

Aide mé moire à la transition de SQL à XPath 1.0.

Cette documentation s'adresse à ceux qui connaissent le langage SQL mais qui débutent avec XSLT/XPath. Si vous savez comment, via une requête SQL, récupérer les informations de votre base de données mais que vous êtes face à une base de données XML, cette doc devrait vous aider.

Note: Ce travail est en cours de rédaction. Si certaines parties manquent de clarté votre contribution est bienvenue. Malgré tous mes efforts des erreurs peuvent apparaître. Veuillez me les communiquer le cas échéant. N'hésitez pas non plus à m'envoyer vos commentaires.

Auteur: Frédéric Desmoulins <xpath At fragbase.com>

Réf. Web: <http://www.fragbase.com/sql2xpath.php>

Dernière modification: 21/03/2004

Licence: GNU FDL

AVG

XPath 1.0 ne dispose pas d'une fonction permettant de calculer une moyenne. Il faudra effectuer la somme (cf. [sum](#)) puis la diviser par le nombre d'occurrence (cf. [count](#)).

PERSONNE

id (unsigned int)
nom (varchar 256)
prenom (varchar 256)
poids (unsigned float)

```
<personnes>
  <personne id="1">
    <nom>Dupont</nom>
    <prenom>Michel</prenom>
    <poids>77</poids>
  </personne>
  <personne id="2">
    <nom>Durand</nom>
    <prenom>Paul</prenom>
    <poids>68</poids>
  </personne>
  <personne id="3">
    <nom>Clavier</nom>
    <prenom>George</prenom>
    <poids>92</poids>
  </personne>
</personnes>
```

Exemple : Calculer la moyenne des poids.

```
SQL: SELECT AVG(poids) FROM PERSONNE
```

```
XPath 1.0:
sum(/personnes/personne/poids)
div
count(/personnes/personne/poids)
```

COUNT

On utilise la fonction `count(nodes)` pour compter le nombre de noeud localisé par l'expression XPath `nodes`. On peut également obtenir le nombre d'élément contextuel via la fonction `last()`.

PERSONNE

id (unsigned int)
nom (varchar 256)
prenom (varchar 256)

```
<personnes>
  <personne id="1">
    <nom>Dupont</nom>
    <prenom>Michel</prenom>
  </personne>
  <personne id="2">
    <nom>Durand</nom>
    <prenom>Paul</prenom>
  </personne>
  <personne id="3">
    <nom>Clavier</nom>
    <prenom>George</prenom>
  </personne>
</personnes>
```

Exemple : Compter le nombre de personne.

```
SQL: SELECT COUNT(*) FROM PERSONNE
```

```
XPath: count(/personnes/personne)
```

DISTINCT

Il n'y a pas fonction équivalente au DISTINCT de SQL dans les spécifications de XPath 1.0. Cependant il est possible d'y remédier grâce au sélecteur d'axe. Sélectionner les noeuds distincts revient à sélectionner les noeuds qui n'ont pas de précédent - ou de suivant - 'semblable'.

PERSONNE
<code>id (unsigned int)</code>
<code>nom (varchar 256)</code>
<code>prenom (varchar 256)</code>
<code>taille (unsigned float)</code>

```
<personnes>
  <personne id="1">
    <nom>Dupont</nom>
    <prenom>Michel</prenom>
    <taille>1.70</taille>
  </personne>
  <personne id="2">
    <nom>Durand</nom>
    <prenom>Paul</prenom>
    <taille>1.7</taille>
  </personne>
  <personne id="3">
    <nom>Dupont</nom>
    <prenom>George</prenom>
    <taille>1.80</taille>
  </personne>
</personnes>
```

Exemple 1 : Sélectionner les noms de personnes différents.

```
SQL: SELECT DISTINCT nom FROM PERSONNE
```

```
XPath 1.0:
/personnes/personne[
  not(nom = preceding-sibling::personne/nom)
]/nom
ou
/personnes/personne[
  not(nom = following-sibling::personne/nom)
]/nom
```

Exemple 2 : Sélectionner les tailles de personnes différents.

```
SQL: SELECT DISTINCT taille FROM PERSONNE
```

```
XPath 1.0:
/personnes/personne[
  not(number(taille) = preceding-sibling::personne/taille)
]/taille
ou
/personnes/personne[
  not(number(taille) = following-sibling::personne/taille)
]/taille
```

Important: N'oubliez pas d'utiliser `number(node)` lorsque vous effectuez des comparaisons entre des nombres.

