Pentaho Data Integration

version 2.5.0

Foire aux questions

traduction française *en cours*
Partie 1/6

en cas de doute se référer au document original en anglais

Table des matières

1 PRÉFACE

Ce document contient la “Foire aux questions” fréquemment posées au sujet de “Pentaho Data Integration”, anciennement connu sous le nom de Kettle. Les questions et réponses de ce document sont principalement issues des questions posées sur le forum de discussion. Si une question se trouve ici, il est probablement inutile de la poser de nouveau sur le forum de discussion. Cependant, si une réponse de ce document vous apparaît ,ne pas être claire, merci de faire référence à cette foire aux questions dans le forum de discussion.

2 LES QUESTIONS DES UTILISATEURS DÉBUTANTS

2.1 Quelle est la différence entre une transformation (ndt en anglais transformation) et une tâche (ndt en anglais job) ?

Q: Dans Spoon, je peux fabriquer des tâches et des transformations. Quelle est la différence entre les deux ?

A: Les transformations servent à déplacer et transformer des lignes depuis un flux source de données vers un flux cible. On utilise les tâches à un niveau conceptuel plus élevé dans le contrôle des processus, par exemple pour exécuter des transformations, envoyer un courriel sur erreur, charger un fichier etc.

2.2 Règle sur le mélange des type de lignes sur un lien dans une transformation

Q: Dans les documentations, je lis que les types de lignes ne peuvent pas être mélangés; qu'est-ce que cela signifie ?

A: Ne pas mélanger les lignes signifie que chaque ligne qui est envoyée sur un même lien doit avoir la même structure : les mêmes noms de champs, types de champs, ordre des champs. Si vous souhaitez faire quelque chose du genre « ajouter un champ optionnel si tel condition est vraie ou fausse, alors », cela ne fonctionnera pas (parce que vous récupérerez des type de lignes différents selon la condition). Vous pouvez basculer en « Mode sécurisé » pour vérifier explicitement cette l'identité des types au démarrage.

La vérification des types de lignes est automatiquement réalisée depuis la version 2.5.0 lors de l'étape de design/vérification. Cependant, le « mode sécurisé » doit être activé pour que cette vérification se fasse dès le démarrage.

De la même façon qu'une ligne d'une table de base de données aura toujours la même structure, vous ne pourrez pas enregistrer plusieurs types de lignes dans un flux PDI.

Théoriquement, la raison provient de ce que PDI veux être capable de réaliser des transformations uniformes et consistantes sur vos données. Se doter de type de lignes variables rendrait beaucoup plus complexe ces transformations.

Techniquement, la plupart des étapes d'une transformation sont optimisées (en utilisant des techniques d'identifiant et d'indexation). Se doter de types de lignes variables rendrait cette optimisation impossible.

2.3 Nom de champ en double au sein d'une transformation

Q: Puis-je avoir deux mêmes noms de champ au sein d'une même transformation ?

A: Vous ne pouvez pas. PDI vous avertira de cette impossibilité dans la plupart des cas de doublons sur le nom de champs. Avant la version 2.5.0, vous pouviez forcer cette contrainte, mais seulement la première valeur des champs en doublon était utilisée.

2.4 Au sujet des chaînes de caractères vides ou nulles (NULL)

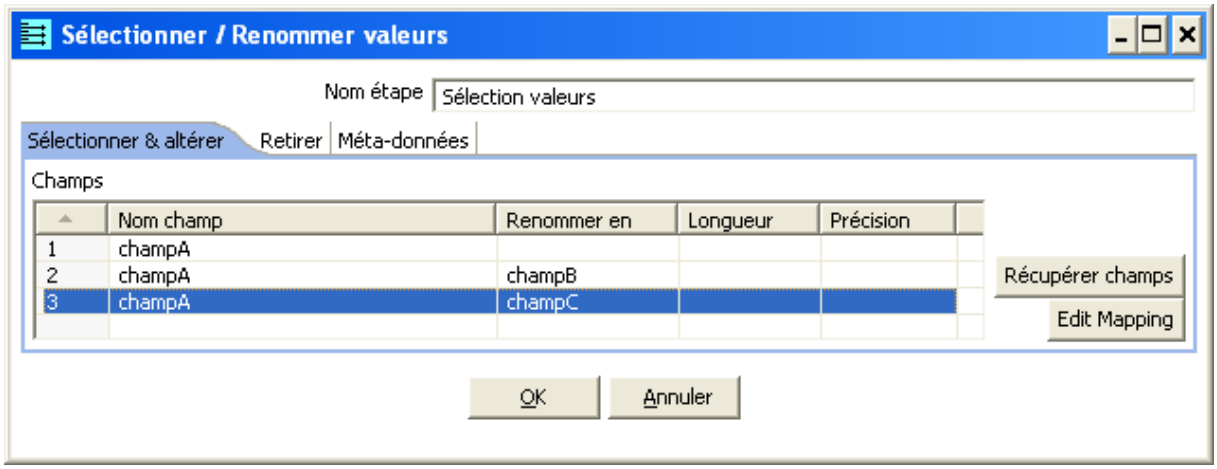
PDI suit Oracle dans son utilisation des chaînes de caractères vides ou nulles : il considère que c'est la même chose. Si vous trouvez une étape qui ne suit pas cette convention, signalez-le. C'est probablement un bogue.

