IBM DB2 Universal Database

Extension XML
Administration et programmation

Version 7

IBM DB2 Universal Database

Extension XML
Administration et programmation

Version 7

Important

Avant d'utiliser ce manuel et le produit associé, prenez connaissance des informations générales figurant à la section «Remarques» à la page 299.

LE PRESENT DOCUMENT EST LIVRE ″EN L'ETAT″. IBM DECLINE TOUTE RESPONSABILITE, EXPRESSE OU IMPLICITE, RELATIVE AUX INFORMATIONS QUI Y SONT CONTENUES, Y COMPRIS EN CE QUI CONCERNE LES GARANTIES DE QUALITE MARCHANDE OU D'ADAPTATION A VOS BESOINS. Certaines juridictions n'autorisent pas l'exclusion des garanties implicites, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. Les informations qui y sont fournies sont susceptibles d'être modifiées avant que les produits décrits ne deviennent eux-mêmes disponibles. En outre, il peut contenir des informations ou des références concernant certains produits, logiciels ou services non annoncés dans ce pays. Cela ne signifie cependant pas qu'ils y seront annoncés.

Pour plus de détails, pour toute demande d'ordre technique, ou pour obtenir des exemplaires de documents IBM, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial.

Vous pouvez également consulter les serveurs Internet suivants :

- http://www.fr.ibm.com (serveur IBM en France)
- http://www.can.ibm.com (serveur IBM au Canada)
- http://www.ibm.com (serveur IBM aux Etats-Unis)

Compagnie IBM France
Direction Qualité
Tour Descartes
92066 Paris-La Défense Cedex 50

Table des matières

Tableaux

Avant-propos

Cette section contient les informations suivantes :

Lecteurs concernés

Le présent manuel s'adresse aux lecteurs suivants :

- Utilisateurs manipulant des données XML dans des applications DB2 et familiarisés avec les concepts XML. Les lecteurs de ce manuel doivent avoir des notions d'XML et de DB2. Pour plus d'informations sur XML et les rubriques connexes, reportez-vous au site Web suivant :

  http://www.w3c.org/XML

  Pour plus d'informations sur DB2, reportez-vous au site Web suivant :

  http://www.ibm.com/software/data/db2/library
- Administrateurs de bases de données DB2 familiarisés avec les concepts, les outils et les techniques d'administration de DB2.
- Programmeurs d'applications DB2 familiarisés avec le langage SQL et un ou plusieurs langages de programmation pouvant être utilisés pour les programmes d'application DB2.

Dernière version du manuel

Vous pouvez télécharger la dernière version du présent manuel à partir du site Web de l'Extension XML :

http://www.ibm.com/software/data/db2/extenders/xmlext/library.html

Structure du manuel

Le présent manuel contient les parties suivantes :

Partie 1. Introduction

Cette partie présente l'Extension XML et vous indique comment l'utiliser dans vos applications de gestion. Elle contient un scénario d'initiation qui facilite la mise en route.

Partie 2. Administration

Cette partie indique comment préparer une base de données DB2 pour des données XML et assurer sa maintenance. Consultez-la si vous devez administrer une base de données DB2 qui contient des données XML.

Partie 3. Programmation

Cette partie indique comment gérer les données XML. Consultez-la si vous devez accéder à des données XML et les manipuler dans un programme d'application DB2.

Partie 4. Guide de référence

Cette partie indique comment utiliser les commandes d'administration, les types UDT, les fonctions UDF et les procédures mémorisées de l'Extension XML. Elle répertorie également les messages et les codes émis par l'Extension XML. Consultez-la si vous êtes familiarisé avec les concepts et les tâches de l'Extension XML et que vous avez besoin d'informations sur un type défini par l'utilisateur (UDT), une fonction définie par l'utilisateur (UDF), une commande, un message, des tables de métadonnées, des tables de contrôle ou un code.

Partie 5. Annexes

Les annexes décrivent la définition ou déclaration de type de document (DTD) applicable à la définition d'accès au document (DAD). Elles contiennent des échantillons de programmes exemples, un scénario d'initiation et d'autres produits XML IBM.

Nouveautés de ce manuel concernant DB2 UDB version Fixpak 2

Le présent manuel contient des informations nouvelles ou mises à jour concernant :

- La configuration et le démarrage de l'assistant d'administration
- Les modifications apportées par la fonction UDF Update aux documents XML
- Le mode de renvoi des informations utilisé par la fonction UDF XMLClob
- La façon dont vous pouvez augmenter les limites CLOB des procédures mémorisées
- Les modifications des conditions requises pour les fichiers DAD :
  - Inclusion d'instructions de traitement des feuilles de style lors de la composition de nouveaux documents XML
  - Utilisation d'une condition ″manquante″ ou ″vide″ pour le premier noeud RDB lorsqu'une seule table est spécifiée.
- La méthode de transtypage de marqueurs de paramètre avec JDBC

- L'utilisation des pages de code :
  - Déclarations ENCODING admises
  - Hypothèses de conversion
  - Scénarios de conversion
- L'annexe sur les limites des paramètres de l'Extension XML

Les modifications apportées au présent manuel sont signalées par une barre verticale, «|», en marge gauche.

Pour plus d'informations sur les défauts corrigés dans le présent fixpak, reportez-vous au fichier Readme.

Reportez-vous à la page d'aide du site Web associé à l'Extension XML pour connaître les questions les plus couramment posées et les problèmes connus, ainsi que pour obtenir d'autres informations sur les correctifs et solutions correspondants :
http://www.ibm.com/software/data/db2/extenders/xmlext/support.html

Insertion de ce manuel dans le centre d'aide et d'information de DB2 UDB version 7

Pour inclure la documentation de l'Extension XML dans le centre d'aide et d'information DB2, procédez comme suit :

- Copiez les fichiers HTML de ce manuel, du sous-répertoire `DOC` correspondant à votre langue, vers le sous-répertoire où se trouve la documentation DB2 UDB dans votre langue :

  Sous Windows, le répertoire du centre d'aide et d'information est `sqllib\doc\html\db2sx`

  Sous UNIX, le répertoire du centre d'aide et d'information est *`rép_install`*`/doc/html/db2sx`
- Redémarrez le centre d'aide et d'information. Le manuel est ajouté sur la page de l'onglet **Documentation**.

Conventions de mise en évidence

Le présent manuel utilise les conventions suivantes :

Gras — Le texte en gras indique les éléments suivants :
- Commandes
- Noms de zones
- Noms de menus
- Boutons de commande

Italique	Le texte en italique indique les éléments suivants : • Paramètres de variables à remplacer par une valeur • Mots mis en évidence • Première occurrence d'un terme du glossaire
MAJUSCULES	Les majuscules indiquent les éléments suivants : • Types de données • Noms de colonnes • Noms de tables
`Exemple`	Le texte Exemple indique les éléments suivants : • Messages système • Valeurs entrées • Exemples de programmation • Noms de répertoires • Noms de fichiers • Noms de chemins

Lecture des diagrammes de syntaxe

Dans le présent manuel, des diagrammes de syntaxe décrivent la syntaxe des commandes et des instructions SQL.

Ils se lisent de la manière suivante :

- Les diagrammes de syntaxe se lisent de la gauche vers la droite et du haut vers le bas. Ils figurent après le chemin d'accès.

  Le symbole ▶▶— indique le début d'une instruction.

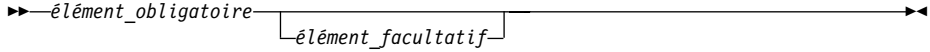
  Le symbole —▶ indique que la fin de la syntaxe figure sur la ligne suivante.

  Le symbole ▶— indique que le début d'une instruction figure sur la ligne précédente.

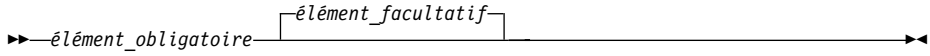
  Le symbole —▶◀ indique la fin d'une instruction.

  Les diagrammes représentant des unités syntaxiques autres que des instructions complètes commencent par le symbole ▶— et se terminent par le symbole —▶.
- Les éléments obligatoires apparaissent sur la ligne horizontale (chemin principal).

- Les éléments facultatifs apparaissent au-dessous du chemin principal.

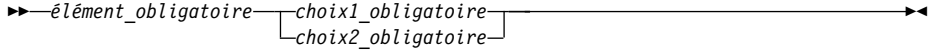


Si un élément facultatif apparaît au-dessus du chemin principal, il n'a aucun effet sur l'exécution de l'instruction et est uniquement utilisé pour une meilleure lisibilité.

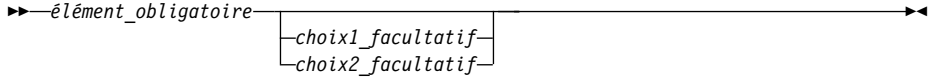


- Si vous pouvez choisir entre plusieurs éléments, ceux-ci apparaissent verticalement, les uns au-dessus des autres.

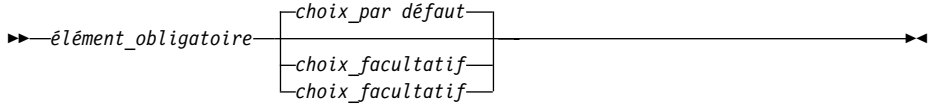
  Si vous *devez* choisir l'un de ces éléments, un élément de la liste apparaît dans le chemin principal.



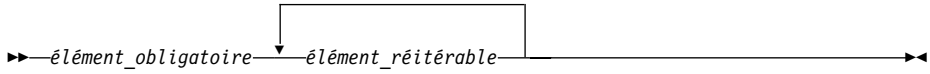
Si le choix d'un élément est facultatif, l'ensemble de la liste apparaît en-dessous du chemin principal.



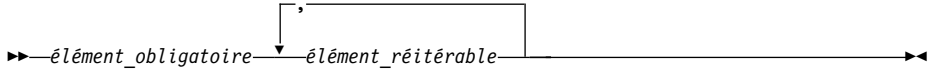
Si l'un des éléments correspond à la valeur par défaut, il apparaît au-dessus du chemin principal et les autres options possibles sont indiquées au-dessous.



- Une flèche revenant vers la gauche, au-dessus du chemin principal, indique que l'élément peut être répété.



  Si la flèche de répétition contient des signes de ponctuation, vous devez séparer les éléments réitérables avec ces derniers.



  Une flèche de répétition au-dessus d'une liste indique que vous pouvez répéter plusieurs éléments de cette dernière.
- Les mots clés apparaissent en majuscules, (par exemple, FROM). L'Extension XML admet toutes les majuscules pour les mots clés. Les autres termes apparaissent en minuscules (par exemple, *nom-colonne*). Ils représentent les noms ou les valeurs fournis par l'utilisateur.
- Les signes de ponctuation, les parenthèses, les opérateurs arithmétiques et tout autre symbole présentés font également partie de la syntaxe.

Informations connexes

Les documents suivants peuvent être utiles lors de l'utilisation de l'Extension XML et des produits associés :

Document	Référence de la commande	Description
Call Level Interface Guide and Reference	SC09–2950	Ce manuel explique comment écrire des applications à l'aide de CLI pour accéder à des serveurs DB2.
DB2 Application Development Guide	SC09–2949	Ce manuel décrit la procédure de développement d'applications. Il indique comment coder, compiler et exécuter des programmes d'application qui accèdent à la base de données par des instructions SQL imbriquées et des API.