Note: `sibling` exige que les éléments précédent - ou suivant - aient le même élément parent que l'élément contextuel (l'axe parent). Dans notre exemple qui ne contient que des personnes on aurait pu tout aussi bien utiliser `preceding::nom` ou `following::nom`. Ce qui revient à rechercher les éléments 'nom' précédent ou suivant l'élément contextuel (Attention: XPath fait alors la sélection à l'intérieur comme à l'extérieur de l'axe parent!).

FROM

Par défaut XPath parcourt les éléments appartenant au document XML courant (celui qui a été chargé par votre processeur XSLT ou bien celui qui est référencé dans votre document comme une feuille de style XSL). On peut néanmoins sélectionner des éléments appartenant à un autre document XML en le référençant - via une URI, un chemin absolu ou relatif de votre système de fichier - grâce à l'instruction `document('URI')`.

Il est également possible d'effectuer une sélection multiple en combinant des noeuds grâce à l'opérateur 'ou' représenté par: `|`.

PERSONNE
<code>id (unsigned int)</code>
<code>nom (varchar 256)</code>
<code>prenom (varchar 256)</code>

```
<personnes>
  <personne id="1">
    <nom>Dupont</nom>
    <prenom>Michel</prenom>
  </personne>
  <personne id="2">
    <nom>Durand</nom>
    <prenom>Paul</prenom>
  </personne>
  <personne id="3">
    <nom>Clavier</nom>
    <prenom>George</prenom>
  </personne>
</personnes>
```

Exemple 1 : Sélectionner tous les éléments de type 'personne' dans le document situé à l'URL `http://example.com/personnes.xml`.

```
XPath:
document('http://example.com/personnes.xml')//personne
```

Exemple 2 : Sélectionner tous les éléments de type 'personne' dans le document situé à l'URL `http://example.com/personnes.xml` avec ceux appartenant au document courant.

```
XPath:
document('http://example.com/personnes.xml')//personne |
//personne
```

GROUP BY & HAVING

Le regroupement ne peut pas s'effectuer via XPath. Il faudra le faire via le langage manipulant vos expressions XPath (XSLT par exemple).

PERSONNE
<i>id</i> (unsigned int) <i>nom</i> (varchar 256) <i>prenom</i> (varchar 256) <i>poids</i> (unsigned float) <i>taille</i> (unsigned float)

```
<personnes>
  <personne id="1">
    <nom>Dupont</nom>
    <prenom>Michel</prenom>
    <poids>77</poids>
    <taille>1.70</taille>
  </personne>
  <personne id="2">
    <nom>Durand</nom>
    <prenom>Paul</prenom>
    <poids>68</poids>
    <taille>1.70</taille>
  </personne>
  <personne id="3">
    <nom>Clavier</nom>
    <prenom>George</prenom>
    <poids>92</poids>
    <taille>1.80</taille>
  </personne>
</personnes>
```

Exemple : Afficher le poids moyen en fonction des différentes tailles.

Méthode XPath:

a) Parcourir l'ensemble des tailles distinctes (cf. distinct)

b) A chaque taille nommée 't(i)' provenant de (a), calculer la moyenne (cf. avg) des poids du groupe défini par la règle 'taille=t(i)'.

