

	Service Développement Programme SIERA <i>Projet Pyrénées</i> Application ARC Préparation et Rapport des tests de performance	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 1 sur 44

Historique des versions successives

n°	Date	Auteur	Modification
0.1	17/10/2016	RP	Version initiale
0.2	06/04/2017	RP	Mise en page+modif exemple indépendance
0,3	25/09/2017	RP	Correction suite aux tests de G.Fourmont

Table des matières

1 Généralités.....	2
2 La phase de chargement.....	4
2.2 Le chargement d'un fichier.....	4
3 La phase de normage.....	26
4 La phase de contrôle.....	35
5 La phase de filtrage.....	38
6 La phase de mapping.....	39

	<p>Service Développement Programme SIERA <i>Projet Pyrénées</i></p> <p style="text-align: center;">Application ARC Préparation et Rapport des tests de performance</p>		
Rédacteur(s) : R.PEPIN	Documentation technique	Page 2 sur 44	

1 Généralités

1.1 Utilisation des types composites et parallélisation des traitements

Suite à un audit de performances sur ARC, il a été décidé d'implémenter l'utilisation de type composite les bases de données de ARC. Pour ne pas provoquer un changement trop grand dans le fonctionnement de l'application les types composites ne seront utilisés que pour le stockage des fichiers en fin de phase, et pour la création de tables de travail en début de phase.

Ainsi la création d'une table n'est plus une simple copie de table, mais une procédure plus complexe dont voici un exemple.

	Service Développement Programme SIERA <i>Projet Pyrénées</i> Application ARC Préparation et Rapport des tests de performance	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 3 sur 44

table chargement :

id_source (text)	id (integer)	date_intégration (text)	data (text)
fichier1	1	JJMMAAAA	(1,1,XXX,1,YYY...)
fichier1	2	JJMMAAAA	(1,2,AAA,2,BBB..)
fichier2	1	JJMMAAAA	(1,C,1,1,D...)
fichier3	1	JJMMAAAA	(1,1,1,1,EEE,1,FFF...)

Etape 1 : on sélection les lignes liées à un id source

table_pilotage

id_source	...	generation_composite
fichier1		CREATE TYPE composite AS ...
fichier2		CREATE TYPE composite AS ...
fichier3		CREATE TYPE composite AS ...

Etape 2 : on exécute la requête de création du type lié à l'id_srouce

table_travail_temp



id_source (text)	id (integer)	date_intégration (text)	data (composite)
fichier1	1	JJMMAAAA	(1,1,XXX,1,YYY...)
fichier1	2	JJMMAAAA	(1,2,AAA,2,BBB..)

Etape 3 : les données sont castées grâce au type composite du fichier1

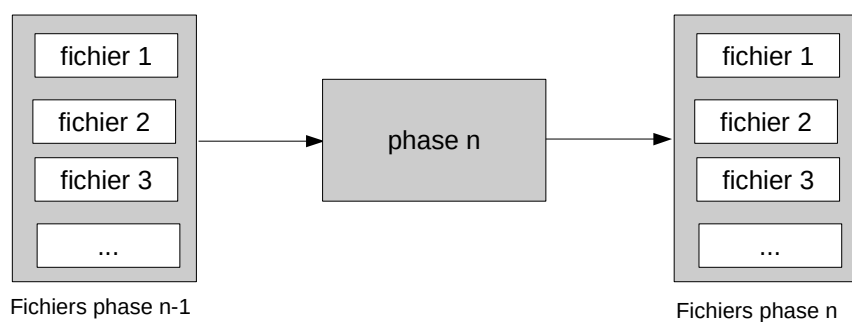
Etape 4 : on crée la table travail sans type composite en extrayant du type composite le nom des colonnes et leurs valeurs.

table_travail

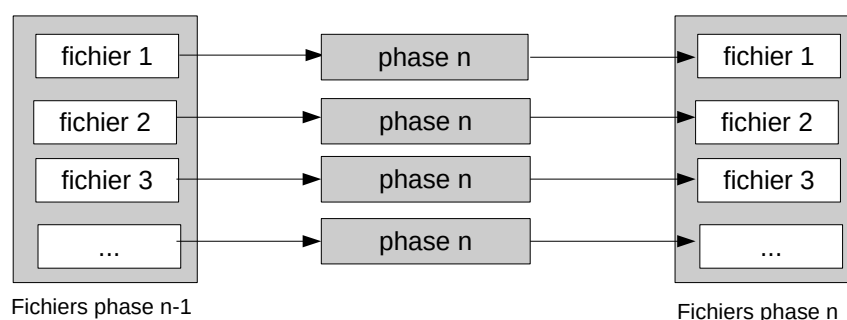
id_source (text)	id (integer)	date_intégration (text)	col1	col2	col3	col4	...
fichier1	1	JJMMAAAA	1	1	XXX	1	...
fichier1	2	JJMMAAAA	1	2	AAA	2	...



	Service Développement Programme SIERA <i>Projet Pyrénées</i> Application ARC Préparation et Rapport des tests de performance	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 4 sur 44

Ainsi, au lieu de créer une table de travail globale nous en créons une par id source. Cela permet donc une parallélisation des traitements naturelles. Donc au lieu d'avoir un traitement de la forme



nous avons un traitement comme ceci :



	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 5 sur 44

2 La phase d'identification

2.1 La détection de la forme du fichier.

Dans le lot 2, le chargeur ARC devra pouvoir prendre en entrée différents types de fichiers (clef-valeur, plat, xml). Cela va demander une modification de la manière de charger les fichiers dans ARC. Ainsi, avant le chargement dans la base, une étape de détection de la norme doit être faite. C'est la norme qui nous donnera le chargeur à appliquer en fonction de ce que l'utilisateur aura rentré dans l'IHM gestion des normes.

La détection de la norme d'un fichier commence par la récupération des règles de chargement et de détection des normes, respectivement contenu dans les tables CHARGEMENT_REGLE et NORME. Ces règles ont été au préalable écrites par l'utilisateur via l'IHM de gestion des normes. Ensuite on va charger le fichier ligne par ligne en base et appliquer sur cette table les différentes requêtes de détection des normes jusqu'à avoir testé toutes les normes ou avoir deux normes valides. Si deux normes sont valides sur une même fichier ce fichier est mis KO, de même que si aucune norme valide n'est trouvée. Enfin si une seule norme est trouvée, on va regarder le type de fichier auquel elle est liée et utiliser le chargeur correspondant.

3 Le chargement d'un fichier


3.1 Le chargeur XML

3.1.1 La lecture du fichier

C'est le chargeur qui sera appliqué à un fichier qui est sous forme de fichier xml. Nous utilisons l'API SAX pour parser le fichier xml. Ce choix est dû au fait que les fichiers en entrée peuvent être très volumineux, il est donc plus optimal de les traiter via un flux de donnée comme le propose Sax.

Sax fonctionne par événement. Voici les événements implémentés dans le chargeur

- startDocument() : appelé au début du document. Initialise le type composite
- startElement() : lit le contenu de l'élément.
- characters() : lit les données de l'élément
- endElement() : ajout des données si l'élément en avait
- endDocument() : génère la requête de jointure

	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 6 sur 44


<u>Fichier 1</u>	<u>Fichier 2</u>
<pre> <racine> <tronc> <couleur> marron </couleur> <etat> bon </etat> <branche> <feuille> feuille1 </feuille> </branche> <branche> <feuille> feuille3 </feuille> </branche> <branche> <feuille> feuille4 </feuille> </branche> </tronc> <tronc> <couleur> noir </couleur> <etat> brûlé </etat> <branche> <feuille> feuille6 </feuille> </branche> <branche> <feuille> feuille7 </feuille> </branche> </tronc> <racine_attribut> <age> 45 </age> <taille> 15m </taille> </racine_attribut> </racine> </pre>	<pre> <racine> <tronc> <branche> <feuille> feuille1 </feuille> </branche> <branche> <feuille> feuille3 </feuille> </branche> <branche> <feuille> feuille4 </feuille> </branche> <couleur> marron </couleur> <etat> bon </etat> </tronc> <tronc> <branche> <feuille> feuille6 </feuille> </branche> <branche> <feuille> feuille7 </feuille> </branche> <couleur> noir </couleur> <etat> brûlé </etat> </tronc> <racine_attribut> <age> 45 </age> <taille> 15m </taille> </racine_attribut> </racine> </pre>

Traitements réalisés lors de la lecture


Étape 0 : Création d'une table temporaire *tableA* avec les colonnes *ids* (= nom du fichier), *id* (= index de la requête générée), *date* (date de l'insertion).

Étape 1 : Le fichier est lu en streaming élément par élément grâce à l'API SAX.