Document	Référence de la commande	Description
Page DB2 Extensions	Non disponible	Cette page contient des informations sur les extensions DB2 ainsi que sur les technologies associées. Son adresse Web est la suivante : http://www.software.ibm.com/data/db2/extenders
DB2 SQL Reference for Universal Database Parts 1 and 2	• Part 1 : SC09–2974 • Part 2 : SC09–2975	Ces manuels décrivent la syntaxe, la sémantique et les règles du langage SQL. Ils contiennent également des informations sur les incompatibilités entre les versions, sur les limites liées aux produits et des vues du catalogue.
• *DB2 Universal Database Administration Guide: Implementation* • *DB2 Universal Database Administration Guide: Performance* • *DB2 Universal Database Administration Guide: Planning*	• Implementation : SC09–2944 • Performance : SC09–2945 • Planning : SC09–2946	Ces manuels décrivent comment concevoir, mettre en oeuvre une base de données DB2 et assurer sa maintenance.
DB2 UDB Extensions Image, Audio et Video - Administration et programmation	SC11–1682	Ce manuel indique comment administrer une base de données DB2 UDB pour des données image, audio et vidéo. Il explique également comment utiliser les API (interfaces de programme d'application) fournies pour accéder aux données de ce type et les manipuler.

Document	Référence de la commande	Description
DB2 UDB Extension Texte - Administration et programmation	SC11–1683	Ce manuel indique comment administrer une base de données DB2 pour des données textuelles. Il explique également comment utiliser les API (interfaces de programme d'application) fournies pour accéder aux données de ce type et les manipuler.

Partie 1. Introduction

Cette partie présente l'Extension XML et vous indique comment l'utiliser dans vos applications de gestion.

Chapitre 1. Présentation de l'Extension XML

La famille IBM DB2 Extensions offre des solutions de gestion des données et métadonnées pour le traitement des données traditionnelles et non traditionnelles. L'Extension XML vous permet d'intégrer la puissance de DB2 Universal Database (DB2 UDB) à la souplesse du langage XML.

DB2 Extension XML permet de stocker des documents XML, de les générer à partir de données relationnelles existantes et de les broyer (c'est-à-dire de les décomposer, en stockant le contenu des éléments et des attributs sans les balises) en données relationnelles. L'Extension XML offre de nouveaux types de données, de nouvelles fonctions et procédures mémorisées pour la gestion des données XML dans DB2.

L'Extension XML est disponible pour les systèmes d'exploitation suivants :

- Windows NT
- AIX
- Sun Solaris
- Linux
- NUMA-Q

Documents XML

L'informatique dispose d'un large éventail d'applications, chacune ayant ses avantages et ses inconvénients. Les utilisateurs d'aujourd'hui ont la possibilité de choisir l'application qui répond le mieux à leurs besoins pour effectuer des tâches spécifiques. Toutefois, comme ils ont tendance à partager des données entre des applications distinctes, ils doivent continuellement faire face à la nécessité de répliquer, transformer, exporter ou sauvegarder ces données dans un format différent, pour pouvoir les importer dans une autre application. Il peut s'agir d'un inconvénient majeur dans le cas d'applications de gestion car de nombreux processus de transformation peuvent entraîner la disparition de données, ou exigent que l'utilisateur suive la procédure fastidieuse de vérification de la cohérence des données. C'est une perte de temps et d'argent.

Désormais, pour pallier cet inconvénient, les développeurs d'applications peuvent écrire des applications *ODBC (Open Database Connectivity)* pour sauvegarder les données dans un système de gestion de base de données. A ce stade, les données peuvent être manipulées et présentées sous la forme requise par une autre application. Les applications de base de données doivent être écrites pour convertir les données au format approprié.

Cependant, ces applications changent rapidement et deviennent obsolètes. Les applications de conversion HTML apportent des solutions de présentation, mais les données présentées ne peuvent quasiment pas être utilisées à d'autres fins. S'il existait une autre méthode permettant de séparer le contenu de la présentation, elle serait adoptée comme format d'échange pratique entre les applications.

XML est apparu pour apporter la solution. XML est l'acronyme de *eXtensible Markup Language* (langage de balisage extensible). Il est extensible du fait qu'il s'agit d'un métalangage qui vous permet de créer votre propre langage en fonction des besoins de votre entreprise. XML ne se contente pas de capturer les données destinées à une application, il capture aussi la structure de ces données. XML n'est pas le seul format d'échange sur le marché. Toutefois, c'est désormais la norme reconnue pour l'échange de données. En adhérant à cette norme, les applications peuvent enfin partager des données entre elles sans devoir les convertir au préalable à l'aide de formats propriétaires.

Applications XML

XML étant la norme reconnue pour l'échange de données, de nombreuses applications apparaissant sur le marché peuvent en tirer parti.

Supposons que vous utilisiez une application de gestion de projet et que vous vouliez partager certaines des données avec votre application d'agenda. Avec XML, cette opération est toute simple. Dans le monde interconnecté d'aujourd'hui, un fournisseur d'applications n'est pas compétitif s'il ne fournit pas les fonctions d'échange XML intégrées à ses applications. Ainsi, dans cet exemple, l'application de gestion de projet peut exporter des tâches au format XML, qui peuvent ensuite être importées en l'état dans l'application d'agenda (si les informations sont conformes à une DTD admise).

Pourquoi XML et DB2 ?

Même si le langage XML a permis de résoudre de nombreux cas par l'apport d'un format standard aux échanges de données, d'autres questions doivent encore être traitées. Lors de la construction d'une application de données d'entreprise, vous devez répondre à des questions de type :

- Combien de fois faut-il répliquer les données ?
- Quel type d'informations doit être partagé entre les applications ?
- Comment rechercher rapidement l'information dont j'ai besoin ?
- Comment obtenir qu'une action, telle que l'ajout d'une entrée, déclenche automatiquement un échange de données entre toutes mes applications ?

Les considérations de ce type sont du ressort exclusif d'un système de gestion de base de données. En intégrant les informations et méta-informations XML

directement dans la base de données, vous pouvez obtenir directement (et plus rapidement) les résultats XML requis par d'autres applications. C'est là qu'intervient l'Extension XML. Avec l'Extension XML, vous pouvez bénéficier de la puissance de DB2 dans de nombreuses applications XML.

Avec le contenu des documents créés avec XML dans une base de données DB2, vous pouvez combiner des informations de type XML avec des données relationnelles classiques. En fonction de l'application, vous pouvez stocker des documents XML entiers dans DB2 sous forme de type de données défini par l'utilisateur (non traditionnel) ou convertir le contenu XML en données traditionnelles de tables relationnelles. Dans le cas de données XML non traditionnelles, l'Extension XML permet d'examiner les types de données ″riches″ des éléments ou des valeurs d'attributs XML, outre la recherche structurelle fournie par DB2 UDB Extension Texte.

Avec l'Extension XML, votre application peut effectuer les tâches suivantes :

- Stockage de documents XML entiers sous forme de *données de colonne* dans une table d'application ou de manière externe sous forme de fichier local, avec extraction des valeurs d'attributs ou d'éléments XML souhaitées pour leur insertion dans des *tables annexes* en vue d'une recherche. En mode colonne XML, vous pouvez :
  - effectuer des recherches rapides sur des attributs ou des éléments XML associés à des *types de données* SQL généraux, extraits pour insertion dans des tables annexes et indexés,
  - mettre à jour le contenu d'un *élément XML* ou la valeur d'un *attribut XML*,
  - extraire des attributs ou des éléments XML de manière dynamique dans une requête SQL,
  - valider des documents XML lors d'opérations d'insertion et de mise à jour,
  - effectuer des recherches structurelles à l'aide de l'Extension Texte.
- Composition ou décomposition de documents XML à l'aide d'une ou plusieurs tables relationnelles, par la méthode d'accès et de stockage liée aux collections XML.

Intégration de documents XML dans DB2

L'Extension XML comporte les fonctions suivantes facilitant la gestion et l'exploitation des données XML avec DB2 :

- Outils d'administration vous aidant à intégrer des données XML dans des tables relationnelles.
- Méthodes d'accès et de stockage applicables aux données XML.

- Référentiel des DTD dans lequel vous stockez les DTD permettant de valider des données XML : DTD_REF.
- Un schéma de mappage, appelé fichier DAD (définition d'accès au document), permettant de mapper les documents XML vers des données relationnelles.

Outils d'administration

Les outils d'administration de l'Extension XML permettent d'activer la base de données et les colonnes de table pour XML et de mapper les données XML vers les structures relationnelles DB2. Vous les utilisez en fonction de vos besoins : développement d'une application en vue de l'exécution de tâches d'administration, ou simple utilisation d'un assistant. Les outils à votre disposition pour l'exécution des tâches d'administration de l'Extension XML sont les suivants :

- Les assistants d'administration de l'Extension XML offrent une interface graphique destinée aux tâches d'administration.
- La commande **dxxadm** comporte une option que vous pouvez entrer à partir de la ligne de commande.
- Les procédures mémorisées d'administration de l'Extension XML comprennent également des options de développement d'applications dédiées aux tâches d'administration.

Méthodes d'accès et de stockage

L'Extension XML comporte deux méthodes d'accès et de stockage pour l'intégration des documents XML dans DB2 : la colonne et la collection XML. Ces deux méthodes ont des utilisations très différentes, mais peuvent cohabiter dans la même application.

Colonne XML

Cette méthode permet de stocker des documents XML entiers dans DB2. Elle est particulièrement adaptée à l'archivage des documents. Les documents sont insérés dans des colonnes activées pour XML, puis peuvent faire l'objet d'opérations de mise à jour, d'extraction ou de recherche. Les données d'éléments et d'attributs peuvent être mappées vers des tables DB2 (tables annexes) qui, à leur tour, peuvent être indexées pour des recherches structurelles rapides.

Collection XML

Cette méthode permet de mapper des structures de documents XML vers des tables DB2 pour composer des documents XML à partir de données DB2 existantes, ou inversement, pour décomposer des documents XML en données DB2 (c'est-à-dire stocker le contenu des éléments ou des attributs sans les balises). Cette méthode est particulièrement adaptée aux applications d'échange de données, notamment lorsque le contenu des documents XML est fréquemment mis à jour.

Référentiel des DTD

L'Extension XML comporte un *référentiel des DTD (définition de type de document)*, qui constitue un ensemble de déclarations pour les attributs et éléments XML. Lors de l '*activation* d'une base de données pour XML, une *table de référence des DTD (DTD_REF)* est créée. Chaque ligne de cette table représente une DTD accompagnée d'autres informations sur les métadonnées. Les utilisateurs peuvent accéder à cette table pour y insérer leurs propres DTD. Les *DTD* de la table DTD_REF permettent de valider des documents XML.

Définitions DAD

Vous indiquez dans une *définition d'accès au document (DAD)* le mode selon lequel les documents XML structurés doivent être traités. Cette DAD est elle-même un document au format XML. Elle associe la structure des documents XML à une base de données DB2 lors de l'utilisation des colonnes ou des collections XML. La structure de la DAD varie selon que vous définissez une colonne ou une collection XML.