Sortie attendue:

```
<moyennePoids taille="1.7">72.5</moyennePoids>
<moyennePoids taille="1.8">92</moyennePoids>
```

SQL: SELECT taille, AVG(poids) FROM PERSONNE GROUP BY taille;

XSLT:

```
<xsl:for-each select="
  /personnes/personne[
    not(number(taille)=preceding-sibling::personne/taille)
  ]/taille">
  <moyennePoids taille="number(.)">
    <xsl:value-of select="
      sum(/personnes/personne[number(taille)=current()]/poids)
      div
      count(/personnes/personne[number(taille)=current()])
    " />
  </moyennePoids>
</xsl:for-each>
```

MIN/MAX

XPath 1.0 ne propose pas de fonctions équivalentes aux MIN et MAX de SQL. Cependant il est possible d'arriver au même résultat grâce au sélecteur d'axe. Sélectionner le noeud le plus 'petit'/'grand' revient à sélectionner le noeud qui n'a pas de précédent ni de suivant plus 'petit'/'grand'.

PERSONNE
<i>id</i> (unsigned int) <i>nom</i> (varchar 256) <i>prenom</i> (varchar 256) <i>poids</i> (unsigned float)

```
<personnes>
  <personne id="1">
    <nom>Dupont</nom>
    <prenom>Michel</prenom>
    <poids>77</poids>
  </personne>
  <personne id="2">
    <nom>Durand</nom>
    <prenom>Paul</prenom>
    <poids>68</poids>
  </personne>
  <personne id="3">
    <nom>Clavier</nom>
    <prenom>George</prenom>
    <poids>92</poids>
  </personne>
</personnes>
```

Exemple 1 : Obtenir la ou les personnes ayant le poids le plus léger.

SQL:

```
SELECT * FROM PERSONNE WHERE poids IN (
  SELECT MIN(poids) FROM PERSONNE
)
```

XPath 1.0:

```
/personnes/personne[
  not(number(poids) > (
    preceding-sibling::personne/poids |
    following-sibling::personne/poids
  ))
]
```

Exemple 2 : Obtenir la ou les personnes ayant le poids le plus lourd.

SQL:

```
SELECT * FROM PERSONNE WHERE poids IN (
  SELECT MAX(poids) FROM PERSONNE
)
```

XPath 1.0:

```
/personnes/personne[
  not(number(poids) < (
    preceding-sibling::personne/poids |
    following-sibling::personne/poids
  ))
]
```

Note: sibling exige que les éléments précédent - ou suivant - aient le même élément parent que l'élément contextuel. Dans notre exemple qui ne contient que des personnes on aurait pu tout aussi bien utiliser preceding::poids et following::poids: Ce qui revient à rechercher les éléments 'poids' précédent ou suivant l'élément contextuel (Attention: XPath fait alors la sélection à l'intérieur comme à l'extérieur de l'axe parent !).

ORDER BY

Le tri d'une sélection ne se fait pas via une expression XPath mais généralement grâce au langage manipulant XPath. Si l'on utilise XPath au travers d'XSLT on peut alors parcourir une sélection dans un ordre prédéfini.

PERSONNE
<code>id (unsigned int)</code>
<code>nom (varchar 256)</code>
<code>prenom (varchar 256)</code>
<code>poids (unsigned float)</code>

```
<personnes>
  <personne id="1">
    <nom>Dupont</nom>
    <prenom>Michel</prenom>
    <poids>77</poids>
  </personne>
  <personne id="2">
    <nom>Durand</nom>
    <prenom>Paul</prenom>
    <poids>68</poids>
  </personne>
  <personne id="3">
    <nom>Clavier</nom>
    <prenom>George</prenom>
    <poids>92</poids>
  </personne>
</personnes>
```

Exemple 1 : Parcourir les personnes dans l'ordre alphabétique de leur nom.

```
SQL: SELECT * FROM PERSONNE ORDER BY nom
```

```
XSLT:
<xsl:for-each select="/personnes/personne">
  <xsl:sort select="nom" />
  <!-- Implémentation -->
</xsl:for-each>
```

Exemple 2 : Parcourir les personnes du plus lourd au plus léger.

```
SQL: SELECT * FROM PERSONNE ORDER BY poids DESC
```

```
XSLT:
<xsl:for-each select="/personnes/personne">
  <xsl:sort select="poids"
    data-type="number"
    order="descending" />
  <!-- Implémentation -->
</xsl:for-each>
```

Important: N'oubliez pas d'utiliser `data-type="number"` lorsque vous classez des nombres.

