dÃ©tail dans Integrator

Description

Integrator est gÃ©nÃ©ralement utilisÃ© pour lire ou extraire des donnÃ©es et ensuite de transformer ces donnÃ©es en vue de la construction d'un ModÃ¨le. Un ModÃ¨le lui-mÃªme peut Ãªtre utilisÃ© comme entrÃ©e dans un processus Integrator. Cette astuce explique comment.

Utilisez un ModÃ¨le, en particulier, un ModÃ¨le de dÃ©tail (Detail Model) ou indexÃ©, comme un objet de source d'entrÃ©e dans un processus Integrator.
Filtrez les donnÃ©es en utilisant un tableau PARM Integrator.

Il y a des cas oÃ¹ nous avons besoin de filtrer l'entrÃ©e d'un ModÃ¨le dÃ©taillÃ© pour un affichage personnalisÃ© des donnÃ©es. Il y a Ã©galement des cas oÃ¹ nous avons besoin de contrÃ´ler les filtres de ces donnÃ©es en utilisant des fichiers externes. Cette astuce aborde ces cas.

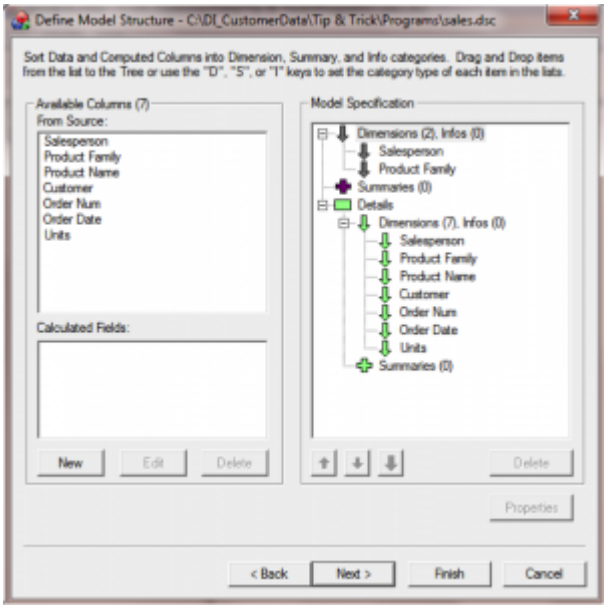
Un ModÃ¨le de dÃ©tail peut Ãªtre utilisÃ© comme un objet d'entrÃ©e dans Integrator, et on peut utiliser les attributs suivants pour filtrer les entrÃ©es, nous permettant de spÃ©cifier les valeurs Ã garder des ModÃ¨les:

filter_input spÃ©cifie un fichier (ou flux de donnÃ©es) Ã utiliser comme source de valeurs Ã utiliser comme filtre.
filter_columns est une liste des colonnes utilisÃ©es comme des filtres Ã partir du flux de donnÃ©es filter_input.
filter_values est une liste (ou tableau) des valeurs spÃ©cifiques Ã garder Ã partir du ModÃ¨le d'entrÃ©e.

Cette astuce utilise des tableaux de valeurs Ã partir de fichiers externes. Nous fournissons ces valeurs aux paramÃ¨tres, qui sont Ã leur tour utilisÃ©s dans l'attribut de l'objet ***filter_values*** de l'objet entrÃ©e ModÃ¨le.

Etape 1. Construisez un ModÃ¨le de dÃ©tail (ou indexÃ©).

La figure 1 montre un ModÃ¨le de dÃ©tail de l'Ã©chantillon utilisÃ© dans cette astuce.



Â© Bimaxis - Tous droits rÃ©servÃ©s

Figure 1.