Avant même de lire le fichier, le parser SAX va déclencher la méthode startDocument. Ceci va avoir pour effet d'initialiser la requête de création du type composite.

	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 7 sur 44

<pre> <racine> <tronc> <couleur> marron </couleur> <etat> bon </etat> <branche> <feuille> feuille1 </feuille> </branche> <branche> <feuille> feuille3 </feuille> </branche> <branche> <feuille> feuille4 </feuille> </branche> </tronc> <tronc> <couleur> noir </couleur> <etat> brûlé </etat> <branche> <feuille> feuille6 </feuille> </branche> <branche> <feuille> feuille7 </feuille> </branche> </tronc> <racine_attribut> <age> 45 </age> <taille> 15m </taille> </racine_attribut> </racine> </pre>	<p>← Rencontre de l'élément racine → élément racine déjà rencontrée ? → non → ajout racine aux colonnes rencontrés (allCols = [racine], cols = [(racine, 1)])</p> <p><i>La variable allCols répertorie seulement les colonnes rencontrées, cols associe en plus le nombre de fois où on les a rencontrés. L'index associé à une colonne dans allCols est unique, ce qui permet une bijection index-nom colonne. Dans un souci de rapidité lors du chargement sera utilisé les index, puis une fois le fichier chargé on récupérera le vrai nom des colonnes</i></p> <p>→ On va ajouter une colonne i0 à la table.</p> <p>→ On ajoute un élément null à la liste lineValue : lineValue = [null]</p>
--	---

	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p style="text-align: center;">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 8 sur 44

<pre> <racine> <tronc> <couleur> marron </couleur> <etat> bon </etat> <branche> <feuille> feuille1 </feuille> </branche> <branche> <feuille> feuille3 </feuille> </branche> <branche> <feuille> feuille4 </feuille> </branche> </tronc> <tronc> <couleur> noir </couleur> <etat> brûlé </etat> <branche> <feuille> feuille6 </feuille> </branche> <branche> <feuille> feuille7 </feuille> </branche> </tronc> <racine_attribut> <age> 45 </age> <taille> 15m </taille> </racine_attribut> </racine> </pre>	<p>← Rencontre de l'élément tronc L'élément tronc déjà rencontrée ? → non → ajout de tronc aux colonnes rencontrées (allCols = [racine,tronc], cols = [(racine,1), (tronc,1)]) → On va ajouter une colonne i1 à la table → On ajoute un élément null à la liste lineValue : lineValue = [null,null]</p>
--	---

	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p style="text-align: center;">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 9 sur 44

<pre> <racine> <tronc> <couleur> marron </couleur> <etat> bon </etat> <branche> <feuille> feuille1 </feuille> </branche> <branche> <feuille> feuille3 </feuille> </branche> <branche> <feuille> feuille4 </feuille> </branche> </tronc> <tronc> <couleur> noir </couleur> <etat> brûlé </etat> <branche> <feuille> feuille6 </feuille> </branche> <branche> <feuille> feuille7 </feuille> </branche> </tronc> <racine_attribut> <age> 45 </age> <taille> 15m </taille> </racine_attribut> </racine> </pre>	<p>← Rencontre de l'élément couleur L'élément couleur déjà rencontrée ?</p> <ul style="list-style-type: none"> → non → ajout de « couleur » aux colonnes rencontrées (allCols = [racine,tronc,couleur], cols = [(racine,1), (tronc,1), (couleur,1)]) → On va ajouter une colonne i2 à la table → On ajoute un élément null à la liste lineValue : lineValue = [null,null,null]
--	--

	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	 <i>Pirénés</i>
Rédacteur(s) : R.PEPIN	Documentation technique	Page 10 sur 44

<pre> <racine> <tronc> <couleur> marron </couleur> <etat> bon </etat> <branche> <feuille> feuille1 </feuille> </branche> <branche> <feuille> feuille3 </feuille> </branche> <branche> <feuille> feuille4 </feuille> </branche> </tronc> <tronc> <couleur> noir </couleur> <etat> brûlé </etat> <branche> <feuille> feuille6 </feuille> </branche> <branche> <feuille> feuille7 </feuille> </branche> </tronc> <racine_attribut> <age> 45 </age> <taille> 15m </taille> </racine_attribut> </racine> </pre>	<p>← Rencontre de la valeur de l'élément couleur → Ajout de « marron » à <i>currentData</i> → <i>currentData</i> = marron</p>
--	---

	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 11 sur 44


<pre> <racine> <tronc> <couleur> marron </couleur> <etat> bon </etat> <branche> <feuille> feuille1 </feuille> </branche> <branche> <feuille> feuille3 </feuille> </branche> <branche> <feuille> feuille4 </feuille> </branche> </tronc> <tronc> <couleur> noir </couleur> <etat> brûlé </etat> <branche> <feuille> feuille6 </feuille> </branche> <branche> <feuille> feuille7 </feuille> </branche> </tronc> <racine_attribut> <age> 45 </age> <taille> 15m </taille> </racine_attribut> </racine> </pre>	<p>← rencontre de la fin de l'élément couleur. → L'élément contient une valeur ? → oui → ajout d'une colonne v2 à la table → suppression du dernier élément de <i>lineValue</i>, ajout de la valeur de <i>currentData</i> à la liste → <i>currentData</i> = [null, null, marron]</p>
--	--

	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 12 sur 44


<pre> <racine> <tronc> <couleur> marron </couleur> <etat> bon </etat> <branche> <feuille> feuille1 </feuille> </branche> <branche> <feuille> feuille3 </feuille> </branche> <branche> <feuille> feuille4 </feuille> </branche> </tronc> <tronc> <couleur> noir </couleur> <etat> brûlé </etat> <branche> <feuille> feuille6 </feuille> </branche> <branche> <feuille> feuille7 </feuille> </branche> </tronc> <racine_attribut> <age> 45 </age> <taille> 15m </taille> </racine_attribut> </racine> </pre>	<p>← Rencontre de l'élément « etat » → on vide <i>currentData</i> L'élément « etat » déjà rencontrée ? → non → ajout de « couleur » aux colonnes rencontrés (allCols = [racine,tronc,couleur,etat], cols = [(racine,1), (tronc,1), (couleur,1), (etat,1)]) → On va ajouter une colonne i3 à la table → On ajoute un élément null à la liste lineValue : lineValue = [null,null,marron,null]</p> <p>* Rencontre de la valeur de l'élément « etat » → Ajout de « bon » aux valeurs rencontrées → <i>currentData</i> = bon</p> <p>* Rencontre de la fin de l'élément « etat ». → la élément contient une valeur? → oui → ajout d'une colonne v3 à la table → suppression du dernier élément de <i>lineValue</i>, ajout de la valeur de <i>currentData</i> à la liste → <i>linevalue</i> = [null, null, marron,bon]</p>
--	---

	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	 <i>Pirénés</i>
Rédacteur(s) : R.PEPIN	Documentation technique	Page 13 sur 44


<pre> <racine> <tronc> <couleur> marron </couleur> <etat> bon </etat> <branche> <feuille> feuille1 </feuille> </branche> <branche> <feuille> feuille3 </feuille> </branche> <branche> <feuille> feuille4 </feuille> </branche> </tronc> <tronc> <couleur> noir </couleur> <etat> brûlé </etat> <branche> <feuille> feuille6 </feuille> </branche> <branche> <feuille> feuille7 </feuille> </branche> </tronc> <racine_attribut> <age> 45 </age> <taille> 15m </taille> </racine_attribut> </racine> </pre>	<p>← Rencontre de l'élément « branche » → on vide <i>currentData</i> L'élément « branche » déjà rencontrée ? → non → ajout de « branche » aux colonnes rencontrés allCols = [racine,tronc,couleur,etat, branche], cols = [(racine,1), (tronc,1), (couleur,1), (etat,1), (branche,1)] → On va ajouter une colonne i4 à la table → On ajoute un élément null à la liste lineValue : lineValue = [null,null,marron,bon,null]</p>
--	---