2.5 Comment copier/dupliquer un champ au sein d'une transformation ?

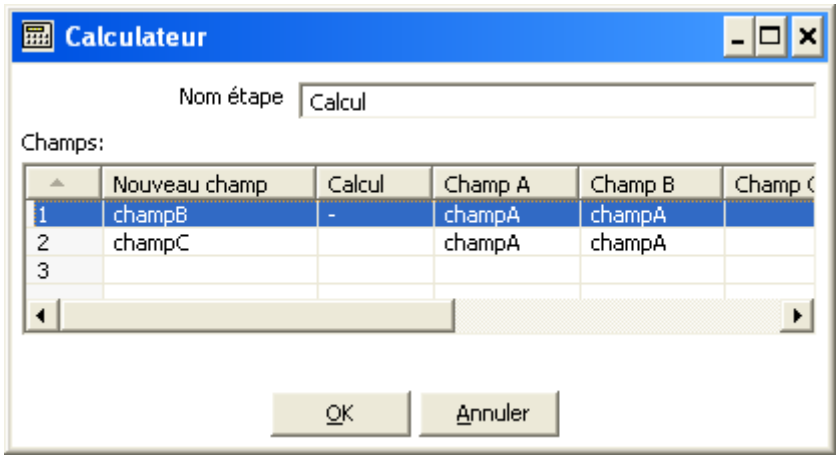
Q: Comment copier/dupliquer un champ au sein d'une transformation ?

A: Plusieurs solutions existent :

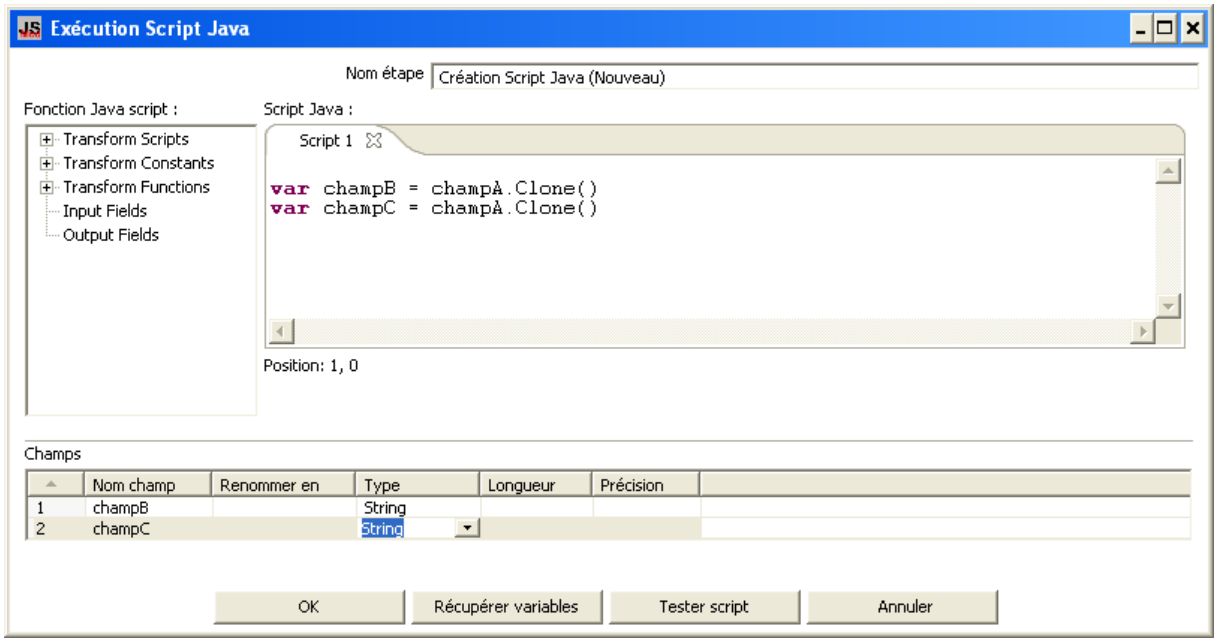
1) Utiliser une étape « Sélectionner valeurs » en renommant un champ sélectionné. Le champ original sera alors dupliqué avec un nouveau nom. On aura par exemple :
Cela dupliquera le champA en champB et champC