Etape 2. CrÃ©ez le tableau de filtre

Dans notre exemple, nous avons deux fichiers qui sont utilisÃ©s comme source pour nos filtres, un fichier pour chaque colonne Ã filtrer. Les deux fichiers contiennent une liste de valeurs que nous voulons garder au niveau de l'entrÃ©e, et ne contiennent aucun en-tÃªte de colonne. Un fichier va filtrer les vendeurs (Salespersons) et l'autre va filtrer les familles de produits (Product Family) . Dans le ModÃ¨le, nous avons 34 enregistrements contenant des donnÃ©es pour les Ã©lÃ©ments suivants:

Product Family : ***Butter, Ice Cream*** et ***Milk***
Salespersons : ***Burritt, Bailey*** et ***Orr***

Ces dossiers seront filtrÃ©s sur la base des fichiers tableau d'entrÃ©e.

Product Family Array
Butter
Ice Cream

Salesperson Array
Burritt
Orr

AprÃ¨s filtrage, nous ne devrions voir que ***Butter*** et ***Ice Cream*** vendus par ***Burritt*** et ***Orr***.

Ã?tape 3. DÃ©finisez le tableau PARM d'Integrator

Les paramÃ¨tres tableau sont dÃ©finis dans le script Integrator et sont configurÃ©s pour lire les deux fichiers qui contiennent les valeurs de filtre pour chaque colonne. Voir Figure 2.

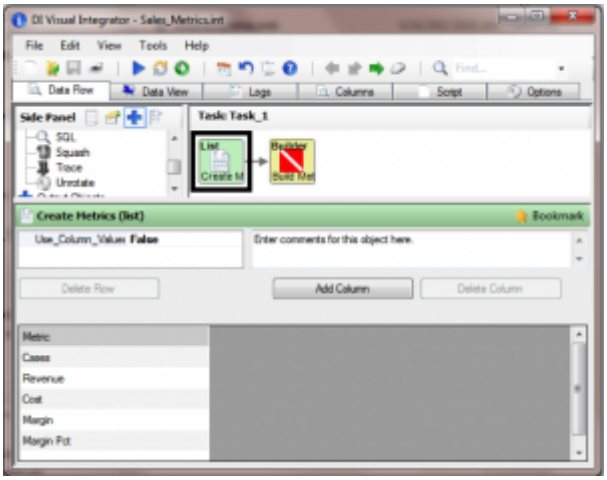


Figure 2.

Ã?tape 4. DÃ©finissez les attributs de l'objet INPT
Pour afficher l'utilisation de paramÃ¨tres dans Visual Integrator, la version 3.0 (6) ou supÃ©rieure est nÃ©cessaire.

Pour voir le script en dehors de Visual Integrator, cliquez sur Script 1 ci-dessous.

Â© Bimaxis - Tous droits rÃ©servÃ©s

```
version "1";

object 'PARM' "parms" {
   parms = {
            { name = "ProdFam" , default = "../data/Product_Family.tbl",type = "array"},
            { name = "SalesRep", default = "../data/Salesperson.tbl"   ,type = "array"}
           }
};

object 'TSKL' "Main" {
        contents = {
                'filter_inputs_2parms'
        }
};

object 'TASK' "filter_inputs_2parms" {
        inputs = {
                "model_in",
        },
        output = "fileout",
};

object 'INPT' "model_in" {
        input_type     = 'model',
        detail         = 'true',
        models         = { '../Models/sales.mdl' },
        filter_columns = { "Product Family", "Salesperson" },
        filter_values  = { '$(ProdFam)'    , '$(SalesRep)' }
};

object 'OUTP' "fileout" {
   output_type      = "fileout",
   file_type        = "column_headers",
   input            = "model_in",
   filename         = "../temp/filter_inputs_2parms.txt",
```

Script 1.

Dans les figures 3 et 4, nous voyons que les deux colonnes sÃ©lectionnÃ©es pour filtrer les donnÃ©es sont bien dÃ©finies.