	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 14 sur 44


<pre> <racine> <tronc> <couleur> marron </couleur> <etat> bon </etat> <branche> <feuille> feuille1 </feuille> </branche> <branche> <feuille> feuille3 </feuille> </branche> <branche> <feuille> feuille4 </feuille> </branche> </tronc> <tronc> <couleur> noir </couleur> <etat> brûlé </etat> <branche> <feuille> feuille6 </feuille> </branche> <branche> <feuille> feuille7 </feuille> </branche> </tronc> <racine_attribut> <age> 45 </age> <taille> 15m </taille> </racine_attribut> </racine> </pre>	<p>← Rencontre de l'élément « feuille » L'élément feuille déjà rencontrée ? → non → ajout de « feuille » aux colonnes rencontrés allCols = [racine,tronc,couleur,etat,branche, feuille], cols = [(racine,1), (tronc,1), (couleur,1), (etat,1), (branche,1), (feuille,1)] → On va ajouter une colonne i5 à la table → On ajoute un élément null à la liste lineValue : lineValue = [null,null,marron,null]</p> <p>* Rencontre de la valeur de l'élément « feuille » → Ajout de « feuille1 » aux valeurs rencontrées → <i>currentData</i> = feuille1</p> <p>* Rencontre de la fin de l'élément « feuille » → la élément contient une valeur? → oui → ajout d'une colonne v5 à la table → On ajoute un élément null à la liste lineValue : lineValue = [null,null,marron,bon,null,feuille1]</p>
--	--

	<p>Service Développement Programme SIERA <i>Projet Pyrénées</i></p> <p style="text-align: center;">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 15 sur 44

<pre> <racine> <tronc> <couleur> marron </couleur> <etat> bon </etat> <branche> <feuille> feuille1 </feuille> </branche> <branche> <feuille> feuille3 </feuille> </branche> <branche> <feuille> feuille4 </feuille> </branche> </tronc> <tronc> <couleur> noir </couleur> <etat> brûlé </etat> <branche> <feuille> feuille6 </feuille> </branche> <branche> <feuille> feuille7 </feuille> </branche> </tronc> <racine_attribut> <age> 45 </age> <taille> 15m </taille> </racine_attribut> </racine> </pre>	<p>← Rencontre de la fin de l'élément « branche » → l'élément contient une valeur? → non</p> <p>* Rencontre de l'élément branche L'élément branche déjà rencontrée ? → oui → incrémentation de nombre d'éléments « branche » rencontrés cols = [(racine,1), (tronc,1), (couleur,1), (etat,1), (branche,2), (feuille,1)] → On vient de fermer l'élément branche et on en ouvre une de suite après. Génération d'une requête d'insertion de la forme INSERT INTO maTable (ids, id,i0, i1, i2, v2, i3, v3, i4, i5, v5) VALUES ('fic', 1,1, 1, 1, 'marron', 1, 'bon', 1, 1, 'feuille1')</p> <p>→ On vide lineValue jusqu'à l'indice suivant de l'élément qui a déclenché l'insertion. Ici l'élément est branche. Il a pour indice 4, on va conserver les 5 premiers objets des listes. Ainsi on obtient lineValue = [null,null,marron,bon,null]</p>
--	---

	<p>Service Développement Programme SIERA <i>Projet Pyrénées</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 16 sur 44



<pre> <racine> <tronc> <couleur> marron </couleur> <etat> bon </etat> <branche> <feuille> feuille1 </feuille> </branche> <branche> <feuille> feuille3 </feuille> <branche> <branche> <feuille> feuille4 </feuille> </branche> </tronc> <tronc> <couleur> noir </couleur> <etat> brûlé </etat> <branche> <feuille> feuille6 </feuille> </branche> <branche> <feuille> feuille7 </feuille> </branche> </tronc> <racine_attribut> <age> 45 </age> <taille> 15m </taille> </racine_attribut> </racine> </pre>	<p>→ L'élément « feuille » déjà rencontrée ? → oui → incrémentation de nombre d'éléments « feuille » rencontrés cols = [(racine,1), (tronc,1), (couleur,1), (etat,1), (branche,2), (feuille,2)] → On ajoute un élément null à la liste lineValue : lineValue = [null,null,marron,bon,null,null]</p> <p>* Rencontre de la valeur de l'élément « feuille » → Ajout de « feuille3 » aux données rencontrées → <i>currentData</i> = feuille3</p> <p>* Rencontre de la fin de l'élément « feuille » → l'élément contient une valeur ? → oui → la colonne existe déjà dans allCols, on ne rajoute pas une colonne à la table → suppression du dernier élément de <i>lineValue</i>, ajout de la valeur de <i>currentData</i> à la liste → <i>linevalue</i> = [null, null, marron,bon, null, feuille3]</p> <p>* Rencontre de la fin de l'élément « branche » → l'élément contient une valeur ? → non</p> <p>* Rencontre de l'élément « branche » L'élément « branche » déjà rencontrée ? → oui → incrémentation de nombre d'éléments</p>
---	--

	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 17 sur 44

	<p>« branche » rencontrés cols = [(racine,1), (tronc,1), (couleur,1), (etat,1), (branche,3), (feuille,2)] → On vient de fermer un élément « branche » et on en ouvre une de suite après. Génération d'une requête d'insertion de la forme INSERT INTO maTable (ids, id, i0, i1, i2, v2, i3, v3, i4, i5, v5) VALUES ('fic', 2,1, 1, 1, 'marron', 1, 'bon', 1, 2, 2, 'feuille3')</p> <p>→ On vide lineValue jusqu'à l'indice suivant de l'élément qui a déclenché l'insertion. Ici l'élément est branche. Il a pour indice 4, on va conserver les 5 premiers objets des listes Ainsi on obtient lineValue = [null,null,marron,bon,null]</p>
--	---

	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p style="text-align: center;">Application ARC Préparation et Rapport des tests de performance</p>	 <i>Pirénés</i>
Rédacteur(s) : R.PEPIN	Documentation technique	Page 18 sur 44

<pre> <racine> <tronc> <couleur> marron </couleur> <etat> bon </etat> <branche> <feuille> feuille1 </feuille> </branche> <branche> <feuille> feuille3 </feuille> </branche> <branche> <feuille> feuille4 </feuille> </branche> </tronc> <tronc> <couleur> noir </couleur> <etat> brûlé </etat> <branche> <feuille> feuille6 </feuille> </branche> <branche> <feuille> feuille7 </feuille> </branche> </tronc> <racine_attribut> <age> 45 </age> <taille> 15m </taille> </racine_attribut> </racine> </pre>	<p>→ de la même façon que précédent suite à ces éléments une requête d'insertion va être générée INSERT INTO maTable (ids, id, i0, i1, i2, v2, i3, v3, i4, i5, v5) VALUES ('fic', 1,1, 1, 1, 'marron', 1, 'bon', 1 3, 3, 'feuille4')</p> <p>→ On vide lineValue jusqu'à l'indice suivant de l'élément qui a déclenché l'insertion. Ici l'élément est tronc. Il a pour indice 1, on va conserver les 2 premiers objets des listes Ainsi on obtient lineValue = [null,null]</p>
--	--

	<p>Service Développement Programme SIERA <i>Projet Pyrénées</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 19 sur 44

Base de donnée après cette étape


ids	id	i0	i1	i2	v2	i3	v3	i4	i5	v5
fic	3	1	1	1	marron	1	bon	3	3	feuille4
fic	2	1	1	1	marron	1	bon	2	2	feuille3
fic	1	1	1	1	marron	1	bon	1	1	feuille1

	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 20 sur 44

<pre> <racine> <tronc> <couleur> marron </couleur> <etat> bon </etat> <branche> <feuille> feuille1 </feuille> </branche> <branche> <feuille> feuille3 </feuille> </branche> <branche> <feuille> feuille4 </feuille> </branche> </tronc> <tronc> <couleur> noir </couleur> <etat> brûlé </etat> <branche> <feuille> feuille6 </feuille> </branche> <branche> <feuille> feuille7 </feuille> </branche> </tronc> <racine_attribut> <age> 45 </age> <taille> 15m </taille> </racine_attribut> </racine> </pre>	<p>Comme précédemment on va ajouter les valeurs lues dans la base. Aucune nouvelle élément n'est rencontré alors il n'y a pas d'insertion de nouvelle colonne.</p>
--	--


Base de données après cette étape

ids	id	i0	i1	i2	v2	i3	v3	i4	i5	v5
fic	5	1	2	2	noir	2	brûlé	5	5	feuille7
fic	4	1	2	2	noir	2	brûlé	4	4	feuille6
fic	3	1	1	1	marron	1	bon	3	3	feuille4
fic	2	1	1	1	marron	1	bon	2	2	feuille3
fic	1	1	1	1	marron	1	bon	1	1	feuille1

	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 21 sur 44

<pre> <racine> <tronc> <couleur> marron </couleur> <etat> bon </etat> <branche> <feuille> feuille1 </feuille> </branche> <branche> <feuille> feuille3 </feuille> </branche> <branche> <feuille> feuille4 </feuille> </branche> </tronc> <tronc> <couleur> noir </couleur> <etat> brûlé </etat> <branche> <feuille> feuille6 </feuille> </branche> <branche> <feuille> feuille7 </feuille> </branche> </tronc> <racine_attribut> <age> 45 </age> <taille> 15m </taille> </racine_attribut> </racine> </pre>	<p>3 nouveaux éléments sont rencontrés, dont 2 contenant des valeurs. On ajoute ainsi 5 colonnes à la table et une ligne supplémentaire. De plus à la suite de ça le fichier est terminé.</p>
--	---

Lors de la fin du document SAX exécute la méthode endDocument. Elle va avoir pour effet d'insérer les dernières lignes en mémoire et de générer le début de la requête de jointure utilisée dans la phase de normage.