2) Utiliser une étape « Calculatrice » et se servir de l'opération NLV(A,B) comme calcul (non disponible en version 2.4.5) comme suit :
Cela aura le même effet que la première solution. Cela dupliquera le champA en champB et champC



3) Utiliser une étape « JavaScript » pour copier le champ
Cela aura le même effet que les précédentes solutions : 3 champs dans le flux de sortie qui sont des copies les uns des autres :champA, champB, champC.



2.6 Comment réaliser une jointure au sein d'une base de données avec PDI ?

Q: Comment réaliser une jointure au sein d'une base de données avec PDI ?

A: Si vous désirez réaliser une jointure de deux tables provenant d'une même base de données, alors, vous utiliserez l'étape « Entrée table » et exécuterez une instruction SQL de jointure. Avec une seule base de données, c'est la méthode la plus rapide.

Si vous désirez réaliser une jointure de deux tables qui ne proviennent pas de la même base de données, vous pouvez utiliser une étape « Jointure Base de données » mais vous devez vous rendre compte que PDI exécutera une instruction pour chacune des ligne en entrée de la jointure. Vous pourrez trouver des indications supplémentaires dans les « trucs et astuces hebdomadaires » sur le site de Pentaho (http://kettle.pentaho.org/tips/).

2.7 Comment sérialiser les transformations ?

Q: Toutes les étapes, par défaut, tournent en parallèle au sein d'une transformation. Comment puis-je faire pour qu'une ligne soit complètement traitée avant que PDI commence à traiter la ligne suivante ?

A: Ce n'est pas possible, PDI est construit sur cette notion de traitement en parallèle. Il faudrait changer d'architecture pour permettre une « sérialisation » des traitements. Cela induirait un temps de traitement très lent.

3 LES QUESTIONS D'AUTRES UTILISATEURS

3.1 Chaîne de caractères plus longue que la longueur définie

Q: J'essaie de limiter une chaîne de caractères avec le paramètre « longueur » mais des chaînes plus longues passent au travers de mon étape et cela cause une erreur SQL par la suite. Le programme n'est-il pas censé tronquer la chaîne de caractères à la longueur saisie ? Ou dois-je tronquer moi-même la chaîne de caractères avant insertion ? Si oui, quel est la meilleure façon de faire cela ?

A: Une transformation fonctionne sur les méta-données. Au début de la conception de Kettle, le concepteur a pris le parti de travailler sur les méta-données et non les données elles-mêmes. Ainsi, l'utilisateur n'est pas toujours arrété dans son cheminement par des messages d'erreur sur le typage des données. Cependant, si vous désirez tronquer un champ, vous pourrez le faire en écrivant un script javascript comme celui ci-dessous :

```
var string = originalfield.getString();
if ( string.length() > 50 )
{
     string = string.substring(0,50);
}
```

3.2 Le point décimal n'est pas affiché dans la sortie CSV (ndt fichier délimité par des virgules)

Q: Dans une table de ma base de données, j'ai une colonne contenant un décimal :

```
100.23
100.20
100.00
100.00
```

En me servant d'un étape « sortie vers fichier » et en utilisant l'extension CSV, lorsque j'ouvre ce fichier dans mon tableur (MS Excel), j'obtiens ceci :

```
100.23
100.20
100
100
```

Q: Comment obtenir systématiquement les décimales (“.00”) même si elles sont nulles ?

A: En premier lieu, il vous faut utiliser un format "#.00" pour ce champ dans le paramétrage de votre étape. Vous obtiendrez une précision de deux décimales, quelles qu'elles soient, ce que vous pourrez vérifier en ouvrant votre fichier dans un programme d'édition de fichier (Bloc Notes par exemple).