Les fichiers DAD sont gérés à l'aide de la table XML_USAGE, qui est créée lorsque vous activez une base de données pour XML.

Colonne XML : extraction et stockage de documents structurés

XML contenant toutes les informations nécessaires à la création d'un ensemble de documents, vous pouvez stocker et conserver la structure du document en l'état.

Par exemple, si vous faites partie de la rédaction d'un magazine qui publie des articles sur le Web, vous souhaiterez peut-être archiver les articles publiés. Dans ce cas, l'Extension XML vous permet de stocker des articles XML entiers ou partiels dans une colonne de table DB2. Ce type de stockage de document XML est appelé *colonne XML*, comme le montre la figure 1 à la page 8.

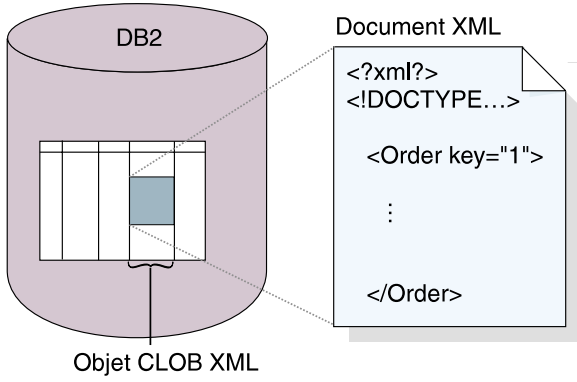


Figure 1. Stockage de documents XML structurés dans une colonne de table DB2

L'Extension XML comporte les types UDT (définis par l'utilisateur) suivants à utiliser avec les colonnes XML :

- XMLVarchar
- XMLCLOB
- XMLFILE

Tous les types UDT de l'Extension XML portent le préfixe db2xml, correspondant au *nom de schéma*. Ces types de données servent à identifier le type de stockage des documents XML dans la table d'application. L'Extension XML prend en charge les fichiers à plat existants. Vous n'avez pas besoin de stocker les documents XML dans DB2.Vous pouvez également stocker des documents XML sous forme de fichiers dans le *système de fichiers local*, indiqué par un nom de fichier local.

DB2 Extension XML offre des *fonctions UDF (définies par l'utilisateur)* performantes pour stocker des documents dans les colonnes XML ou en extraire, ainsi que pour extraire des valeurs d'attributs ou d'éléments XML. Une fonction UDF est définie sur le système de gestion de base de données et peut être utilisée par la suite dans des requêtes SQL. L'Extension XML comporte les types suivants de fonctions UDF :

- Stockage : Stocke des documents XML entiers, en tant que types de données XML, dans des colonnes activées pour XML.
- Extraction : Extrait des documents XML, ou les valeurs d'éléments ou d'attributs indiquées, en tant que types de données de base.
- Mise à jour : Met à jour des documents XML entiers ou les valeurs d'attributs ou d'éléments indiquées.

Les fonctions d'extraction permettent d'effectuer des recherches puissantes sur des types de données SQL généraux. Par ailleurs, vous pouvez utiliser DB2 UDB Extension Texte avec l'Extension XML pour effectuer des *recherches*

intégrales ou structurelles dans un document XML. Cette fonction de recherche très performante peut être utilisée, par exemple, pour améliorer l'utilisation d'un site Web publiant de grandes quantités de texte, tels que des articles de journaux ou des applications *EDI (Echange de données informatisées)*, qui contiennent souvent des attributs ou des éléments consultables.

Toutes les fonctions UDF de l'Extension XML portent le préfixe db2xml, correspondant au nom de schéma. Les fonctions UDF sont appliquées aux types UDT XML et utilisées dans les colonnes XML.

Chemin d'emplacement

Un *chemin d'emplacement* est une séquence de *balises XML* identifiant un attribut ou un élément XML. L'Extension XML utilise le chemin d'emplacement pour identifier la structure du document XML, indiquant le contexte de l'élément ou de l'attribut. Un chemin avec une barre oblique simple (/) indique que le contexte est constitué du document entier. Le chemin d'emplacement est utilisé dans les cas suivants, pour :

- identifier les éléments et les attributs à extraire, lors de l'utilisation des fonctions UDF d'extraction,
- indiquer le fichier de mappage entre un attribut ou un élément XML et une colonne DB2, lors de la définition du schéma d'indexation dans la DAD de colonne XML,
- identifier un élément ou un attribut XML lors de l'utilisation de la fonction de recherche structurelle de l'Extension Texte.

La figure 2 à la page 10 représente un exemple de chemin d'emplacement et de ses relations avec la structure du document.

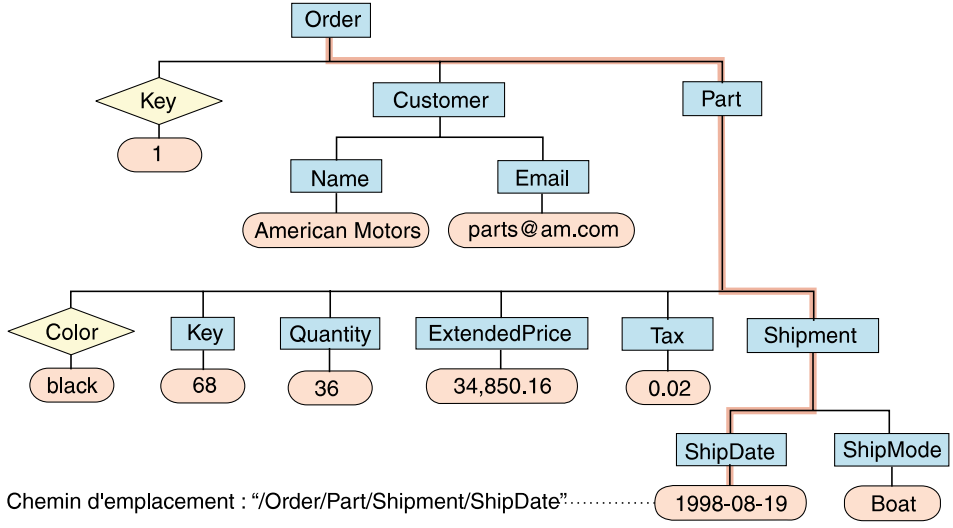


Figure 2. Stockage de documents sous forme de documents XML structurés dans une colonne de table DB2

Pour préciser le chemin d'emplacement, l'Extension XML utilise un sous-ensemble de *XSLT (XML Stylesheet Language Transformation)* et de *XPath (XML Path Language)*. Le présent manuel emploie le terme *chemin d'emplacement*, défini dans la spécification de XPath. Le chemin d'emplacement est une séquence de balises XML identifiant un attribut ou un élément XML. Ce manuel utilise également la syntaxe abrégée de XSLT ou XPath pour le *chemin d'emplacement absolu*, défini dans les spécifications XPath. Le chemin d'emplacement absolu désigne le chemin d'accès complet d'un objet.

XSLT est un langage de transformation de documents XML en d'autres documents XML. Il est conçu pour être utilisé comme une partie de *XSL (XML Stylesheet Language)*, langage de feuille de style pour XML. Parallèlement à XSLT, XSL comprend un vocabulaire XML de spécification de la mise en forme. XSL précise le style d'un document XML à l'aide de XSLT pour indiquer comment le document est transformé en un autre document XML utilisant le vocabulaire de mise en forme.

XPath est un langage qui permet de se rapporter à des parties d'un document XML, devant être utilisé par XSLT. Tous les chemins d'emplacement peuvent être exprimés à l'aide de la syntaxe définie pour XPath.

Pour plus d'informations sur XSLT et XPath, consultez les pages Web suivantes :

- Pour XSLT, reportez-vous à : http://www.w3.org/TR/WD-xslt
- Pour XPath, reportez-vous à : http://www.w3.org/TR/xpath

Pour consulter la syntaxe et les restrictions applicables, reportez-vous à la section «Chemin d'emplacement» à la page 55.

Terminologie relative aux colonnes XML

Cette section décrit les concepts et la terminologie XML cités dans le présent manuel.

DAD (définition d'accès à un document)
: En mode colonne XML, mappage d'une structure de document XML vers des tables annexes DB2 indexées pour les recherches structurelles.

DXX_INSTALL
: Répertoire d'installation de l'Extension XML.

Table annexe
: Tables complémentaires créées par l'Extension XML pour améliorer les performances lors de la recherche d'éléments ou d'attributs dans une colonne XML.

Colonne XML
: Méthode permettant d'accéder à des documents XML après l'activation d'une colonne DB2 pour les types de données XML et de stocker un document XML entier dans la colonne activée. Se rapporte également à une colonne activée pour XML à l'aide de l'un des outils d'administration.

Table XML
: Table d'application contenant une ou plusieurs colonnes activées pour XML à l'aide de l'un des outils d'administration.

Fonction UDF XML
: Fonction UDF DB2 fournie par l'Extension XML.

Type de données UDT XML
: Type de données utilisateur DB2 fourni par l'Extension XML.

Collection XML : gestion de données intégrée

Les données SQL traditionnelles sont soit *décomposées* à partir de documents XML entrants ou utilisées pour *composer* des documents XML sortants. Si vos données doivent être partagées avec d'autres applications, vous voulez peut-être composer et décomposer des documents XML entrants et sortants et gérer les données en conséquence pour tirer parti des fonctions relationnelles de DB2. Ce type de stockage de document XML est appelé *collection XML*.

La figure 3 à la page 12 présente un exemple de collection XML.

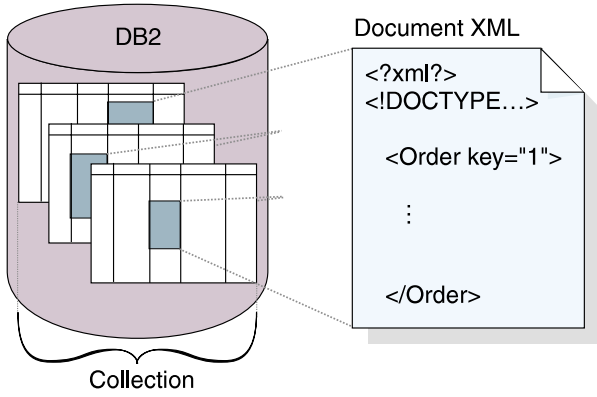


Figure 3. Stockage de documents sous forme de données non balisées dans des tables DB2

La collection XML est définie dans un fichier DAD, qui indique le mode de mappage des éléments et des attributs vers une ou plusieurs tables relationnelles. Pour définir une collection, vous l'activez puis vous l'utilisez avec les procédures mémorisées pour la composition ou décomposition de documents XML.

Lorsque vous définissez une collection dans le fichier DAD, vous vous servez de l'un des deux schémas de mappage disponibles, le mappage SQL et le mappage de noeud RDB. Le mappage SQL fait appel à des instructions SQL SELECT pour définir les tables et les conditions DB2 à utiliser pour la collection. Le mappage du noeud RDB définit les tables, les colonnes et les conditions à l'aide de l'élément RDB_node basé sur XPath.