	Service Développement Programme SIERA <i>Projet Pyrénées</i> Application ARC Préparation et Rapport des tests de performance	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 22 sur 44


Base finale de chargement

ids	id	i0	i1	i2	v2	i3	v3	i4	i5	v5	i6	i7	v7	i8	v8
fic	5	1	2	2	noir	2	brûlé	5	5	feuille7					
fic	4	1	2	2	noir	2	brûlé	4	4	feuille6					
fic	3	1	1	1	marron	1	bon	3	3	feuille4					
fic	2	1	1	1	marron	1	bon	2	2	feuille3					
fic	1	1	1	1	marron	1	bon	1	1	feuille1					
fic	6										1	1	45	1	15m

En utilisant la même logique que précédemment sur le deuxième fichier xml Fichier 2 le résultat obtenu en base de données est néanmoins différent de celui obtenu avec le Fichier 1

ids	id	i0	i1	i4	i5	v5	i2	v2	i3	v3	i6	i7	v7	i8	v8
fic	6	1	2	5	5	feuille7									
fic	5	1	2	4	4	feuille6									
fic	3	1	1	3	3	feuille4									
fic	2	1	1	2	2	feuille3									
fic	1	1	1	1	1	feuille1									
fic	7	1	2				2	noir	2	brûlé					
fic	4	1	1				1	marron	1	bon					
fic	8	1									1	1	45	1	15m

La structure du fichier xml influe donc sur le résultat obtenu en base de données à l'issue du chargement. Afin de découpler le résultat obtenu en base de données et le format initial du fichier, il est nécessaire d'ajouter une opération à effectuer à l'issue du chargement : le normage. Cette opération va permettre de réduire le nombre de ligne en base.

	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p>Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 23 sur 44

3.1.2 Le chargeur clef-valeur

Le but de ce chargeur va être de transformer un fichier clef-valeur en un fichier xml. Pour ce faire nous avons besoin d'un fichier xml contenant la hiérarchie du fichier cible. Ce fichier à seulement besoin de contenir les noms des rubriques du fichier. Voici un exemple de fichier xml format attendu :


```

<racine>
  <tronc>
    <couleur> </couleur>
    <etat> </etat>
    <branche>
      <feuille> </feuille>
    </branche>
  </tronc>
  <racine_attribut>
    <age> </age>
    <taille> </taille>
  </racine_attribut>
</racine>

```



Ce fichier va être lu et sa structure enregistrer. Ensuite en lisant le fichier clef-valeur nous allons créer le fichier xml correspondant qui sera ensuite chargé par le lecteur xml.

couleur, marron etat, bon feuille, feuille1 feuille, feuille3 feuille, feuille4 couleur, noir etat, brûlé feuille, feuille6 feuille, feuille7 age, 45 taille, 15m	
---	--

	<p>Service Développement Programme SIERA <i>Projet Pyrénées</i></p> <p style="text-align: center;">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 24 sur 44

couleur, marron etat, bon feuille, feuille1 feuille, feuille3 feuille, feuille4 couleur, noir etat, brûlé feuille, feuille6 feuille, feuille7 age, 45 taille, 15m	<pre> <racine> <tronc> <couleur> marron </couleur> </pre>
--	---

couleur, marron etat, bon feuille, feuille1 feuille, feuille3 feuille, feuille4 couleur, noir etat, brûlé feuille, feuille6 feuille, feuille7 age, 45 taille, 15m	<pre> <racine> <tronc> <couleur>marron</couleur> <etat> bon</etat> </pre>
---	---

	Service Développement Programme SIERA <i>Projet Pyrénées</i> Application ARC Préparation et Rapport des tests de performance	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 25 sur 44

couleur, marron etat, bon feuille, feuille1 feuille, feuille3 feuille, feuille4 couleur, noir etat, brûlé feuille, feuille6 feuille, feuille7 age, 45 taille, 15m	<pre> <racine> <tronc> <couleur>marron</couleur> <etat> bon</etat> <branche> <feuille>feuille1</feuille> </pre>
--	---

couleur, marron etat, bon feuille, feuille1 feuille, feuille3 feuille, feuille4 couleur, noir etat, brûlé feuille, feuille6 feuille, feuille7 age, 45 taille, 15m	<pre> <racine> <tronc> <couleur>marron</couleur> <etat> bon</etat> <branche> <feuille>feuille1</feuille> </branche> <branche> <feuille>feuille3</feuille> </pre>
---	--

Et ainsi de suite. Ce choix de conversion vient du fait que le chargeur xml est déjà recetté et que le gros traitement réaliser lors de la création de la jointure xml n'est pas à redévelopper.

3.1.3 Le chargeur CSV

Le but de ce chargeur va être de transformer un fichier CSV (comma separator value). Les fichiers Grand Format et PE seront dans ce format par exemple. Ici on ne passera pas dans le xml, pour des gains de performance. La commande COPY sera utilisé pour également gagner du temps.

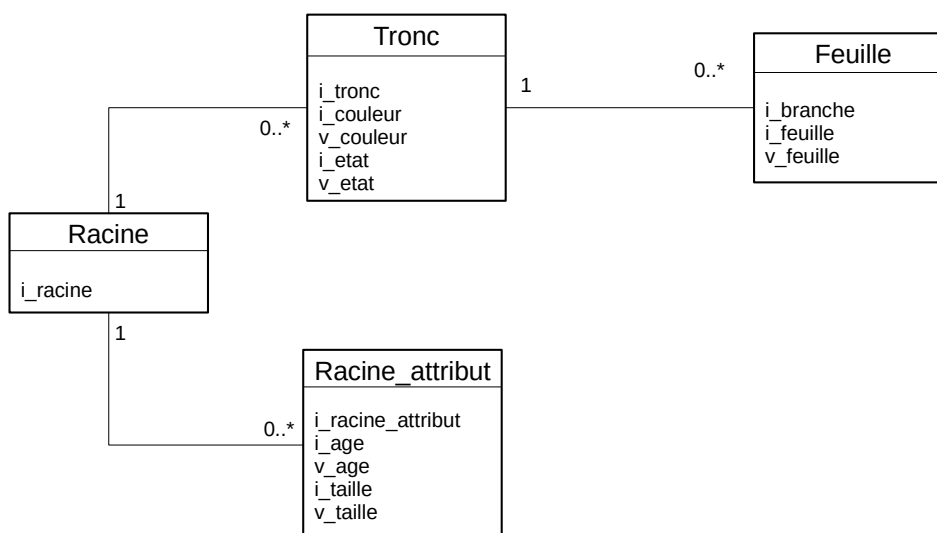
	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p style="text-align: center;">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 26 sur 44

3.1.4 Requête de jointure

En fin de chargement, on génère une première version de la requête de jointure utilisée lors du normage. Elle pose les premières briques nécessaires à la jointure : la création des différentes tables temporaires nécessaires, chacune lié à un bloc du fichier, et l'ordre des jointures à faire pour créer des lignes « complètes ». Cela revient à faire un diagramme de classe avec les blocs du fichier, et ensuite partir du plus haut élément et de faire des jointures de proche en proche pour obtenir un fichier « aplatis »

Exemple étape par étape de la requête de jointure :

Voici le diagramme de classe du fichier xml de l'exemple.