Si vous ouvrez le fichier d'extension csv avec MS Excel, les deux décimales lorsqu'elles sont nulle n'apparaîtront peut-être pas car MS Excel utilise par défaut un formatage qui cache les décimales nulles. Utilisez donc l'étape « Sortie vers excel » qui, en version 2.4.0, permet des formatages simples pour Excel.

3.3 L'appel à des fonctions retournant une valeur boolénne échoue avec Oracle

Q: La valeur retour de type booléen d'une fonction de base de données amène une erreur "Type de colonne invalide" dans Pentaho Data Integration. Voici ma fonction :

```
FUNCTION test(id number) RETURN BOOLEAN IS
BEGIN
     return true;
END
```

A: Ceci est un comportement interne à Oracle. Un tel code retour n'est valide que dans un bloc PL/SQL. En effet, une table oracle ne peut contenir une valeur booléenne (qui ne fait pas partie du standard SQL). Il est donc suggéré de retourner une valeur de type « varchar » avec 'true' / 'false' (ou 'Y' / 'N'). Ensuite, en convertissant le type de données en boléen, vous pourrez récupérer un booléen.

Voici une référence à ce comportement dans la documentation d'Oracle :

> “It is not feasible for Oracle JDBC drivers to support calling arguments or return values of the PL/SQL RECORD, BOOLEAN, or table with non-scalar element types. However, Oracle JDBC drivers support PL/SQL index-by table of scalar element types. For a complete description of this, see Chapter 11, “Accessing PL/SQL Index-by Tables”
>
> As a workaround to PL/SQL RECORD, BOOLEAN, or non-scalar table types, create wrapper procedures that handle the data as types supported by JDBC. For example, to wrap a stored procedure that uses PL/SQL booleans, create a stored procedure that takes a character or number from JDBC and passes it to the original procedure as BOOLEAN or, for an output parameter, accepts a BOOLEAN argument from the original procedure and passes it as a CHAR or NUMBER to JDBC. Similarly, to wrap a stored procedure that uses PL/SQL records, create a stored procedure that handles a record in its individual components (such as CHAR and NUMBER) or in a structured object type. To wrap a stored procedure that uses PL/SQL tables, break the data into components or perhaps use Oracle collection types.“

3.4 Différence entre variables/arguments dans le lanceur

Q: Lorsque j'exécute une transformation, la boîte de dialogue propose deux tables : l'une pour les arguments, l'autre pour les variables. Quelle est la différence ?

A: Les arguments sont ceux de la ligne de commande qui seront spécifiés pendant le traitement en différé (via Pan). Les variables sont des variables d'environnement ou de PDI qui seront affectées lors d'une étape précédente, une transformation précédente (pour une tâche) ou par le système d'exploitation.

3.5 Comment utiliser les connexions aux bases de données avec un référentiel

Q: Lorsque l'on crée directement dans le référentiel une nouvelle connexion à une base de données, comment peut-on l'utiliser dans une transformation (nouvelle ou existante) ?

A: Créez une nouvelle transformation (ou tâche) ou fermez et ouvrez de nouveau les transformations chargées dans Spoon pour « voir » votre nouvelle connexion.

3.6 Au sujet de l'insertion de booléens au sein d'une base MySQL

Q: Comment insérer des booléens dans une base MySQL, sachant que PDI utilise les valeurs 'Y' et 'N' et que cela ne peut être inséré comme une colonne de type BIT(1) dans MySQL ?

A: BIT n'est pas un type standard SQL. Ce n'est pas non plus un standard MySQ v5. De plus, un BIT utilise 2 bytes dans MySQL. C'est pourquoi dans PDI, nous avons fait le choix de la sécurité et enregistrons les booléens comme char(1). La façon de faire simple est de changer le type de données avec une étape « Sélectionner valeurs » en entier qui vaut 0 pour « false » et 1 pour « true », comme demandé dans MySQL.

3.7 L'étape « Calcul » ignore le type du résultat lors des divisions

Q: Dans un transformation utilisant la fonction A/B au sein d'une étape « calcul », le résultat de la division de deux champ de type « entier » devient un nombre de type Number(6, 4). Par exemple, si je demande l'exécution de 28/222, j'obtiens 0.0 au lieu de 0.1261 which I expected. So it seems the result type is ignored. If I change the input types both to Number(6, 4) I get as result 0.12612612612612611 which still ignores the result type (4 places after the comma).

Q: Why is this?