Les procédures mémorisées permettent de composer ou de décomposer des documents XML. Elles ont recours au préfixe db2xml, qui constitue le *nom de schéma* de l'Extension XML. Utilisez les procédures mémorisées suivantes avec les collections XML :

- Composition :
  - dxxGenXML() : compose des documents XML avec un fichier DAD pour collection XML.
  - dxxRetrieveXML() : compose des documents XML avec une collection XML activée.
- Décomposition :
  - dxxShredXML() : décompose des documents XML avec un fichier DAD pour collection XML.
  - dxxInsertXML() : décompose des documents XML avec une collection XML activée.

Terminologie relative aux collections XML

Les termes suivants sont propres à l'Extension XML et apparaissent fréquemment dans le présent manuel.

Composition
Génération de documents XML à partir de données relationnelles existantes, conformément à une définition DAD.

Décomposition
Stockage de documents XML sous forme de données relationnelles non balisées, conformément à une définition DAD.

DAD (définition d'accès à un document)
En mode collection XML, mappage des structures de documents XML vers des structures de données DB2 pour la composition ou la décomposition des documents XML.

DXX_INSTALL
Répertoire d'installation de l'Extension XML.

Collection XML
Méthode de stockage et d'accès relative aux données XML ayant recours à un ensemble de tables relationnelles. Les données non balisées peuvent être composées en documents XML, ou décomposées à partir de ceux-ci. Désigne également l'ensemble de tables dans lesquelles ou à partir desquelles les documents XML sont composés et décomposés.

Procédures mémorisées XML
Procédures mémorisées permettant de composer ou de décomposer des documents XML.

Chapitre 2. Initiation à DB2 Extension XML

Ce chapitre vous indique comment débuter avec Extension XML pour accéder à des données XML et les modifier pour vos applications. Le cours de formation permet d'installer une base de données à l'aide des exemples de données fournis, de mapper des données SQL vers un document XML, de stocker des documents XML dans la base de données, puis d'effectuer des opérations de recherche et d'extraction sur ces documents XML.

Dans le cours, vous utilisez la fenêtre de commande DB2 avec des commandes d'administration. Vous pouvez exécuter ces tâches à l'aide de l'assistant d'administration, également décrit dans le manuel. Dans le cours d'initiation à la gestion des données XML, vous faites appel aux fonctions UDF (définies par l'utilisateur) et aux procédures mémorisées de l'Extension XML. La plupart des exemples cités dans les pages suivantes sont basés sur des exemples fournis dans le présent chapitre.

Logiciels requis : Pour pouvoir exécuter les tâches expliquées dans ce chapitre, vous devez disposer de DB2 UDB version 6.1 ou suivante. Si vous utilisez la version 6.1, vous devez installer le kit de mise à jour 2. Par ailleurs, nous supposons que votre système d'exploitation est Windows NT®.

Les tâches d'initiation sont les suivantes :

- Stockage d'un document XML entier dans une colonne de table DB2
  - Planification du type UDT XML à utiliser pour le stockage du document, des éléments et des attributs XML en vue de recherches fréquentes
  - Configuration de la base de données et des tables
  - Activation de la base de données pour XML
  - Insertion de la DTD dans le référentiel des DTD
  - Préparation de la DAD pour une colonne XML
  - Ajout d'une colonne dans une table existante de type XML
  - Activation de la nouvelle colonne pour XML
  - Indexation des tables annexes
  - Stockage d'un document XML dans la colonne XML
  - Recherche dans la colonne XML avec les fonctions UDF de l'Extension XML
- Création d'un document XML à partir de données existantes
  - Planification de la structure de données du document XML
  - Configuration de la base de données et des tables

  - Activation de la base de données pour XML
  - Préparation d'une DAD de collection XML
  - Composition du document XML à partir de données existantes
  - Extraction du document XML à partir de la base de données
- Nettoyage de la base de données

Scénario de formation

Dans ce cours, nous supposons que vous travaillez pour ACME Auto Direct, société de distribution de véhicules de tourisme et d'utilitaires aux concessionnaires automobiles. On vous a confié deux tâches. La première consiste à configurer un système permettant d'archiver des commandes dans la base SALES_DB pour que le service commercial puisse les consulter. La seconde consiste à examiner une base de commandes existante (SALES_DB) pour y extraire les informations de clé à stocker dans des documents XML.

Formation : stockage d'un document XML dans une colonne XML

Scénario

Vous êtes chargé d'archiver les données relatives aux ventes pour le service de maintenance. Les données sont stockées dans des documents XML utilisant la même DTD (définition du type de document). Le service de maintenance aura recours à ces documents XML lors du traitement des demandes et des réclamations clients.

Le service de maintenance a recommandé une structure de document XML et précisé les données d'éléments qui seront probablement les plus demandées. Il voudrait que les documents XML soient stockés dans la table SALES_TAB de la base de données SALES_DB et pouvoir y effectuer des recherches rapides. La table SALES_DB comportera deux colonnes de données sur chaque vente et une troisième colonne pour le stockage du document XML. Cette colonne s'appelle ORDER.

Vous déterminerez les types de données XML sous lequel le document XML devra être stocké, ainsi que les éléments et les attributs XML fréquemment demandés. Ensuite, vous allez configurer la base de données SALES_DB pour XML, créer la table SALES_TAB et activer la colonne ORDER pour pouvoir stocker le document entier dans DB2. Vous allez également insérer une DTD pour la validation du document XML, puis stocker celui-ci sous le type de données XMLVARCHAR. Lors de l'activation de la colonne, vous allez définir dans un fichier DAD (définition d'accès au document) les tables annexes à indexer pour les recherches structurelles. Ce fichier DAD est un document

XML qui précise la structure des tables annexes. Pour consulter des exemples de fichier DAD, de définition DTD et de document XML, reportez-vous à l'«Annexe B. Exemples» à la page 277.

La table SALES_TAB est décrite au tableau 1.

Tableau 1. Table SALES_TAB

Nom de colonne	Type de données
INVOICE_NUM	CHAR(6) NOT NULL PRIMARY KEY
SALES_PERSON	VARCHAR(20)
ORDER	XMLVARCHAR

Planification

Avant de stocker un document XML avec l'Extension XML, vous devez comprendre sa structure pour déterminer le mode de recherche à utiliser. A cet effet, vous devez déterminer les points suivants :

- le type UDT (défini par l'utilisateur) XML sous lequel vous allez stocker le document XML,
- les éléments et les attributs XML qui seront fréquemment examinés par le service de maintenance, pour pouvoir les indexer en vue d'améliorer les performances de recherche.

Les sections ci-après indiquent la procédure à suivre.

Structure du document XML

Pour ce cours, la structure de document XML reçoit des informations relatives à une commande client déterminée, structurée au niveau supérieur par le numéro de commande et au niveau suivant, par les informations sur le client, la pièce et la livraison. Le document XML est décrit figure 4 à la page 18.

Ce cours fournit également un exemple de DTD permettant de comprendre et de valider la structure du document XML. Le fichier DTD se trouve à l'«Annexe B. Exemples» à la page 277. Il correspond à la structure illustrée figure 4 à la page 18.

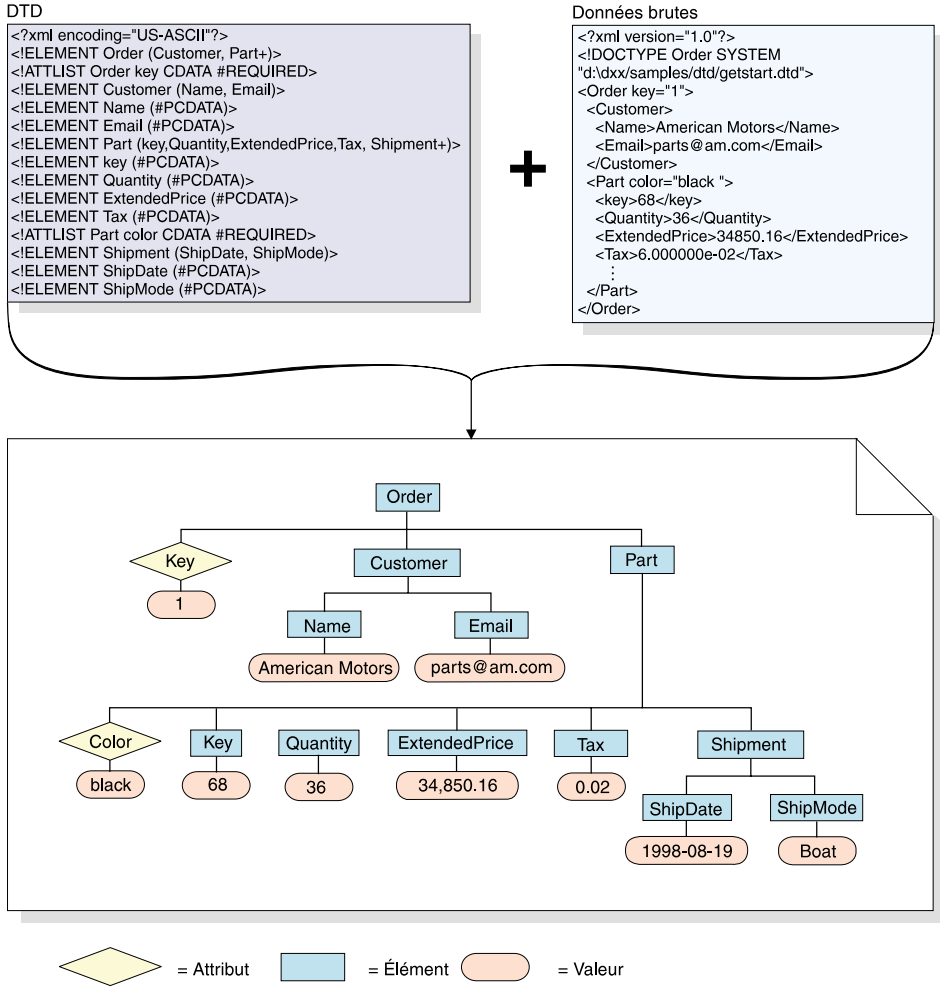


Figure 4. Structure hiérarchique de la DTD et du document XML

Détermination du type de données de colonne XML

L'Extension XML comporte des types de données utilisateur XML sous lesquels vous définissez une colonne qui doit contenir des documents XML. Ces types de données sont les suivants :

- XMLVarchar : pour les petits documents stockés dans DB2
- XMLCLOB : pour les gros documents stockés dans DB2
- XMLFILE : pour les documents stockés hors DB2

Dans ce cours, vous allez stocker un petit document dans DB2 et, par conséquent, utiliser le type de données XMLVarchar.

Détermination des éléments et des attributs à examiner

Lorsque vous comprenez la structure du document XML et les besoins de l'application, vous pouvez déterminer les éléments et les attributs à examiner, ceux qui feront l'objet de recherches fréquentes ou longues. Le service de maintenance a indiqué qu'il demandera souvent la clé de la commande, le nom du client, le prix et la date de livraison, et qu'il aura besoin d'effectuer des recherches rapides. Ces informations sont contenues dans les éléments et les attributs de la structure de document XML. La tableau 2 décrit les chemins d'emplacement de chaque élément et attribut.