Voilà les tables créées

table racine

i_racine
1

table tronc

i_tronc	i_racine	i_couleur	v_couleur	i_etat	v_etat
1	1	1	marron	1	bon
2	1	2	noir	2	brûlé



table racine_attribut

i_racine_attribut	i_racine	i_age	v_age	i_taille	v_taille
1	1	1	45	1	15m

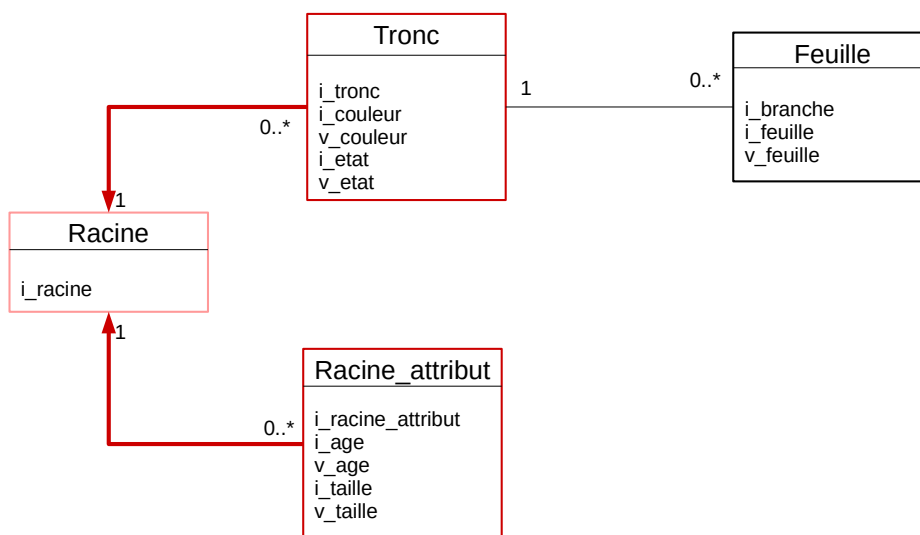
table feuille

ibranche	i_tronc	i_feuille	v_feuille
1	1	1	feuille1
2	1	2	feuille3
3	1	3	feuille4
4	2	4	feuille5
5	2	5	feuille6

On va ensuite joindre les tables en suivant la hiérarchie de l'arbre.

	Service Développement Programme SIERA <i>Projet Pyrénées</i> Application ARC Préparation et Rapport des tests de performance	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 28 sur 44

Ainsi dans un premier temps ce sont les tables Tronc et Racine_attribut qui vont être jointes à Racine.

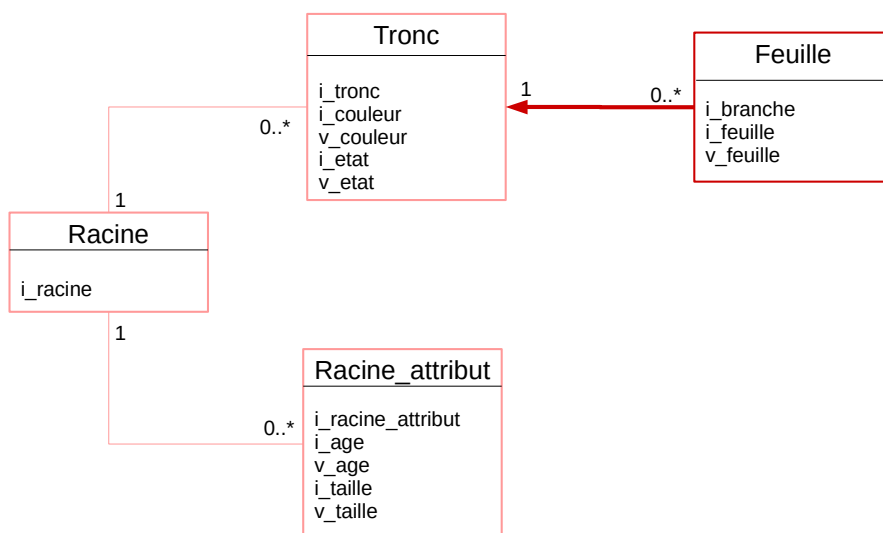


La table obtenue est la suivante

i_racine	i_tronc	i_couleur	v_couleur	i_etat	v_etat	i_racine_attribut	i_age	v_age	i_taille	v_taille
1	1	1	marron	1	bon	1	1	15	1	45m
1	2	2	noir	2	brûlé	1	1	15	1	45m

	Service Développement Programme SIERA <i>Projet Pyrénées</i> Application ARC Préparation et Rapport des tests de performance	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 29 sur 44

On joint ensuite la table Feuille à cette table intermédiaire



La table finale obtenue est la suivante

i_racine	i_tronc	i_couleur	v_couleur	i_etat	v_etat	i_racine_attribut	i_age	v_age	i_taille	v_taille	i_branche	i_feuille	v_feuille
1	1	1	marron	1	bon	1	1	15	1	45m	1	1	feuille1
1	1	1	marron	1	bon	1	1	15	1	45m	2	2	feuille3
1	1	1	marron	1	bon	1	1	15	1	45m	3	3	feuille4
1	2	2	noir	2	brûlé	1	1	15	1	45m	4	4	feuille5
1	2	2	noir	2	brûlé	1	1	15	1	45m	5	5	feuille7

En divisant notre base de chargement en différentes table déterminées par la hiérarchie de notre fichier xml puis en les joignant en suivant l'ordre hiérarchique on arrive ainsi à obtenir une table finale avec des lignes complètes et indépendantes de la forme du fichier initiale.

	<p>Service Développement Programme SIERA <i>Projet Pyrénées</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 30 sur 44

4 La phase de transformation

La phase de transformation a pour but d'aplatir les données en sortie de chargement ainsi que de lui appliquer des règles pour obtenir une table ayant un sens métier. Cela va se faire par l'exécution d'une requête sql de jointure entre les différents blocs du fichier. Cette requête sera créée à partir de la requête issue du chargement. Pour la suite on va considérer la requête de jointure comme divisée en deux parties. La première contient l'intégralité des requêtes de création des tables liées aux différents blocs, la seconde contient la requête de jointure entre les tables. On parlera de requête de jointure pour la requête en son ensemble et de partie de création des tables ou de partie de jointure pour les différentes parties.


Les règles de transformation utilisées dans cette phase sont celle rentrées dans l'onglet «transformation» de l'IHM « gestion des normes » et des règles générées automatiquement par l'application en fonction des règles de l'IHM et de la forme du fichier. Elles sont de 6 types, *suppression, relation, cartesian, unicité* (règles IHM), *duplication* et *indépendance* (règles générées).

La phase de transformation débute par la récupération des id_source issus de la phase de chargement. Elle va ensuite créer un pool de **connection** et un pool de **threadNormageService** et normer les fichiers en parallèle.

La transformation d'un fichier commence par la création d'une table de transformation temporaire contenant les lignes de CHARGEMENT_TODO avec l'id_source du thread, puis la création de tables de transformation ok et ko temporaire que l'on remplira si la transformation se déroule correctement ou non.

Ensuite on va éditer la requête de jointure des blocs du fichier. Cela commence par la récupération de l'intégralité des règles utilisées dans les différentes phases de Arc. Ces règles sont contenues dans les tables : TRANSFORMATION_REGLE, CONTROLE_REGLE, FILTRAGE_REGLE, MAPPING_REGLE. On récupère ensuite spécifiquement les règles de transformation. Une règle de transformation est caractérisé par 2 variables : son type, et les rubriques qu'elle touche. On va récupérer également la « la carte d'identité » du fichier (id_source, id_norme, validité, périodicité, jointure) contenu dans PILOTAGE_FICHIER. Grâce à cela, on va éliminer des règles de normage celles qui ne correspondent pas à la norme du fichier.

Si le fichier ne possède pas de requête de jointure, on va l'insérer tel quel dans la table temporaire de transformation ok. Sinon on va appliquer les règles de transformation sur la requête de sortie du transformation. **L'ordre d'application des règles n'est pas modifiable !**

	Service Développement Programme SIERA Projet Pyrénées Application ARC Préparation et Rapport des tests de performance		
Rédacteur(s) : R.PEPIN	Documentation technique	Page 31 sur 44	

Les premières règles à être appliquées sont les *règles de suppression*. Elles ont pour but de supprimer des rubriques considérées comme inutiles pour le reste des traitements par les experts. Avant de modifier la requête de jointure on vérifie si aucune des règles de suppression ne concerne une rubrique utilisée dans une autre règle de l'application Arc. Si c'est le cas, une exception est levée signalant la rubrique concernée et le transformation prend fin et le fichier est mis en ko. Sinon, on va appliquer les règles en supprimant de la requête les rubriques à supprimer ainsi que leurs filles, si elles en ont. Pour ce faire on va parcourir la requête de jointure et y retirer toutes les occurrences des rubriques et de leurs filles, aussi bien dans la partie de création des tables que dans la partie de jointure.

Le second type de règle à être appliquée est la *duplication*. Elle a pour but de dupliquer un bloc pour lui donner différent sens métier. En effet, en fonction où le bloc se situe, il n'a pas forcément le même sens métier. Par exemple, le bloc S21_G00_85 représente soit un lieu de travail, soit un établissement utilisateur. Dans le cas d'une femme de ménage cela va représenter aussi bien l'entreprise dont elle dépend (établissement utilisateur) et les endroits où elle va faire ses ménages (lieux de travail). Sans duplication de la rubrique S21_G00_85 il va être impossible de faire une jointure en différenciant ces informations.