A: Length & Precision are just metadata pieces.

If you want to round to the specified precision, you should do this in another step. However: please keep in mind that rounding double point precision values is futile anyway. A floating point number is stored as an approximation (it floats) so 0.1261 (your desired output) could (would probably) end up being stored as 0.126099999999 or 0.1261000000001 (Note: this is not the case for BigNumbers)
So in the end we round using BigDecimals once we store the numbers in the output table, but NOT during the transformation. The same is true for the Text File Output step. If you would have specified Integer as result type, the internal number format would have been retained, you would press "Get Fields" and it the required Integer type would be filled in. The required conversion would take place there and then. In short: we convert to the required metadata type when we land the data somewhere, NOT BEFORE.

3.8 Questions au sujet de l'étape « client HTTP »

3.8.1 The HTTP client step doesn't do anything

Q: The HTTP client step doesn't do anything, how do I make it work?

A: The HTTP client step needs to be triggered. Use a Row generator e.g. and link that with a hop to the HTTP client.

3.8.2 The HTTP client step and SOAP

Q: Does the HTTP client support SOAP?

A: No, it just calls an URL with arguments. Future steps may provide SOAP functionality, Work is underway on a WebService step supporting WSDL. The first experimental version will appear in PDI v2.5.0

3.9 Questions au sujet de Javascript

3.9.1 How to check for the existence of fields in rows

Q: How do I check for the existence of a certain value in a row?

A: The following snippet will let you check this. But keep in mind that you can not mix rows in PDI, all rows flowing over a single hop have to have the same number of fields, which have to be of the same name and type.

The snippet:

```
var idx = row.searchValueIndex("lookup");
if ( idx < 0 )
{
     // doesn't exist
}
else
{
     var lookupValue = row.getValue(idx);
}
```

3.9.2 How to add a new field in a row

Q: How do I add a new field to a row in a Javascript step?

A: Note upfront that the order in which add fields to a row is important. You always have to add the fields in the same order to the row to always get the same structure of row. Now to add a field:

1) Define it as “var” in the source and add it as a field in the lower level table:

2) Clone an existing field and add it to the row:

```
var newValue = row.getValue(0).Clone();
row.addValue(newValue);
```

3) Use following steps

```
var value = Packages.be.ibridge.kettle.core.value.Value.getInstance();
value.setName("name_of_field");
value.setValue("value_of_field");   // possibly using types other than String
row.addValue(value);
```

3.9.3 How to replace a field in a row

Q: How do I replace a field in a row in a Javascript step?

A: Use setValue on the input field as follows (assuming field1 is a field in the input row):

```
field1.setValue(100);
setValue() takes all possible types that can be used in PDI (also String,
Dates, ...).
```

3.9.4 How to create a new row

Q: How do I create a new row in the javascript step?

A: In the javascript step you have 2 special objects: “row” and “_step”:

var newRow = row.Clone(); // make a copy

```
// modify newRow
_step_.putRow(newRow);    // sends an extra row on the output of the step
Note that you should make sure that the row you're putting out to the next steps is of the same layout as the row that normally gets sent out. And also not that newRow is being put out before the regular row.
```

3.9.5 How to use something as nvl in javascript?

Q: How do I use something as the Oracle nvl in javascript?

A: You can use the following construction (to get something as nvl(a, '0'))

```
var a;
if ( fieldname.isNull() )
{
    a = '0';
}
else
{
    a = fieldName.getString();
}
```

and you can also use:

```
fieldName.nvl('1');
```

which would replace the value of fieldName with the value of '1' if fieldName is null.

3.9.6 Example of how to split fields

Q: In a field I have merchant code containing numbers and characters, ex. "12345McDonalds", i want to split it, the field doesn't have a consistent layout and I want to split the first part which is numeric from the second part. How to do this?

A: Use following piece of javascript, Merchant_Code is the name of the input field

java;

```
var str = Merchant_Code.getString();
var code = "";
var name = "";
for (i = 0; i < str.length(); i++ )
{
    c = str.charAt(i);
    if ( ! java.lang.Character.isDigit(c) )
    {
```

```
            code = str.substring(0, i);
            name = str.substring(i);
            Alert("code="+code+", name="+name);
            break;
        }
}
```

The Alert() is just to show the fields of course. After the outer for loop you could add code and name in new separate fields e.g.

3.10 Shell job entry questions

3.10.1 How to check for the return code of a shell script/batch file

The Shell script considers a return code of 0 to mean success, anything else is failure. You can use hops to control the resulting flow.