Tableau 2. Eléments et attributs à examiner

Données	Chemin d'emplacement
Clé de la commande	/Order/@key
Client	/Order/Customer/Name
Prix	/Order/Part/ExtendedPrice
Date de livraison	/Order/Part/Shipment/ShipDate

Mappage du document XML vers les tables annexes

Dans ce tutoriel, vous allez créer un fichier DAD pour la colonne XML qui permet de stocker le document XML dans DB2. Elle permet également de mapper le contenu des éléments et des attributs XML vers des tables annexes DB2 utilisées pour l'indexation, ce qui améliore les performances de recherche. Dans la dernière section, vous avez identifié les éléments et les attributs à examiner. Dans la présente section, vous obtenez davantage d'informations sur le mappage de valeurs d'éléments et d'attributs vers des tables DB2 pouvant être indexées.

Après l'identification des éléments et des attributs à examiner, vous déterminez leur disposition dans les tables annexes, le nombre de tables et les colonnes appartenant à chaque table. En général, pour organiser les tables annexes, vous regroupez des informations apparentées dans une même table. La structure dépend également de la possibilité ou non de répéter le chemin d'emplacement des éléments dans un document. Par exemple dans votre document, l'élément part (pièce) peut être répété et, par conséquent, les éléments price (prix) et date peuvent avoir des occurrences multiples. Les éléments qui peuvent être répétés doivent figurer dans leurs propres tables.

En outre, vous avez également besoin de déterminer les types DB2 de base que les valeurs d'éléments ou d'attributs devront utiliser. En général, ces types sont facilement indiqués par le format des données. Choisissez VARCHAR, INTEGER ou DATE, selon qu'il s'agit respectivement de texte, d'entiers numériques ou d'une date pour effectuer des recherches d'après des plages de valeurs.

Dans ce tutoriel, les éléments et les attributs sont mappés vers les tables annexes suivantes :

ORDER_SIDE_TAB

Nom de colonne	Type de données	Chemin d'emplacement	Occurrences multiples ?
ORDER_KEY	INTEGER	/Order/@key	Non
CUSTOMER	VARCHAR(16)	/Order/Customer/Name	Non

PART_SIDE_TAB

Nom de colonne	Type de données	Chemin d'emplacement	Occurrences multiples ?
PRICE	DECIMAL(10,2)	/Order/Part/ExtendedPrice	Oui

SHIP_SIDE_TAB

Nom de colonne	Type de données	Chemin d'emplacement	Occurrences multiples ?
DATE	DATE	/Order/Part/Shipment/ShipDate	Oui

Pour ce tutoriel, nous fournissons un jeu de scripts pour la configuration de votre environnement. Ces scripts se trouvent dans le répertoire *DXX_INSTAL*\samples\cmd (*DXX_INSTALL* désignant l'unité et le répertoire d'installation de l'Extension XML, par exemple c:\dxx\samples\cmd). Ils se présentent comme suit :

getstart_db.cmd
Crée la base de données et remplit quatre tables.

getstart_prep.cmd
Définit les accès de la base de données aux procédures mémorisées de l'Extension XML et à l'interface CLI de DB2.

getstart_insertDTD.cmd
Insère la DTD utilisée pour valider le document XML dans la colonne XML.

getstart_createTabCol.cmd
Crée une table d'application avec une colonne activée pour XML.

getstart_alterTabCol.cmd
Modifie la table d'application en ajoutant la colonne qui sera activée pour XML.

getstart_enableCol.cmd
Active la colonne XML.

getstart_createIndex.cmd
Crée des index sur les tables annexes pour la colonne XML.

getstart_insertXML.cmd
Insère le document XML dans la colonne XML.

getstart_queryCol.cmd
Exécute une instruction SELECT sur la table d'application et renvoie le document XML.

getstart_clean.cmd
Nettoie l'environnement du tutoriel.

Configuration

Dans cette section, vous allez préparer la base de données pour l'utiliser avec l'Extension XML. Vous allez exécuter les tâches suivantes :

1. Création de la base de données.
2. Activation de la base de données.

Création de la base de données

Dans cette section, vous allez configurer la base de données par une commande. Cette commande crée un exemple de base de données, vous y connecte, créé les tables qui contiendront les données, puis insère celles-ci.

Pour créer la base de données, procédez comme suit :

1. Placez-vous dans le répertoire *DXX_INSTALL*\samples\cmd, *DXX_INSTALL* désignant l'unité et le répertoire d'installation de l'Extension XML. Ce cours utilise le répertoire c:\dxx. Modifiez ces valeurs si vous avez recours à une unité et à un répertoire différents.
2. Ouvrez la fenêtre de commande DB2 à partir du menu Démarrage de Windows NT, ou entrez la commande suivante à partir de l'invite de commande Windows NT :

   ```
   DB2CMD
   ```

3. A partir de la fenêtre de commande DB2, lancez la commande suivante :

   ```
   getstart_db.cmd
   ```

Activation de la base de données

Pour pouvoir stocker des informations XML dans la base de données, vous devez d'abord activer celle-ci pour l'Extension XML. Lorsque vous activez une base de données pour XML, DB2 Extension XML exécute les opérations suivantes :

- Création de tous les types (UDT) et fonctions (UDF) définis par l'utilisateur.

- Création et peuplement des tables de contrôle avec les métadonnées requises par l'Extension XML.
- Création du schéma db2xml et affectation des privilèges nécessaires.

Pour activer la base de données pour XML, procédez comme suit :

A partir de la fenêtre de commande DB2, lancez le script suivant pour activer la base de données SALES_DB :

```
getstart_prep.cmd
```

Ce script définit les accès de la base de données aux procédures mémorisées de l'Extension XML et à l'interface CLI de DB2. Il exécute également l'option de la commande **dxxadm** qui permet d'activer la base de données.

```
dxxadm enable_db VENTES_BD
```

Création de la colonne XML

L'Extension XML offre une méthode d'accès et de stockage, appelée colonne XML, applicable à des documents XML entiers. Cette méthode permet de stocker un document en lui affectant des types de fichiers XML, d'indexer la colonne dans des tables annexes, puis de lancer des recherches dans ce document XML. Elle est particulièrement utile pour les applications d'archivage dans lesquelles les documents ne sont pas souvent mis à jour. Dans ce tutoriel, vous allez stocker dans la colonne XML le document XML fourni.

Dans ce cours, vous allez stocker le document dans la table SALES_TAB. Pour ce faire, vous allez exécuter les tâches suivantes :

1. Insertion de la DTD du document XML dans la table de référence des DTD, DTD_REF.
2. Préparation du fichier DAD indiquant l'emplacement du document XML et des tables annexes pour les recherches structurelles.
3. Ajout dans la table SALES_TAB d'une colonne de type UDT XML XMLVARCHAR.
4. Activation de la colonne pour XML.
5. Indexation des tables annexes pour les recherches structurelles.
6. Stockage du document à l'aide d'une fonction UDF de l'Extension XML.

Insertion de la DTD dans le référentiel des DTD

Vous pouvez valider les données XML dans une colonne XML à l'aide d'une DTD. L'Extension XML crée une table, DTD_REF, dans la base de données activée pour XML. Cette table, désignée table de référence des DTD, est mise à votre disposition pour le stockage des DTD. Lorsque vous validez des documents XML, vous devez stocker la DTD dans ce référentiel. Dans ce tutoriel, la DTD s'appelle `c:\dxx\samples\dtd\getstart.dtd`.

Pour insérer la DTD, procédez comme suit :

A partir de la fenêtre de commande DB2, entrez la commande SQL INSERT suivante sur une même ligne :

```
DB2 CONNECT TO SALES_DB
DB2 INSERT into db2xml.dtd_ref values('c:\dxx\samples\dtd\getstart.dtd',
    db2xml.XMLClobFromFile('c:\dxx\samples\dtd\getstart.dtd'), 0, 'user1',
    'user1', 'user1')
```

Vous pouvez également insérer la DTD en exécutant le fichier de commandes suivant :

```
getstart_insertDTD.cmd
```

Préparation du fichier DAD

Le fichier DAD pour la colonne XML a une structure simple. Vous indiquez le mode de stockage colonne XML et vous définissez les tables et les colonnes pour l'indexation.

Dans la procédure suivante, les éléments de la DAD sont appelés *balises* et ceux de la structure du document XML, *éléments*. Vous trouverez un fichier DAD exemple, identique à celui que vous allez créer, dans `c:\dxx\samples\dad\getstart_xcolumn.dad`. Cet exemple comporte quelques légères différences par rapport au fichier qui sera généré au cours des étapes suivantes. Si vous l'utilisez pour le cours, notez que les chemins d'accès aux fichiers peuvent différer de ceux de votre environnement et que la valeur OUI de l'élément <validation> est remplacée par NON.

Pour préparer le fichier DAD, procédez comme suit :

1. Ouvrez un éditeur de texte et nommez le fichier `getstart_xcolumn.dad`.

   Notez que la distinction entre les majuscules et les minuscules est prise en compte dans toutes les balises employées dans le fichier DAD.
2. Créez l'en-tête de la DAD avec les déclarations XML et Doctype.

   ```
   <?xml version="1.0"?>
   <!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
   ```

   Le fichier DAD est un document XML pour lequel des déclarations XML sont nécessaires.
3. Insérez des balises de début et de fin <DAD></DAD>. Toutes les autres balises sont imbriquées dans ces balises DAD.
4. A l'aide des balises de début et de fin <DTDID></DTDID>, précisez l'identificateur de DTD qui associe la DAD à la DTD du document XML et indique l'emplacement de la DTD sur le client.

   ```
   <dtdid>c:\dxx\samples\dtd\getstart.dtd</dtdid>
   ```

Vérifiez que cette chaîne correspond à la valeur utilisée comme première valeur de paramètre lors de l'insertion de la DTD dans la table de référence des DTD, à la section «Insertion de la DTD dans le référentiel des DTD» à la page 22. Par exemple, le chemin utilisé pour l'ID DTD peut être différent de la chaîne ci-dessus si vous travaillez sur une autre unité machine.

5. A l'aide des balises de début et de fin <validation></validation>, indiquez que l'Extension XML doit valider la structure du document XML conformément à la DTD placée dans le référentiel des DTD.

   ```
   <validation>YES</validation>
   ```

   La valeur de l'élément <validation> doit être en majuscules.

6. A l'aide des balises de début et de fin <Xcolumn></Xcolumn>, définissez la méthode de stockage colonne XML. Cette méthode indique que les données XML doivent être stockées dans une colonne XML.

   ```
   <Xcolumn>
   </Xcolumn>
   ```

7. Insérez des balises de début et de fin <table></table> pour chaque table annexe à générer.

   ```
   <Xcolumn>
        <table name="order_side_tab">
   </table>
        <table name="part_side_tab">
   </table>
        <table name="ship_side_tab">
   </table>
   </Xcolumn>
   ```

8. Insérez des balises de début et de fin <column></column> pour chaque colonne à inclure dans les tables annexes. Chaque balise <column> comporte quatre attributs :

   - **name :** nom de colonne
   - **type :** type de données de la colonne
   - **path :** chemin d'emplacement de l'élément correspondant dans le document XML, suivant la syntaxe XPath. Pour consulter la syntaxe du chemin d'emplacement, reportez-vous à la section «Chemin d'emplacement» à la page 55.