Ce type n'est pas dans la liste de choix des types dans l'IHM de gestion des normes, c'est l'application qui va créer des règles de duplication en fonction des règles de transformation qu'elle a récupéré précédemment. Une règle de duplication est créée si une règle de l'IHM si une règle de relation a les caractéristiques suivantes :

- elle est de type relation
- il y a un alias suivit d'un point avant le nom de rubrique dans la colonne rubrique nomenclature
- et que les rubriques de la règle sont présentes dans le fichier (si elles ne le sont pas, la règle est inutile car elle concernerait des rubriques inexistantes)

Une règle de duplication va être caractérisée par les deux rubriques qu'elle lie et l'alias que l'on va ajouter lors de la duplication de la table. Par exemple pour la règle

id	type	rubrique	rubrique nomenclature	commentaire
1	relation	V_S21_G00_40_046	siretutil.V_S21_G00_85_001	relation lieu de travail siret utilisateur

	<p>Service Développement Programme SIERA <i>Projet Pyrénées</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 32 sur 44

on va obtenir une règle duplication

id	type	rubrique	rubrique nomenclature	commentaire
1	duplication	V_S21_G00_40_046	V_S21_G00_85_001_siretutil	relation lieu de travail siret utilisateur

Cette règle va avoir pour effet de dupliquer dans la partie de création des tables de la requête de jointure le bloc contenant la rubrique V_S21_G00_85_001. Elle va également rajouter au nom du nouveau bloc, ainsi qu'à toutes ses colonnes, le suffixe « _siretutil ».

Lorsque l'on a parcouru l'ensemble des règles on passe à l'application de ces règles de duplications.

L'application d'une règle de duplication va se faire par l'ajout, dans la partie de la requête de création de table, des tables dupliquées. Pour ce faire on va copier la requête de création de la table qui contient la rubrique contenue dans la colonne rubrique_nomenclature. Rajouter au nom de la table dupliquée, ainsi qu'à toutes ses colonnes, le suffixe défini par la règle. Enfin on va rajouter dans la partie jointure de la requête la table ainsi créée.

Ensuite on applique les règles d'*indépendance*. Comme pour les règles de duplications, ce ne sont pas des règles que l'on peut rentrer depuis l'IHM, elles sont générées par l'application. Les règles d'indépendances ont pour but de limiter le nombre de lignes dans le cas de deux blocs qui n'ont pas de lien métier entre eux.

Exemple


Si on imagine l'arbre suivant

entreprise

->employé

->filiale

avec les informations suivantes

	Service Développement Programme SIERA <i>Projet Pyrénées</i> Application ARC Préparation et Rapport des tests de performance	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 33 sur 44

entreprise

id_ent	nom_ent
1	Boucherie Martin
2	La Boul'ange

employé

id_emp	id_ent	nom_emp
1	1	Alain
2	1	Albane
1	2	Charles
2	2	Pierre
3	2	Celine

filiale

id_fil	id_ent	nom_fil
1	1	Bouchette
1	2	Bouliard
2	2	Pain frais


Sans indépendance on obtiendrait lors de la jointure

id_ent	id_emp	nom_emp	id_fil	nom_fil
1	1	Alain	1	Bouchette
1	2	Albane	1	Bouchette
2	1	Charles	1	Bouliard
2	1	Charles	2	Pain frais
2	2	Pierre	1	Bouliard
2	2	Pierre	2	Pain frais
2	3	Celine	1	Bouliard
2	3	Celine	2	Pain frais

Avec l'indépendance on obtient

id_ent	id_emp	nom_emp	id_fil	nom_fil
1	1	Alain	1	Bouchette
1	2	Albane	1	Bouchette
2	1	Charles	1	Bouliard
2	2	Pierre	2	Pain frais
2	3	Celine	1	Bouliard

Avec l'indépendance on conserve bien toutes les données des blocs employé et filiale mais avec le moins de lignes possible.

	<p>Service Développement Programme SIERA <i>Projet Pyrénées</i></p> <p style="text-align: center;">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 34 sur 44

Pour détecter quels blocs sont indépendants on part du principe que tous les blocs sont indépendants pour commencer. Puis on parcourt les règles de transformation. Si deux blocs sont liés par une règle de relation alors ils ne sont pas indépendants. Mais également, tout leur parent non en commun.

Exemple

A

->A1
 ->A11
 ->A111
 ->A12
 ->A121
 ->A1211

Si A111 et A1211 ne sont pas indépendants, alors A11, A12 et A121 sont également liés. En d'autre terme si deux blocs sont dépendants l'un de l'autre, tous leur père jusqu'au plus petit père commun à ces deux blocs le sont également.

Il est également possible de ne pas lier de bloc par une règle de relation, mais qu'on veuille que ces blocs soit dans une relation cartésienne. Pour ce faire si on spécifie une règle *cartesian* entre ces blocs dans l'IMH on va les exclure des règles d'indépendance. C'est le seul effet des règles cartesian, ne pas rendre indépendant deux blocs.

Pour réaliser cette indépendance, on va dans un premier temps modifier la requête de création des tables liées à ces blocs pour rajouter \$ après le nom de la table et un id technique aux lignes des tables. Cette id va être calculé grâce à la fonction row_number().

Exemple :

Dans l'exemple précédent au lieu d'avoir

```
CREATE TEMPORARY TABLE employe AS (SELECT id_emp, id_ent, nom_emp FROM table_source
WHERE ... ;
```

```
CREATE TEMPORARY TABLE filiale AS (SELECT id_emp, id_fil, nom_fil FROM table_source
WHERE ... ;
```

on a

	<p>Service Développement Programme SIERA <i>Projet Pyrénées</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 35 sur 44

```
CREATE TEMPORARY TABLE employe$ AS (SELECT row_number() over (partition by id_ent) as
r, xx.* from (SELECT id_emp, id_ent, nom_emp FROM table_source WHERE ... )xx);
```

```
CREATE TEMPORARY TABLE filiale$ AS (SELECT row_number() over (partition by id_ent) as
r, xx.* from (SELECT id_ent, id_fil, nom_fil FROM table_source WHERE ... )xx);
```

Cet id technique sert de clef de jointure pour créer une table contenant toutes les informations des tables initiales. Cette table va avoir le nom de la première des deux tables indépendantes qui est apparu dans la partie de création des tables. La table ainsi créée va être utilisée à la place des deux tables dont elle provient dans la partie de la requête de jointure.

Exemple

Si la première table dans la partie création rencontrée est employé, donc on va obtenir


```
CREATE TEMPORARY TABLE employe AS (SELECT id_emp, id_ent, nom_emp, id_fil, nom_fil
FROM employe$, filiale$ WHERE employe$.id_ent=filiale$.id_ent and
employe$.r=filiale$.r) ;
```

ce qui donne bien la table attendue.

Ensuite on applique les règles *d'unicité*. Une règle d'unicité va permettre de ne conserver qu'une ligne, si plusieurs ont le même identifiant. La raison à l'existence de ce type de règle provient du fait que souvent les entreprises vont déclarer plusieurs fois un même lieu de travail. En limitant le nombre de lignes identiques, on limite la taille de la table de jointure en sortie.

Pour implémenter les règles d'unicité, on va modifier la requête sql de création de la table temporaire contenant la rubrique de la règle en y ajoutant une rubrique générée à partir d'une instruction row_number over (partition...). Ensuite via une instruction where on ne va conserver que les lignes avec cette rubrique égale à 1.

Exemple : La requête de création de la table t_S21_G00_85

	<p>Service Développement Programme SIERA <i>Projet Pyrénées</i></p> <p style="text-align: center;">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 36 sur 44

```
CREATE TEMPORARY TABLE t_S21_G00_85 AS (
    SELECT i_S21_G00_85 AS m_S21_G00_85 , i_S20_G00_05 AS i_S20_G00_05 ,
    i_S21_G00_85_001 AS i_S21_G00_85_001 , v_S21_G00_85_001 AS v_S21_G00_85_001 , [...]
    FROM (
        SELECT i_S21_G00_85 ,min( i_S21_G00_85_001 ) as
    i_S21_G00_85_001,min( v_S21_G00_85_001 ) as v_S21_G00_85_001,[...]
        FROM ok_Temp21
        WHERE i_S21_G00_85 IS NOT NULL GROUP BY i_S21_G00_85) a
    , (SELECT DISTINCT i_S21_G00_85 AS pivot, i_S20_G00_05
    FROM ok_Temp21
        WHERE i_S21_G00_85 is not null) b
    WHERE a.i_S21_G00_85 = b.pivot );
```