4 QUESTIONS D'UTILISATEURS SANS SUITE

Ces choses qui ont été proposées mais auxquelles nous avons renoncées

4.1 Implement connection type as a variable parameter

"I can parametrize host, user-id, password in database steps in transformations using variables, but I can't set the connection type like that yet. I have jobs that need to execute on multiple types of databases (Oracle, MySql, ...) and I would like the connection type set in the same way so that I can run the same transformation on multiple databases."

This has been requested a few times. Reasons for not implementing it in PDI:

- For anything but simple SQL statements the SQL you write will be database type dependent. E.g. If you use Oracle analytics you're SQL won't run anymore on DB2 or MySql. Currently you know the type of the database and you could use the full functionality of the database;
- Starting with PDI version 2.3.1 and continuing in later versions more database specific settings were introduced. So just specifying "this is Oracle, MySql, ..." is not sufficient anymore, and a way to parametrize these specific options would need to be found as well (which would make it pretty complex);
- How many data warehouses run on multiple types of databases. Most data warehouses are created based on specific operational systems and targeting only specific database types since resulting reports and cubes would also need to run on those databases. So the use of connection type parameterizing would probably also not be that huge.

Possible workaround: maintain duplicate jobs for multiple databases. Alternatively you can use the generic ODBC which supports variable substitution for the driver as of PDI version 2.5.0GA. The disadvantage of the latter solution being that the special database processing for some types of database will not be done of course.

4.2 Implement "on the fly DDL" creation of tables, ...

"I want tables to be created on the fly when they don't exist yet".

The reason for not implementing "on the fly DDL" in PDI is that it would work for small examples, but it would make your DBA's life harder for real projects. Most companies have setup pretty complex database privileges which would be hard to maintain using "on the fly DDL".

In the database steps there usually is an "SQL" button thats shows the DDL that could be executed. This should mostly be used in quick mock-ups, not for real projects. Do a proper design for real projects.

We are not supporters of any "on the fly DDL" in any step.

4.3 Implement a step that shows a dialog and asks parameters

The reason for not implementing this is that PDI is an ETL tool, not a reporting tool. You cannot assume there will be an interactive user answers questions when a job/transformation is running.

You can have a look at the Pentaho framework where you can indeed build webpages that ask for parameters, drop-downs, selectors, ... to parametrize transformations this way. Alternatively in the examples directory there's an example called dialog.ktr that shows a way to make a dialog box using javascript in PDI.

Besides this, as of PDI v2.4.0 the launching dialog also supports the entering of parameters via a GUI style which satisfies most of the people asking to enter parameters.

As of 2.5GA there is a job entry called “Display Msgbox Info” that will display a message in a dialog box, however even this functionality should only be used for debugging purposes. There's absolutely no guarantee that it will work when scheduled (e.g. On UNIX if you do not have a controlling terminal the job will probably even abort when you display a dialog box).

4.4 Implement serialization of transformations using reflection

It has been advised a few times to use XML “bean” serialization to automatically serialize and deserialize jobs/transformation as that would make it “easier” to develop them.

It is possible to end up with decent XML code using say XStream, but only by setting all the element names manually etc, you end up with a situation that is far worse than the current one.

The situation with the XML serialization that we have right now is:

1) it always works, regardless if the element/parent element/etc exists or not, is empty or not, etc;

2) it's extremely simple, everyone understands it, even those that don't know the XStream/whatever API;

3) the generated XML is understandable and readable. (more so than other tools I might add) This makes PDI more open.

4) we rarely every have any issues with it, no surprises, no NPEs flying around etc. It's very resilient.

5) It's easy to test and debug.

That should be plenty of reason to keep it stable for the time being, ***not*** using reflection.

4.5 Implement retry on connection issues

Usually what is asked is as follows: “My jobs occasionally receive an "I/O Error: Connection reset" while performing DB Lookup steps. This causes the entire transformation to fail. Is there any way to configure the transformation to retry if there are connection issues, or to open another connection?”

An I/O error usually means that the underlying hardware is failing, let's assume it's the network. The connection with the database is dropped so you don't know which SQL statement was executed correctly and which one wasn't. You can't ask the database because the connection is dead and you can't assume that the database rolls back the

statements until the last commit.
Well, the problem is that it can occur at the time you do a commit. If you commit 10.000 rows at a time: did they go into the table or not?