- **multi-occurrence :** cet attribut indique si le chemin d'emplacement de l'élément peut être répété dans la structure du document XML.

```
<Xcolumn>
     <table name="order_side_tab">
         <column name="order_key"
           type="integer"
           path="/Order/@key"
          multi_occurrence="NO"/>
         <column name="customer"
           type="varchar(50)"
           path="/Order/Customer/Name"
          multi_occurrence="NO"/>
</table>
     <table name="part_side_tab">
         <column name="price"
            type="decimal(10,2)"
            path="/Order/Part/ExtendedPrice"
          multi_occurrence="YES"/>
</table>
     <table name="ship_side_tab">
         <column name="date"
            type="DATE"
            path="/Order/Part/Shipment/ShipDate"
          multi_occurrence="YES"/>
</table>
</Xcolumn>
```

9. Vérifiez que vous avez placé une balise de fin </Xcolumn> après la dernière balise </table>.
10. Vérifiez que vous avez placé une balise de fin </DAD> après la balise </Xcolumn>.
11. Sauvegardez le fichier sous le nom `getstart_xcolumn.dad`.

Vous pouvez comparer le fichier que vous venez de créer avec le fichier exemple, `c:\dxx\samples\dad\getstart_xcolumn.dad`. Il s'agit d'une copie de travail du fichier DAD nécessaire pour l'activation de la colonne XML et la création des tables annexes. Le fichier exemple contient peut-être des instructions relatives aux chemins d'accès (path) qui doivent être adaptées à votre environnement pour pouvoir s'exécuter.

Création de la table SALES_TAB

Dans cette section, vous allez créer la table SALES_TAB. A l'origine, cette table comporte deux colonnes d'informations commerciales relatives à la commande client.

Pour créer la table, procédez comme suit :

A partir de la fenêtre de commande DB2, entrez l'instruction CREATE TABLE suivante :

```
DB2 CONNECT TO SALES_DB

DB2 CREATE TABLE SALES_TAB(INVOICE_NUM CHAR(6) NOT NULL PRIMARY KEY,
        SALES_PERSON VARCHAR(20))
```

Vous pouvez également créer la table en exécutant le fichier de commandes suivant :

```
getstart_createTabCol.cmd
```

Ajout d'une colonne de type XML

A présent, vous allez ajouter une colonne dans la table SALES_TAB. Cette colonne est destinée à contenir le document XML entier généré précédemment et doit être du type UDT XML. L'Extension XML fournit plusieurs types de données, décrits au «Chapitre 8. Types de données UDT de l'Extension XML» à la page 173. Dans ce cours, vous allez stocker le document en tant que XMLVARCHAR.

Pour ajouter la colonne de type XML, procédez comme suit :

A partir de la fenêtre de commande DB2, entrez l'instruction SQL suivante :

```
DB2 ALTER TABLE SALES_TAB ADD ORDER DB2XML.XMLVARCHAR
```

Vous pouvez également modifier la table en exécutant le fichier de commandes suivant :

```
getstart_alterTabCol.cmd
```

Activation de la colonne XML

Une fois la colonne XML créée, vous allez l'activer pour l'Extension XML. Lorsque vous activez la colonne, l'Extension XML lit le fichier DAD et crée les tables annexes. Au préalable, vous devez :

- Déterminer si vous créez une vue par défaut de la colonne XML (qui contient le document XML) et des colonnes de tables annexes. Vous pouvez indiquer la vue par défaut lorsque vous effectuez une recherche dans le document XML. Dans ce cours, vous allez préciser cette option à l'aide du paramètre `-v`.

- Déterminer si vous indiquez une clé primaire en tant qu'ID racine (*ROOT ID*), le nom de colonne de la clé primaire appartenant à la table d'application et un identificateur unique associant toutes les tables annexes à cette table d'application. Si vous n'indiquez pas de clé primaire, l'Extension XML ajoute la colonne DXXROOT_ID dans la table d'application et dans les tables annexes. La colonne ROOT_ID associe la table d'application et les tables annexes, permettant à l'Extension XML d'actualiser automatiquement ces dernières lorsque le document XML est mis à jour. Dans ce cours, vous allez indiquer le nom de la clé primaire en plaçant le paramètre `-r` dans la commande (INVOICE_NUM). L'Extension XML utilisera la colonne indiquée en tant que ROOT_ID et l'ajoutera dans les tables annexes.
- Déterminer si vous utilisez l'espace table de votre choix ou celui par défaut. Dans ce cours, vous allez utiliser l'espace table par défaut.

Pour activer la colonne pour XML, procédez comme suit :

A partir de la fenêtre de commande DB2, entrez la commande suivante :

```
dxxadm enable_column SALES_DB SALES_TAB ORDER GETSTART_XCOLUMN.DAD
        -v SALES_ORDER_VIEW -r INVOICE_NUM
```

Vous pouvez également activer la colonne pour XML en exécutant le fichier de commandes suivant :

```
getstart_enableCol.cmd
```

L'Extension XML crée les tables annexes avec la colonne INVOICE_NUM et génère une vue par défaut.

Important : Ne modifiez en aucun cas les tables annexes. Vous ne devez mettre à jour le document XML qu'à l'aide des fonctions UDF de l'Extension XML. Dans ce cas, l'Extension XML actualise automatiquement les tables annexes lorsque vous mettez à jour le document XML dans la colonne XML.

Affichage de la colonne et des tables annexes

Une fois la colonne XML activée, vous avez créé une vue de celle-ci et des tables annexes. Vous pouvez utiliser cette vue en mode colonne XML.

Pour afficher la colonne XML et les colonnes des tables annexes, procédez comme suit :

A partir de la fenêtre de commande DB2, entrez l'instruction SQL SELECT suivante :

```
DB2 SELECT * FROM SALES_ORDER_VIEW
```

La vue affiche les colonnes des tables annexes, comme indiqué dans le fichier `getstart_xcolumn.dad`.

Indexation des tables annexes

L'indexation des tables annexes permet d'effectuer des recherches structurelles rapides dans le document XML. A ce stade, vous allez indexer des colonnes de clé figurant dans les tables annexes générées lors de l'activation de la colonne XML ORDER. Le service de maintenance a identifié les colonnes pouvant d'être fréquemment interrogées par le personnel. La tableau 3 décrit ces colonnes que vous allez indexer :

Tableau 3. Colonnes de tables annexes à indexer

Colonne	Table annexe
ORDER_KEY	ORDER_SIDE_TAB
CUSTOMER	ORDER_SIDE_TAB
PRICE	PART_SIDE_TAB
DATE	SHIP_SIDE_TAB

Pour indexer les tables annexes, procédez comme suit :

A partir de la fenêtre de commande DB2, entrez les commandes SQL suivantes :

```
DB2 CREATE INDEX KEY_IDX
      ON ORDER_SIDE_TAB(ORDER_KEY)

DB2 CREATE INDEX CUSTOMER_IDX
      ON ORDER_SIDE_TAB(CUSTOMER)

DB2 CREATE INDEX PRICE_IDX
      ON PART_SIDE_TAB(PRICE)

DB2 CREATE INDEX DATE_IDX
      ON SHIP_SIDE_TAB(DATE)
```

Vous pouvez également créer les index en exécutant le fichier de commandes suivant :

```
getstart_createIndex.cmd
```

Stockage du document XML

Une fois la colonne activée pour pouvoir recevoir le document XML et les tables annexes indexées, vous pouvez stocker le document à l'aide des fonctions de l'Extension XML. Lorsque vous stockez des données dans une colonne XML, vous faites appel aux fonctions de transtypage par défaut ou aux fonctions UDF de l'Extension XML. Etant donné que vous allez stocker un objet de type de base VARCHAR dans une colonne de type UDT XML XMLVARCHAR, vous allez utiliser la fonction de transtypage par défaut. Pour plus d'informations sur les fonctions de transtypage par défaut dédiées au

stockage et sur les fonctions UDF fournies par l'Extension XML, reportez-vous à la section «Stockage des données» à la page 124.

Pour stocker le document XML, procédez comme suit :

Important : Ouvrez le document XML `c:\dxx\samples\xml\getstart.xml`. Assurez-vous que le chemin de fichier contenu dans DOCTYPE correspond à l'ID DTD indiqué dans la DAD et lors de l'insertion de la DTD dans le référentiel des DTD. Pour vérifier leur concordance, interrogez la table db2xml.DTD_REF et vérifiez l'élément DTDID du fichier DAD. Si vous utilisez une unité et un répertoire autres que ceux par défaut, vous devrez peut-être modifier le chemin d'accès dans la déclaration DOCTYPE.

A partir de la fenêtre de commande DB2, entrez la commande SQL INSERT suivante :

```
DB2 INSERT INTO SALES_TAB (INVOICE_NUM, SALES_PERSON, ORDER) VALUES('123456',
   'Sriram Srinivasan', db2xml.XMLVarcharFromFile('c:\dxx\samples\cmd\getstart.xml '))
```

Lorsque vous stockez le document XML, l'Extension XML exécute la mise à jour automatique des tables annexes.

Vous pouvez également stocker le document en exécutant le fichier de commandes suivant :

```
getstart_insertXML.cmd
```

Pour vérifier la mise à jour des tables, lancez les instructions SELECT suivantes à partir de la fenêtre de commande DB2 :

```
DB2 SELECT * FROM SALES_TAB

DB2 SELECT * FROM PART_SIDE_TAB

DB2 SELECT * FROM ORDER_SIDE_TAB

DB2 SELECT * FROM SHIP_SIDE_TAB
```

Recherche dans le document XML

Vous pouvez effectuer des recherches dans le document XML en interrogeant directement les tables annexes. Dans cet exemple, vous allez rechercher toutes les commandes clients dont le prix est supérieur à 2500,00.

Pour interroger les tables annexes, procédez comme suit :

A partir de la fenêtre de commande DB2, entrez l'instruction SQL SELECT suivante :

```
DB2 "SELECT DISTINCT SALES_PERSON FROM SALES_TAB S, PART_SIDE_TAB P
   WHERE PRICE > 2500.00 AND
   S.INVOICE_NUM=P.INVOICE_NUM"
```

L'ensemble de résultats doit présenter les noms des vendeurs qui ont vendu un article dont le prix est supérieur à 2500,00.

Vous pouvez également effectuer des recherches dans le document en exécutant le fichier de commandes suivant :

```
getstart_queryCol.cmd
```

Vous avez terminé l'exécution du tutoriel d'initiation au stockage de documents XML dans des tables DB2. De nombreux exemples du manuel sont basées sur ce tutoriel.

Formation : composition d'un document XML

Scénario du tutoriel

Vous êtes chargé d'examiner une base de commandes existante (SALES_DB) pour y extraire les informations de clé à stocker dans des documents XML. Le service de maintenance aura recours à ces documents XML lors du traitement des demandes et des réclamations clients. Il a spécifié les données à inclure et recommandé une structure de document XML.

A partir de données existant dans les tables, vous allez composer le document XML`getstart.xml`.

Vous allez également planifier et créer un fichier DAD qui mappe les colonnes des tables associées vers une structure de document XML fournissant un enregistrement de type bon de commande. Ce document ayant été composé à partir de plusieurs tables, vous allez créer une collection XML en associant ces tables à une structure XML et à une DTD. Cette DTD permet de définir la structure du document XML. Elle permet également de valider dans les applications le document XML composé.