devient

```
CREATE TEMPORARY TABLE t_s21_g00_85 AS (
    SELECT *
    FROM (
        SELECT CASE
            WHEN v_s21_g00_85_001 IS NULL THEN 1
            ELSE ROW_NUMBER() OVER (PARTITION BY i_s20_g00_05,v_s21_g00_85_001)
            END AS rk_v_s21_g00_85_001
        , *
        FROM (
            SELECT i_s21_g00_85 AS m_s21_g00_85 , i_s20_g00_05 AS
    i_s20_g00_05 , i_s21_g00_85_001 AS i_s21_g00_85_001 , v_s21_g00_85_001 AS
    v_s21_g00_85_001 ,[...]
            FROM (
                SELECT i_s21_g00_85 ,min( i_s21_g00_85_001 ) AS
    i_s21_g00_85_001,min( v_s21_g00_85_001 ) AS v_s21_g00_85_001,[...]
                FROM ok_Temp21
                WHERE i_s21_g00_85 IS NOT NULL GROUP BY i_s21_g00_85) a
            , (SELECT DISTINCT i_s21_g00_85 AS pivot, i_s20_g00_05
            FROM ok_Temp21
                WHERE i_s21_g00_85 IS NOT NULL) b
            WHERE a.i_s21_g00_85 = b.pivot )
            t0_v_s21_g00_85_001 ) t1_v_s21_g00_85_001
        WHERE rk_v_s21_g00_85_001=1);
```

table t_S21_G00_85 sans unicité

M_S21_G00_85_001	I_S21_G00_85_001	V_S21_G00_85_001	...
1	1	boucherie Martin	
2	1	boucherie Martin	
3	1	La Boul'ange	
4	1	La Boul'ange	

	Service Développement Programme SIERA Projet Pyrénées Application ARC Préparation et Rapport des tests de performance	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 37 sur 44

5	1	boucherie Martin	
6	1	La Boul'ange	
7	1	boucherie Martin	
8	1	La Boul'ange	

table t_S21_G00_85 sans unicité

M_S21_G00_85_001	I_S21_G00_85_001	V_S21_G00_85_001	...
1	1	boucherie Martin	
2	1	La Boul'ange	

Enfin on va appliquer les règles de *relation*. Ces règles permettent de lier deux blocs qui n'ont pas de lien hiérarchique entre eux. Par exemple le bloc S21.G00.34 renseigne l'exposition d'un individu à la pénibilité. Il est à mettre en relation avec un contrat (bloc S21.G00.40). Cette mise en relation passe par une association de numéro de contrat (rubrique S21.G00.34.002 et S21.G00.40.009) et va permettre de mettre sur une même ligne la pénibilité associée à un contrat.

Pour ce faire on va modifier la partie jointure de la requête. On va créer un bloc de jointure avec une condition sur l'égalité entre les rubriques de la règle, et un autre bloc de jointure avec une condition sur le fait que la rubrique nomenclature est nulle ou qu'il n'y a pas d'association entre les rubriques de la règle. Et enfin on va joindre ces deux blocs.

Exemple :

Requête initiale :

```
INSERT INTO ok_temp (...)
SELECT (...)
FROM jointure
```

Après ajout de la règle de relation entre V_S21_G00_40_009 et V_S21_G00_34_002

	<p>Service Développement Programme SIERA <i>Projet Pyrénées</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 38 sur 44

```

INSERT INTO ok_temp (...)
SELECT (...)
    FROM jointure
    WHERE v_s21_g00_34_002=v_s21_g00_40_009 } Égalité des rubriques
UNION ALL
SELECT (...)
    FROM jointure
    WHERE (v_s21_g00_34_002 IS NULL OR v_s21_g00_34_002 NOT IN (SELECT
DISTINCT v_s21_g00_40_009 FROM t_s21_g00_40 WHERE v_s21_g00_40_009 IS NOT NULL)) } On ne trouve pas de rubrique liée

```

Ainsi pour n règles de relation on va obtenir 2^n « blocs » à joindre. Cela provient du fait que l'on doit réaliser toutes les combinaisons possibles entre l'acceptation ou le rejet de la règles de relation.


Le transformation se termine par l'exécution de la requête de jointure modifiée et augmentée par les règles de transformation. La table ainsi aplatie est ensuite insérée dans la table TRANSFORMATION_OK et TRANSFORMATION_OK_TODO. Enfin on ajoute une ligne dans la table PILOTAGE_FICHER pour signaler que la transformation est terminé.

5 La phase de contrôle

La phase de contrôle a pour but de filtrer en amont les données qui ne seraient pas exploitables par la suite du processus. Ces contrôles peuvent être sur le format d'une variable (date, longueur, uniquement caractère numérique), sur la cardinalité entre deux blocs ou une condition plus générale sous la forme d'une requête SQL.

Comme les phases précédentes, la phase de contrôle commence pas récupérer les id_source issues de la phase précédente (ici TRANSFORMATION_OK_TODO). Elle va ensuite créer un pool de **connection** et un pool de **threadContrôleService** et contrôler les fichiers en parallèle.

Le contrôle d'un fichier commence par une phase de préparation. Il va mettre à jour une table de pilotage temporaire avec les informations du jeu de règle appliqué à la table. Ces informations sont contenues dans la table JEUDEREGLE. Ensuite il crée la table de contrôle temporaire avec les lignes contenant cette id_source dans la table TRANSFORMATION_OK_TODO, va marquer dans la table de pilotage temporaire ainsi que dans la table de contrôle temporaire si le fichier est associé à un jeu de règle. Enfin il récupère l'intégralité des règles de contrôle rentré par l'utilisateur via l'IHM gestion des normes liées au jeu de règle du fichier. Les règles sont contenues dans la table CONTROLE_REGLE.

	Service Développement Programme SIERA <i>Projet Pyrénées</i> Application ARC Préparation et Rapport des tests de performance	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 39 sur 44

Maintenant que les règles sont récupérées, il va exécuter sur la table de contrôle temporaire les pré-actions des règles. Elles ont pour but de mettre les données au bon format pour le contrôle. Une pré-action se présente sous la forme d'un bout de requête SQL (exemple : lpad({V_S20_G00_05_001},2,'0')). L'application va générer le reste de la requête pour pouvoir mettre à jour la table de contrôle temporaire en utilisant cette requête.

Une fois cela fait on va exécuter les différentes règles. Elles peuvent être de 5 type.

- « Num » : vérifier si la colonne ne contient que des caractères numériques et est à la taille souhaitée

Exemple pour la règle : num sur la rubrique **V_S21_G00_40_012** de longueur min 4 et max 7.


```
WITH ctl AS (
  SELECT id_source, id
  FROM arc_BAS8.CONTROLE_ENCOURS_1$tmp$7332455058$5508_data a
  WHERE (a.v_s21_g00_40_012 ~ '^-\d*(\.\d+)?$') IS FALSE OR
  CASE
    WHEN i_s21_g00_40_012 IS NULL THEN FALSE
    WHEN v_s21_g00_40_012 IS NULL THEN FALSE
    ELSE (char_length(regex_replace(v_s21_g00_40_012, '^-', ''))<4 OR
char_length(regex_replace(v_s21_g00_40_012, '^-', ''))>7)
  END)
  INSERT INTO arc_BAS8.CONTROLE_ENCOURS_1$tmp$7332455058$5508_mark SELECT
id_source, id, '69' FROM ctl;
```

- « Date » : vérifier si la colonne est au bon format de date.

Exemple pour la règle : date sur la rubrique **V_S21_G00_30_006** avec pour format de date yyyy-mm-dd

```
WITH ctl AS (
  SELECT id_source,id
  FROM arc_BAS8.CONTROLE_ENCOURS_1$tmp$7332455058$5508_data
  WHERE CASE
    WHEN arc.isdate(V_S21_G00_30_006,'yyyy-mm-dd') THEN FALSE
    WHEN V_S21_G00_30_006 IS NULL THEN FALSE
    ELSE TRUE
  END)
INSERT INTO arc_BAS8.CONTROLE_ENCOURS_1$tmp$7332455058$5508_mark SELECT id_source, id,
'51' FROM ctl;
```

- « Alphanum » : vérifier le nombre de caractère des éléments de la colonne



	<p>Service Développement Programme SIERA <i>Projet Pyrénées</i></p> <p>Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 40 sur 44

Exemple pour la règle : alphanum sur la rubrique `v_s21_g00_40_004` avec pour longueur min 4 et max 4.

```
WITH ctl AS (
    SELECT id_source, id
    FROM arc_BAS8.CONTROLE_ENCOURS_1$tmp$7332455058$5508_data
    WHERE 1=2 OR CASE
        WHEN i_s21_g00_40_004 IS NULL THEN FALSE
        WHEN v_s21_g00_40_004 IS NULL THEN FALSE
        ELSE (char_length(regexp_replace(v_s21_g00_40_004, '^-', ''))<4 OR
char_length(regexp_replace(v_s21_g00_40_004, '^-', ''))>4)
    END)
INSERT INTO arc_BAS8.CONTROLE_ENCOURS_1$tmp$7332455058$5508_mark SELECT id_source, id,
'52' FROM ctl;
```

- « Cardinalité » : ...
- « Condition » : ici c'est une requête SQL qui va complètement définir le contrôle. Le normage s'occupe juste de l'exécuter.