4.6 Implement GUI components in a transformation or job

Some people request implementation of GUI elements in a transformation or a job. In ETL this is normally not a good thing, most of the times there are not going to be people around to press buttons when the transformation or job is running. The transformation/job may also be running in a an environment that will not allow it to display dialogs: e.g. When running on a carte server on a remote system.

Some GUI components slipped in, but these are mostly for debugging purposes.

Hardcoding Locale

”No-one is going to use dates in another format as the default English Locale, so why don't we hardcode the English Locale.” or whatever other Locale feature that can be hardcoded (numbers, ...)

Well, not all people use English as default Locale (or your own preference for that matter), so hardcoding a locale is not a good idea, people should have a choice of default Locale.

See also “on Locales” in the developer guidelines.

5 QUESTION DE DÉVELOPPEURS

Directives de développement

5.1 Les priorités du développement

5.1.1 Exactitude/Uniformité

If a tool is not correct it's not going to be trusted however fast it may be. It can't be that the same input will produce output A in one case, and output B in another case.

5.1.2 Compatibilité arrière

Everyone like upgrades to go smoothly. Install the new binaries and be able to run without testing is the ideal. Of course, in some cases compatibility has to be broken for the greater good in the long term, but then it should be clearly documented (for upgrades).

5.1.3 Rapidité

There is a need for a speed. No-one wants to wait 30 minutes to insert 100.000 rows.

5.1.4 Facilité d'emploi

It should not be a torment to use a tool. It should allow both novice and expert users to get their job done. As example: any XML or configuration file should have a GUI element to manage this and should never be edited manually.

5.1.4.1 Division of functionality in steps and job entries

One of the ideas of Pentaho Data Integration is to make simple steps and job entries which have a single purpose, and be able to make complex transformations by linking them together, much like UNIX utilities.

Putting too much (diverse) functionality in 1 step/job entry will make them less intuitive for people to use, and since most people only start reading manuals when they get into problems we need all the intuitivity we can get.

5.1.4.2 Rows on a single hop have to be of the same structure

As in user section of this document: all rows that flow over a single hop have to be of the same structure. You shouldn't try to build things that try to circumvent this, which will be harder as of v2.5.0 because of the design time check on the structure of the rows.

5.1.4.3 Null and “” are the same in PDI

As in Oracle the empty string “” and NULL should be considered the same by all steps. This is to be in line with the rest with PDI.

5.1.4.4 On converting data to fit the corresponding Metadata

Length & Precision are just metadata pieces.
If you want to round to the specified precision, you should do this in another step. However: please keep in mind that rounding double point precision values is futile anyway. A floating point number is stored as an approximation (it floats) so 0.1261 (your desired output) could (would probably) end up being stored as 0.126099999999 or 0.1261000000001 (Note: this is not the case for BigNumbers)
So in the end we round using BigDecimals once we store the numbers in the output table, but NOT during the transformation. The same is true for the Text File Output step. If you would have specified Integer as result type, the internal number format would have been retained, you would press "Get Fields" and it the required Integer type would be filled in. The required conversion would take place there and then.
In short: we convert to the required metadata type when we land the data somewhere, NOT BEFORE.
The strategy of keeping the same datatype as long as possible has saved us from many a conversion error like the one described above.

5.2 Au sujet de la journalisation dans Pentaho Data Integration

Q: How to use logging in PDI steps? This applies to the main PDI steps as to any PDI steps you develop yourself.

A: All detailed, debug, and rowlevel statements needs to be preceded by the following:

```
if (log.isDetailed())
    ...
if (log.isDebug())
    ...
if (log.isRowlevel())
    ...
```

That is because otherwise, the string calculation of what you send to the log is always calculated. For Basic and Minimal logging levels this doesn't matter as normally they would always be “on”, but it does for Debug and Rowlevel.

5.3 Au sujet de l'utilisation de XML dans Pentaho Data Integration

Q: What's the idea of using XML in PDI?

A: XML is for machines, not for humans. So unless the functionality is in fact on the processing of XML itself (XML input step/XML output step) the XML should be kept hidden. Behind the screens XML will be used but users of PDI should not to be required to know this or manipulate XML in any way. Every XML configuration/setup file should be managed through a GUI element in PDI.

On dropdown boxes and storing values

Don't use the index of the dropdown box as a value in the XML export file or database repository.