Les données existantes destinées au document XML sont décrites dans les tableaux ci-après. Les noms de colonnes en *italiques* désignent les colonnes que le service de maintenance a demandé d'inclure dans la structure du document XML.

ORDER_TAB

Nom de colonne	Type de données
ORDER_KEY	INTEGER
CUSTOMER	VARCHAR(16)
CUSTOMER_NAME	VARCHAR(16)
CUSTOMER_EMAIL	VARCHAR(16)

PART_TAB

Nom de colonne	Type de données
PART_KEY	INTEGER
COLOR	CHAR(6)
QUANTITY	INTEGER
PRICE	DECIMAL(10,2)
TAX	REAL
ORDER_KEY	INTEGER

SHIP_TAB

Nom de colonne	Type de données
DATE	DATE
MODE	CHAR(6)
COMMENT	VARCHAR(128)
PART_KEY	INTEGER

Planification

Avant de composer un document avec l'Extension XML, vous devez déterminer la structure du document XML et sa correspondance avec la structure des données de la base. Cette section présente la structure de document XML demandée par le service de maintenance, la DTD avec laquelle vous allez la définir et le mode de mappage du document vers les colonnes de stockage de son contenu.

Détermination de la structure du document

La structure de document XML reçoit de plusieurs tables des informations relatives à une commande client déterminée et créé un document XML pour cette commande. Ces tables contiennent chacune des informations associées sur la commande et peuvent faire l'objet d'une jointure basée sur les colonnes de clés. Le service de maintenance demande un document structuré au niveau supérieur par le numéro de commande et au niveau suivant, par les informations sur le client, la pièce et la livraison. Il souhaite que la structure du document soit intuitive et souple et que les éléments décrivent les données plutôt que la structure du document. (Par exemple, le nom du client doit correspondre à un élément appelé «customer» et non à un paragraphe.) A sa demande, la structure hiérarchique de la DTD et du document XML doit ressembler à celle représentée figure 5 à la page 32.

Une fois la structure de document conçue, vous devez créer une DTD pour la décrire. Ce tutoriel met à votre disposition un document XML et une DTD. Le

fichier DTD se trouve à l'«Annexe B. Exemples» à la page 277. Vous pouvez remarquer qu'il correspond à la structure représentée figure 5.

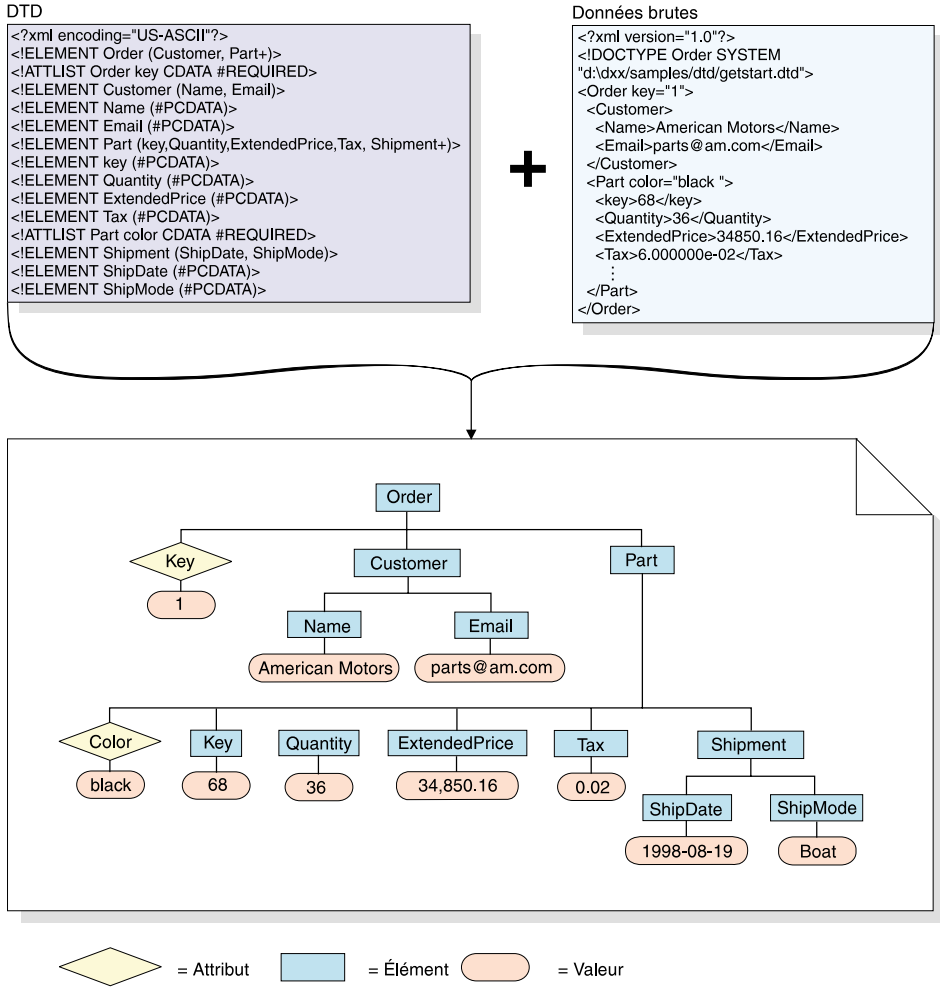


Figure 5. Structure hiérarchique de la DTD et du document XML

Mappage de la relation entre le document XML et la base de données

Une fois la structure conçue et la DTD créée, vous devez présenter la relation existant entre la structure du document et les tables DB2 à l'aide desquelles vous remplirez les éléments et les attributs. Vous pouvez mapper la structure hiérarchique vers certaines colonnes des tables relationnelles (voir figure 6 à la page 33).

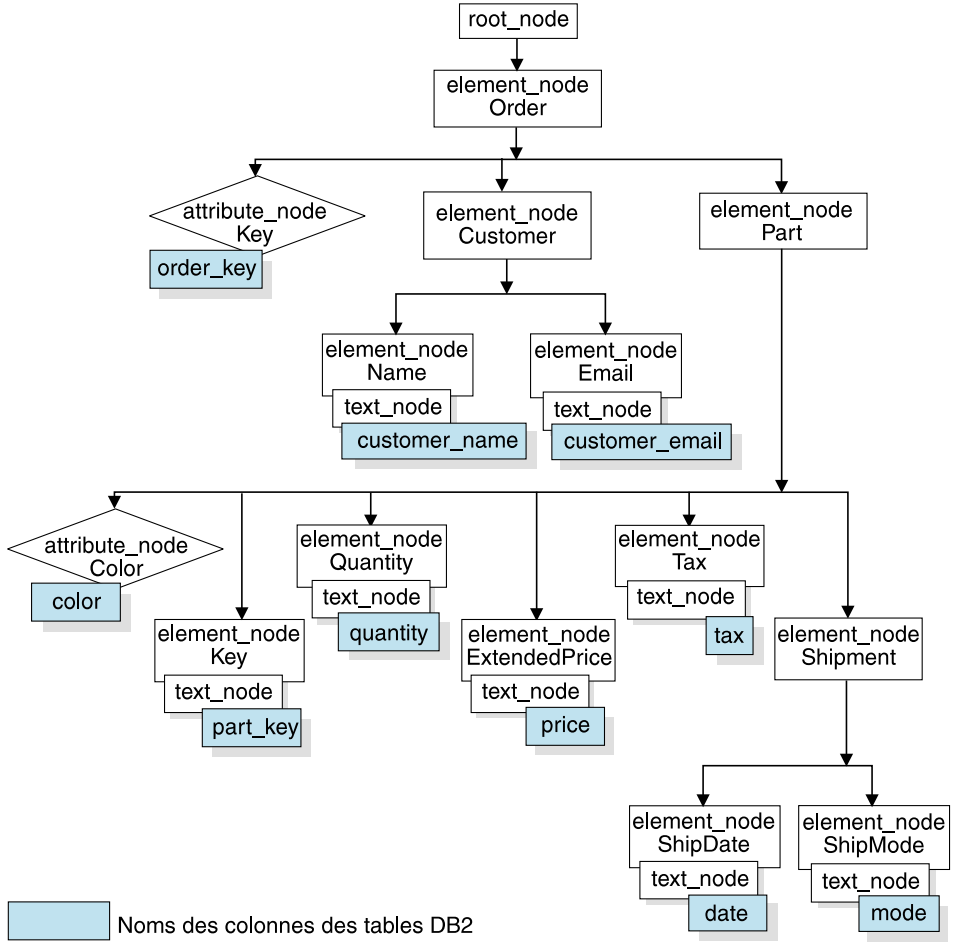


Figure 6. Document XML mappé vers des colonnes de table relationnelle

Cette description permet de créer des fichiers DAD qui définissent la relation existant entre les données relationnelles et la structure du document XML.

Pour pouvoir créer la DAD de collection XML, vous devez comprendre comment le document XML correspond à la structure de la base de données (voir figure 6). Vous pouvez ensuite identifier les tables et les colonnes source à partir desquelles la structure du document XML dérive des données destinées aux éléments et aux attributs. Ces informations permettront de créer le fichier DAD applicable à la collection XML.

Pour ce tutoriel, nous fournissons un jeu de scripts pour la configuration de votre environnement. Ces scripts se trouvent dans le répertoire

DXX_INSTAL\samples\cmd (*DXX_INSTALL* désignant l'unité et le répertoire d'installation de l'Extension XML, par exemple c:\dxx\samples\cmd). Ils se présentent comme suit :

getstart_db.cmd
Crée la base de données et remplit quatre tables.

getstart_prep.cmd
Définit les accès de la base de données aux procédures mémorisées de l'Extension XML et à l'interface CLI de DB2.

getstart_stp.cmd
Exécute la procédure mémorisée pour composer la collection XML.

getstart_exportXML.cmd
Exporte le document XML à partir de la base de données pour l'utiliser dans une application.

getstart_clean.cmd
Nettoie l'environnement du tutoriel.

Configuration

Création de la base de données

Dans cette section, vous allez configurer la base de données par une commande. Cette commande crée un exemple de base de données, vous y connecte, créé les tables qui contiendront les données, puis insère celles-ci.

Important : Si vous avez terminé le cours sur le mode colonne XML et que vous n'avez pas nettoyé votre environnement, vous pouvez sauter cette étape. Vérifiez que vous disposez de la base de données SALES_DB.

Pour créer la base de données, procédez comme suit :

1. Placez-vous dans le répertoire *DXX_INSTALL*\samples\cmd, *DXX_INSTALL* désignant l'unité et le répertoire d'installation de l'Extension XML. Ce cours utilise le répertoire c:\dxx. Modifiez ces valeurs si vous avez recours à une unité et à un répertoire différents.
2. Ouvrez la fenêtre de commande DB2 à partir du menu Démarrage de Windows NT, ou entrez la commande suivante à partir de l'invite de commande Windows NT :

   ```
   DB2CMD
   ```
3. A partir de la fenêtre de commande DB2, lancez la commande suivante :

   ```
   getstart_db.cmd
   ```

Activation de la base de données

Pour pouvoir stocker des informations XML dans la base de données, vous devez d'abord activer celle-ci pour l'Extension XML. Lorsque vous activez une base de données pour XML, DB2 Extension XML exécute les opérations suivantes :

- Création de tous les types (UDT) et fonctions (UDF) définis par l'utilisateur.
- Création et peuplement des tables de contrôle avec les métadonnées requises par l'Extension XML.
- Création du schéma db2xml et affectation des privilèges nécessaires.