Exemple pour la règle : condition {V_S21_G00_51_001} <= ({V_S21_G00_51_002}) OR {V_S21_G00_51_001} IS NULL OR {V_S21_G00_51_002} IS NULL

	<p>Service Développement Programme SIERA <i>Projet Pyrénées</i></p> <p style="text-align: center;">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 41 sur 44

```



WITH ctl AS (
SELECT id_source, id
FROM (
    SELECT id_source, id, CASE
        WHEN (CASE
            WHEN i_s21_g00_51_001 IS NULL THEN TRUE
            WHEN v_s21_g00_51_001 IS NULL THEN TRUE
            ELSE arc.isdate(v_s21_g00_51_001, 'yyyy-mm-dd')
        END) AND (CASE
            WHEN i_s21_g00_51_002 IS NULL THEN TRUE
            WHEN v_s21_g00_51_002 IS NULL THEN TRUE
            ELSE arc.isdate(v_s21_g00_51_002, 'yyyy-mm-dd')
        END) THEN CASE
            WHEN to_date(v_s21_g00_51_001, 'yyyy-mm-dd') <=
(to_date(v_s21_g00_51_002, 'yyyy-mm-dd')) OR to_date(v_s21_g00_51_001, 'yyyy-mm-dd') IS
NULL OR to_date(v_s21_g00_51_002, 'yyyy-mm-dd') IS NULL THEN FALSE
            ELSE TRUE
        END ELSE FALSE
    END AS condition_a_tester
    FROM arc_BAS8.CONTROLE_ENCOURS_1$tmp$7332455058$5508_data
) ww WHERE condition_a_tester )
INSERT INTO arc_BAS8.CONTROLE_ENCOURS_1$tmp$7332455058$5508_mark SELECT id_source, id,
'60' FROM ctl;

```

Les règles sont stockées sous une forme réduite. Pour une règle de contrôle num on ne va avoir que la colonne à tester ainsi que l'intervalle de taille valide. L'application va ensuite générer la requête SQL qui rend le contrôle possible.

Lorsqu'un enregistrement ne passe un contrôle il va mettre à jour une table temporaire pour garder en mémoire l'id_source et l'id de la ligne incriminée ainsi que l'id de la règle de contrôle qui n'a pas été validé.

Lorsque tous les contrôles ont été exécutés, le contrôle calcule le taux d'erreur du fichier ($tx_erreur = \frac{nb_ligne_erreur}{nb_ligne_total}$). Si ce taux dépasse un certain seuil (ce seuil se trouve dans la table SEUIL) le fichier est mis en KO dans la table de pilotage temporaire et enregistré dans la table CONTROLE_KO. Si le fichier présente des lignes en anomalie, mais avec un taux d'erreur inférieur au seuil, alors le fichier est mis en OK_KO, signe que certains contrôles n'ont pas pu être vérifiés. Les lignes incriminées sont insérées dans la table CONTROLE_KO, et le reste du fichier est inséré dans la table CONTROLE_OK. Enfin, si le taux d'erreur est nul, le fichier est mis en OK il est alors enregistré dans la table CONTROLE_OK et CONTROLE_OK_TODO en utilisant un type composite.

	<p>Service Développement Programme SIERA <i>Projet Pyrénées</i></p> <p align="center">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 42 sur 44

	État_traitement	CONTROLE_OK	CONTROLE_KO
$tx_erreur \geq seuil$	KO	aucune ligne	tous le fichier
$0 > tx_erreur > seuil$	OK_KO	ligne sans erreur	ligne en anomalie
$tx_erreur = 0$	OK	tous le fichier	aucune ligne

6 La phase de filtrage



À la différence du contrôle qui accepte ou rejette un fichier dans son ensemble, le filtrage accepte ou rejette que les enregistrements ayant un intérêt métier. Ainsi un même fichier va pouvoir avoir des lignes rejetées et acceptées lors du filtrage.

Avant le filtrage l'application récupère les id_source ayant passé le contrôle (table CONTROLE_OK_TODO) et génère un pool de **connection** ainsi que **threadFiltrageService** pour filtrer les fichiers en parallèle.

Ensuite, vient une phase de préparation du filtrage. Chaque thread va créer une table temporaire contenant les données d'un id_source, une table temporaire de filtrage ok et ko. La préparation du filtrage va également aller chercher en base le seuil d'acceptation des fichiers dans la table SEUIL. Si on refuse trop de ligne d'un fichier, on le considère alors ko.

Enfin on réalise le filtrage. Pour ce faire on va chercher les règles à appliquer dans la table REGLE_FILTRAGE. Ces règles on était au préalable créer par l'utilisateur via l'IHM gestion des normes. Une règle de filtrage se présente sous la forme d'une condition avec la colonne à tester entre crochets (exemple : {V_S10_G00_00_005}='01'). Le filtrage va transformer cette condition en requête SQL et l'exécuter sur la table temporaire. On ne va garder que les lignes qui ne valident pas l'expression. Ainsi une condition de la forme 1=2 va accepter toutes les lignes. Les lignes ayant passé le filtrage sont copiées dans la table de filtrage temporaire ok et les autre dans la table de filtrage temporaire ko.

Exemple de requête sql générée avec le filtre {V_S10_G00_00_005}='01'

	<p>Service Développement Programme SIERA <i>Projet Pyrénées</i></p> <p style="text-align: center;">Application ARC Préparation et Rapport des tests de performance</p>	
Rédacteur(s) : R.PEPIN	Documentation technique	Page 43 sur 44



```

CREATE TEMPORARY TABLE temp_controle_ok_COUNT WITH (autovacuum_enabled = false,
toast.autovacuum_enabled = false) AS
    SELECT id_source, id, (CASE
        WHEN FALSE THEN 2
        WHEN EXISTS(
            SELECT 1 FROM
arc_BAS8.FILTRAGE_PILOTAGE_FICHER_0$tmp$6333509158$9833 pil
            WHERE pil.id_norme='PHASE2v1_v2'
            AND pil.periodicite='M'
            AND pil.validite_inf='2015-01-01'
            AND pil.validite_sup='2017-12-31'
            AND pil.id_source = ctrl.id_source)
        THEN CASE
            WHEN v_s10_g00_00_005='01' THEN 1
            ELSE 0
            END
        ELSE 2
        END)::bigint AS check_against_rule
FROM temp_controle_ok ctrl;

INSERT INTO temp_filtrage_ko (liste_colonne)
SELECT liste_colonne
FROM temp_controle_ok a
WHERE EXISTS (
    SELECT 1
    FROM arc_BAS8.FILTRAGE_PILOTAGE_FICHER_0$tmp$6333509158$9833 b
    WHERE etat_traitement='{KO}' AND a.id_source=b.id_source)
OR (
    SELECT check_against_rule
    FROM temp_controle_ok_COUNT b
    WHERE a.id_source=b.id_source AND a.id=b.id)=1;

INSERT INTO temp_filtrage_ok (liste_colonne)
SELECT liste_colonne
FROM temp_controle_ok a
WHERE (
    SELECT check_against_rule
    FROM temp_controle_ok_COUNT b
    WHERE a.id_source=b.id_source AND a.id=b.id)=0
AND exists (
    SELECT 1
    FROM arc_BAS8.FILTRAGE_PILOTAGE_FICHER_0$tmp$6333509158$9833 b
    WHERE etat_traitement IN ('{OK}', '{OK,KO}') AND a.id_source=b.id_source);

```

	<p>Service Développement Programme SIERA <i>Projet Pirénés</i></p> <p style="text-align: center;">Application ARC Préparation et Rapport des tests de performance</p>		
Rédacteur(s) : R.PEPIN	Documentation technique	Page 44 sur 44	

Finalement on va mettre à jour la table de pilotage et associer les modalités suivantes au déroulement du filtrage : OK si aucune ligne de fichier n'est refusée, OK KO si le taux de refus est inférieur au seuil d'acceptation et KO si le taux d'erreur est supérieur au seuil ou qu'aucune règle n'est trouvé pour ce fichier, ou que ce fichier ne possède aucune ligne. On transfère ensuite les données des tables temporaires vers les tables définitives associées (FILTRAGE_OK, FILTRAGE_OK_TODO, FILTRAGE_KO, PILOTAGE_FICHER)

7 La phase de mapping

Là où on va passer d'un schéma fluctuant à un schéma fixe avec des concepts métiers.