SELECT

Pour sélectionner des noeuds dans l'arbre XML on définit le chemin (absolu ou relatif) de localisation. La localisation s'effectue comme celle que l'on utilise avec les shell UNIX: avec le caractère `/`.

Pour parcourir l'arborescence à partir de la racine (chemin absolu) on commence l'expression XPath avec `/`, puis on descend dans l'arbre. On peut également utiliser une autre technique qui consiste à indiquer l'élément recherché précédé de `//`. Attention ce dernier moyen oblige votre processeur XPath à parcourir tout l'arbre sans considérer aucun axe (à éviter lorsque ce n'est pas nécessaire) !

PERSONNE
<code>id (unsigned int)</code>
<code>nom (varchar 256)</code>
<code>prenom (varchar 256)</code>

```
<personnes>
  <personne id="1">
    <nom>Dupont</nom>
    <prenom>Michel</prenom>
  </personne>
  <personne id="2">
    <nom>Durand</nom>
    <prenom>Paul</prenom>
  </personne>
  <personne id="3">
    <nom>Clavier</nom>
    <prenom>George</prenom>
  </personne>
</personnes>
```

Exemple : Sélectionner tous les éléments de type personne (donc tous les sous éléments et attributs de 'personne': nom, prénom, id).

```
SQL: SELECT * FROM PERSONNE
```

```
XPath:
/personnes/personne
ou
//personne
```

SUM

On utilise la fonction `sum(numberNodes)` pour effectuer la somme des noeuds localisés par l'expression XPath `numberNodes`. Si tous les noeuds ne sont pas des nombres, `NaN` (Not A Number) est retourné.

PERSONNE

Exemple : Compter la somme du poids de chaque personne.

```
id (unsigned int)
nom (varchar 256)
prenom (varchar 256)
poids (unsigned float)
```

```
SQL: SELECT SUM(poids) FROM PERSONNE
```

```
XPath: sum(/personnes/personne/poids)
```

```
<personnes>
  <personne id="1">
    <nom>Dupont</nom>
    <prenom>Michel</prenom>
    <poids>77</poids>
  </personne>
  <personne id="2">
    <nom>Durand</nom>
    <prenom>Paul</prenom>
    <poids>68</poids>
  </personne>
  <personne id="3">
    <nom>Clavier</nom>
    <prenom>George</prenom>
    <poids>92</poids>
  </personne>
</personnes>
```

WHERE

Pour sélectionner des éléments en appliquant une ou plusieurs conditions on utilise des crochets contenant le ou les tests ['condition'].

Les prédicats s'appliquent sur les noeuds définis par le chemin précédent les crochets ou, par défaut, sur les noeuds contextuels. On peut combiner les tests en utilisant and et/ou or dans la même paire de crochet.

PERSONNE

```
id (unsigned int)
nom (varchar 256)
prenom (varchar 256)
poids (unsigned float)
```

Exemple 1 : Sélectionner tous les éléments de type personne qui ont un nom commençant par un 'D'.

```
SQL: SELECT * FROM PERSONNE WHERE nom='D%'
```

```
XPath: /personnes/personne[starts-with(nom, 'D')]
```

```
<personnes>
  <personne id="1">
    <nom>Dupont</nom>
    <prenom>Michel</prenom>
    <poids>77</poids>
  </personne>
  <personne id="2">
    <nom>Durand</nom>
    <prenom>Paul</prenom>
    <poids>68</poids>
  </personne>
  <personne id="3">
    <nom>Clavier</nom>
    <prenom>George</prenom>
    <poids>92</poids>
  </personne>
</personnes>
```

Exemple 2 : Sélectionner les poids des personnes qui sont supérieurs à 70 et qui ont un nom commençant par un 'D'.

```
SQL: SELECT poids FROM PERSONNE WHERE poids>70 AND nom='D%'
```

```
XPath:
/personnes/personne[
  number(poids) > 70 and starts-with(nom, 'D')
]/poids

ou

/personnes/personne[
  starts-with(nom, 'D')
]/poids[number(.) > 70]
```