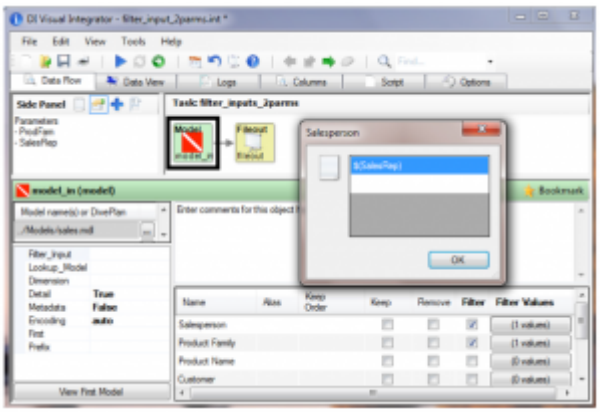


Figure 3.

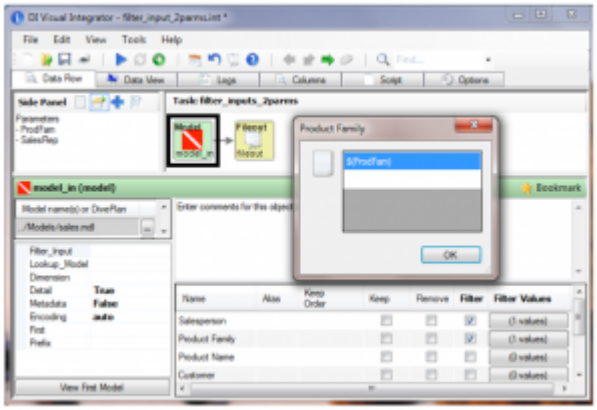


Figure 4.
Les paramÃ¨tres tableau prÃ©cÃ©demment dÃ©finis sont utilisÃ©s comme "filter_values" plutÃ´t que des valeurs explicites dans les dialogues des filtres Salesperson ou Product Family.

Â© Bimaxis - Tous droits rÃ©servÃ©s

Ã?tape 5. Testez le script Integrator

Dans notre exemple, en utilisant les tableaux dÃ©crits prÃ©cÃ©demment, 16 enregistrements sont crÃ©Ã©s, comme illustrÃ© ci-dessous:



Tableau 1.

Nous pouvons modifier les filtres pour inclure "All values" pour l'un des tableaux comme montrÃ© dans l'exemple ci-dessous:



Nous voyons des modifications au niveau de la sortie afin de reflÃ©ter tous les ***Product Family*** pour ***Burritt*** et ***Orr***, produisant 22 enregistrements de ventes, par opposition aux 16 enregistrements prÃ©cÃ©dents :

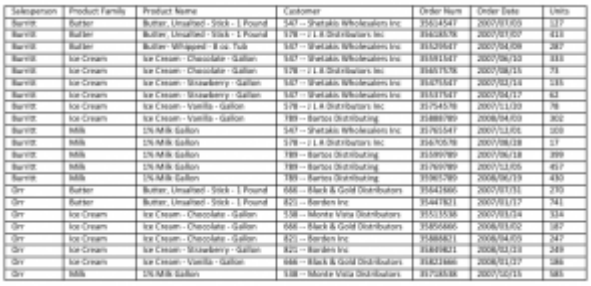


Table 2.

La possibilitÃ© d'utiliser "All values" dans le tableau simule les rÃ©sultats qui seraient passÃ©s lors de la sÃ©lection de "All values" (toutes les valeurs) dans une QuickView. Cela est courant lors de l'utilisation d'une Marque dans DivePort pour passer des valeurs de Quickview dans un script Tunnel Integrator.

Les applications pratiques d'utilisation d'un ModÃ¨le de dÃ©tail et d'un tableau PARM dans Integrator comprennent:

- Le passage des valeurs de QuickView de DivePort sÃ©lectionnÃ©es par l'utilisateur Ã un script Tunnel Integrator.
- Le changement des filtres et des chaÃ®nes de valeur en dehors du script Integrator lors d'un processus de construction de nuit.

Tags

1. Data Integrator
2. script

Â© Bimaxis - Tous droits rÃ©servÃ©s