Suppose you have currently 3 possible values in a dropdown box. And if someone chooses the first value you put “1” in the XML export file to indicate value 1. This would work fine, except that:

- if someone wants to add extra values in the future he must use the order you defined first;
- it makes the XML output very much unreadable.

It's better to convert from a Locale string in the GUI to some English equivalent which is then stored. As example:

- Suppose on the GUI you have a dropdown box with values “Date mask” and “Date time mask”;
- Instead of using a 1 in the output for “Date mask” and 2 for “Date time mask”, it would be better to put in the output “DATE_MASK” for “Date mask” and “DATE_TIME_MASK” for “Date time mask”;

- Also note that DATE_MASK/DATE_TIME_MASK would then not be allowed to be subject to I18N translation (which is ok for transformation/job files).

5.4 Au sujet de l'internationalisation (i18N) dans PDI

Q: Some more details on using I18N

A:

- Only translate what a normal user will see, it doesn't make sense to translate all debug message in PDI. Some performance improvements were achieved in PDI just by removing some of translations for debug messages;
- Make sure you don't translate strings used in the control logic of PDI:
    - If you would e.g. make the default name of a new step “language dependent” this would still make jobs/transformations usable across different locales;
    - If you would e.g. make tags used in the XML generated for the step language dependent there would be a problem when a user would switch his locale;
    - If you would translate non-tag strings used in the control logic you will also have a problem. E.g. in the repository manager “Administrator” is used to indicate which user is administrator (and this is used in the PDI control logic). So if you would translate administrator to a certain language, this would work as long as you wouldn't switch locales.

On using Locale's in PDI

PDI should always use the default Locale, so the Locale should not be hardcoded somewhere to English or so. However some steps may choose to be able to override the default Locale but this is then step specific and it should always be possible to select the Locale via the GUI of the step.

5.5 Au sujet du relookage du code source

Try to keep reformatting code to a minimum, especially on things like {'s at the end of the line or at the start of the next line, not all people like the same and why should your specific preference be used.

When changing code try to use the same formatting as the surrounding code, even if it's not your preference.

If you really feel a need for some reformatting do it in a separate SVN check-in, **DON'T** mix reformatting source code with real changes. It's **VERY** annoying not being able to easily see what changed because someone decided to reformat source code at the same time. “My tool does automatic formatting” is a very lame excuse as all known IDEs allow to switch it off.

5.6 Au sujet de l'utilisation du système de versionnement du code (Subversion)

1. Always make sure that whatever you put in SVN keeps PDI buildable and startable. Nothing is more annoying as not being able to even start PDI as someone checked in half-working code. If the change is too big to do at once, work by making small steps towards the full change (but at all times keeping PDI buildable/runnable).

How do you start developing your own plug-in step

Q: I need some complex calculation in my ETL. I have created own logic in java classes. How can I make my own step and integrate this in PDI?

A: You can start using the DummyPlugin as example. You can also have a look at how some other steps are implemented. In essence you need 4 classes to make a new step implementing the corresponding interfaces:

- StepMetaInterface: contains the meta-data;
- StepInterface: performs the actual work, implements logic;
- StepDialogInterface: pops up a dialog in Spoon;
- StepData: contains temporary data like ResultSets, file handels, input streams, etc.

  Can you change the XML input step to process my file?

Q: I have to process an XML file which currently can't be processed by KETTLE, e.g. there's one optional field which depends on the value of an element and that should also be included as a field in a row, ... Can you build this functionality in in the XML input step?

A: First of all it would depend what functionality you need. If the functionality is generally useful it can be built in. If it would only be useful for you it wouldn't make sense to build it in.

As alternative solutions: consider processing the XML file via a Javascript step, or if what is required is very complex consider writing your own PDI step which you maintain yourself (outside of the PDI distribution).

On Serializable and Binary

Q: If I need to select a type I can choose between Serializable and Binary. Are they the same, or what's the difference?

A: Binary is used for images, sounds, GIS data, BLOBs, ... You can read e.g. a database BLOB from one database and insert it in another database. Binary uses getByte() and setByte() to get and set its value.

Serializable is used for some proprietary plugins build by a company using it to pass Java objects from one step to another. Unless you're developing your own steps/plugins Serializable is not something to be used. The way to read/write data depends on the objects being stored in a Serializable.

6 LES FACTEURS DE SUCCÈS DE PDI

6.1 Modular design

Pentaho Data Integration as a runtime engine consists of a “row engine” taking care of transporting data from 1 step to the next. The steps are separate and can even use a plug-in architecture.