Important : Si vous avez terminé le cours sur le mode colonne XML et que vous n'avez pas nettoyé votre environnement, vous pouvez sauter cette étape.

Pour activer la base de données pour XML, procédez comme suit :

A partir de la fenêtre de commande DB2, lancez le script suivant pour activer la base de données SALES_DB :

```
getstart_prep.cmd
```

Ce script définit les accès de la base de données aux procédures mémorisées de l'Extension XML et à l'interface CLI de DB2. Il exécute également l'option de la commande **dxxadm** qui permet d'activer la base de données.

```
dxxadm enable_db VENTES_BD
```

Création de la collection XML : préparation du fichier DAD

Les données existant dans plusieurs tables, vous allez créer une collection XML pour associer ces tables au document XML. Avant de créer une collection XML, vous la définissez par la préparation d'un fichier DAD.

A la section «Planification» à la page 31, vous avez identifié les colonnes de la base relationnelle contenant les données existantes. Vous avez également déterminé le mode selon lequel les données des tables doivent être structurées dans un document XML. Dans cette section, vous allez créer le schéma de mappage dans le fichier DAD pour indiquer la relation existant entre les tables et la structure du document XML.

Dans la procédure suivante, les éléments de la DAD sont appelés *balises* et ceux de la structure du document XML, *éléments*. Vous trouverez un fichier DAD exemple, identique à celui que vous allez créer, dans `c:\dxx\samples\dad\getstart_xcollection.dad`. Cet exemple comporte quelques légères différences par rapport au fichier qui sera généré au cours des étapes suivantes. Si vous l'utilisez pour le cours, notez que les chemins d'accès aux fichiers peuvent différer de ceux de votre environnement.

Pour créer la DAD de composition d'un document XML, procédez comme suit :

1. A partir du répertoire c:\dxx\samples\cmd, ouvrez un éditeur de texte et créez un fichier getstart_xcollection.dad.
2. Créez l'en-tête du fichier DAD en respectant la syntaxe suivante :

   ```
   <?xml version="1.0"?>
   <!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
   ```

   L'Extension XML considère que vous avez installé le produit dans c:\dxx. Si cette valeur est incorrecte, remplacez-la, pour la présente étape et les suivantes, par l'unité et le répertoire appropriés.
3. Insérez les balises <DAD></DAD>. Toutes les autres balises sont imbriquées dans ces balises DAD.
4. A l'aide des balises <validation> </validation>, indiquez que l'Extension XML doit valider la structure du document XML conformément à la DTD placée dans le référentiel des DTD.

   ```
   <validation>NO</validation>
   ```
5. A l'aide des balises <Xcollection></Xcollection>, définissez la méthode d'accès et de stockage collection XML. Cette méthode d'accès et de stockage indique que les données XML sont stockées dans une collection de tables DB2.

   ```
   <Xcollection>
   </Xcollection>
   ```
6. A l'aide d'une instruction SQL, indiquez les tables et les colonnes à inclure dans la collection XML. Cette méthode, appelée mappage SQL, constitue l'un des deux modes possibles pour mapper des données relationnelles vers une structure de document XML. (Pour plus d'informations sur les schémas de mappage, reportez-vous à la section «Types de schémas de mappage» à la page 63.) Entrez l'instruction suivante :

   ```
   <SQL_stmt>
   SELECT o.order_key, customer_name, customer_email, p.part_key, color, quantity,
      price, tax, ship_id, date, mode from order_tab o, part_tab p,
   table (select substr(char(timestamp(generate_unique())),16)
     as ship_id, date, mode, part_key from ship_tab) s
         WHERE o.order_key = 1 and
                   p.price > 20000 and
                p.order_key = o.order_key and
                s.part_key = p.part_key
         ORDER BY order_key, part_key, ship_id
   </SQL_stmt>
   ```

   Cette instruction SQL respecte les directives ci-après. Pour consulter la structure du document, reportez-vous à la figure 6 à la page 33.

   - Les colonnes sont indiquées dans l'ordre descendant, en fonction de la hiérarchie de la structure du document XML. Par exemple, les colonnes relatives aux éléments Order (commande) et Customer (client) sont

placées en première position, l'élément Part (pièce), en deuxième position et enfin l'élément Shipment (livraison), en troisième position.

- Les colonnes d'une même entité sont regroupées et chaque groupe comporte une colonne ID objet : ORDER_KEY (clé de la commande), PART_KEY (clé de la pièce) et SHIP_ID (identificateur de livraison).
- La colonne ID objet est la première colonne de chaque groupe. Par exemple, O.ORDER_KEY précède les colonnes relatives à l'attribut de clé et p.PART_KEY précède les colonnes destinées à l'élément Part.
- La table SHIP_TAB ne comporte pas de colonne conditionnelle à clé unique et par conséquent, vous générez cette colonne à l'aide de la fonction intégrée DB2 generate_unique.
- Les colonnes ID objet sont indiquées dans l'ordre descendant dans l'instruction ORDER BY. Les noms de colonnes de l'instruction ORDER BY ne doivent être qualifiés par aucun nom de schéma et de table et doivent être identiques au noms de colonnes indiqués dans la clause SELECT.

Pour prendre connaissance des exigences liées à l'écriture d'une instruction SQL, reportez-vous à la section «Exigences liées aux schémas de mappage» à la page 65.

7. Ajoutez les informations de prologue suivantes à utiliser dans le document XML en cours de composition.

```
<prolog>?xml
version="1.0"?</prolog>
```

Ce texte est obligatoire pour tous les fichiers DAD.

8. Insérez les balises <doctype></doctype> à utiliser dans le document XML en cours de composition. La balise <doctype> contient le chemin d'accès à la DTD stockée sur le client.

```
<doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
```

9. Définissez l'élément racine du document XML à l'aide des balises <root_node></root_node>. Dans root_node, indiquez les éléments et les attributs qui composent le document XML.
10. Mappez la structure du document XML vers la structure des tables relationnelles DB2 à l'aide des trois types de noeuds suivants :

noeud d'élément (element_node)
Indique l'élément du document XML. Les noeuds d'éléments peuvent comporter des noeuds d'éléments enfants.

noeud d'attribut (attribute_node)
Indique l'attribut d'un élément du document XML.

noeud de texte (text_node)
Indique le contenu textuel d'un élément et les données de colonne d'une table relationnelle pour les noeuds d'éléments de niveau inférieur.

Pour plus d'informations sur ces noeuds, reportez-vous à la section «Fichier DAD» à la page 60. La figure 6 à la page 33 présente la structure hiérarchique du document XML et les colonnes de tables DB2. Elle indique également le type de noeuds utilisés. Les pavés en grisé indiquent les colonnes des tables DB2 à partir desquelles les données seront extraites pour composer le document XML.

La procédure ci-après permet d'ajouter chaque type de noeud, l'un après l'autre.

a. Définissez une balise <element_node> pour chaque élément du document XML.

```
<root_node>
         <element_node name="Order">
           <element_node name="Customer">
    <element_node name="Name">
     </element_node>
    <element_node name="Email">
     </element_node>
              </element_node>
           <element_node name="Part">
    <element_node name="key">
     </element_node>
<element_node name="Quantity">
              </element_node>
           <element_node name="ExtendedPrice">
              </element_node>
           <element_node name="Tax">
              </element_node>
     <element_node name="Shipment" multi_occurrence="YES">
              <element_node name="ShipDate">
              </element_node>
              <element_node name="ShipMode">
              </element_node>
    </element_node> <!-- end Shipment -->
  </element_node> <!-- end Part -->
</element_node> <!-- end Order -->
      </root_node>
```

Notez que chaque élément enfant <Shipment> est associé à l'attribut multi_occurrence défini sur la valeur YES. Cet attribut est utilisé pour les éléments sans attribut répétés plusieurs fois dans un même document. L'élément <Part> n'utilise pas l'attribut multi_occurrence car il est doté d'un attribut color, ce qui le rend unique.

b. Définissez une balise <attribute_node> pour chaque attribut du document XML. Ces attributs sont imbriqués dans leur noeud d'élément. Les noeuds d'attributs ajoutés sont mis en évidence en gras :

```
<root_node>
         <element_node name="Order">
  <attribute_node name="key">
  </attribute_node>
           <element_node name="Customer">
    <element_node name="Name">
     </element_node>
     <element_node names"Email">
     </element_node>
              </element_node>
            <element_node name="Part">
    <attribute_node name="color">
    </attribute_node>
    <element_node name="key">
     </element_node>
<element_node name="Quantity">
              </element_node>

...

  </element_node> <!-- end Part -->
</element_node> <!-- end Order -->
      </root_node>
```

c. Pour chaque noeud d'élément de niveau inférieur, définissez des balises <text_node> pour indiquer que les éléments XML contiennent des données de type caractères à extraire de DB2 lors de la composition du document.

```
<root_node>
         <element_node name="Order">
<attribute_node name="key">
  </attribute_node>
           <element_node name="Customer">
    <element_node name="Name">
              <text_node>
      </text_node>
     </element_node>
    <element_node name="Email">
              <text_node>
      </text_node>
     </element_node>
              </element_node>
            <element_node name="Part">
    <attribute_node name="color">
              </attribute_node>
    <element_node name="key">
              <text_node>
       </text_node>
     </element_node>
<element_node name="Quantity">
```

```
            <text_node>
      </text_node>
            </element_node>
          <element_node name="ExtendedPrice">
            <text_node>
      </text_node>
            </element_node>
          <element_node name="Tax">
            <text_node>
      </text_node>
            </element_node>
    <element_node name="Shipment" multi-occurrence="YES">
            <element_node name="ShipDate">
            <text_node>
        </text_node>
            </element_node>
            <element_node name="ShipMode">
            <text_node>
        </text_node>
            </element_node>
    </element_node> <!-- end Shipment -->
  </element_node> <!-- end Part -->
</element_node> <!-- end Order -->
      </root_node>
```

d. Pour chaque noeud d'élément de niveau inférieur, définissez une balise <column>. Ces balises identifient la colonne source pour l'extraction de données lors de la composition du document XML et sont généralement imbriquées dans les balises <attribute_node> ou <text_node>. Nous vous rappelons que les colonnes définies ici doivent figurer dans la clause SELECT <SQL_stmt>.

```
<root_node>
        <element_node name="Order">
<attribute_node name="key">
    <column name="order_key"/>
  </attribute_node>
          <element_node name="Customer">
    <element_node name="Name">
              <text_node>
        <column name="customer_name"/>
      </text_node>
            </element_node>
    <element_node name="Email">
              <text_node>
        <column name="customer_email"/>
              </text_node>
            </element_node>
            </element_node>
          <element_node name="Part">
    <attribute_node name="color">
       <column name="color"/>
            </attribute_node>
    <element_node name="key">
              <text_node>
```

```
            <column name="part_key"/>
          </text_node>
    <element_node name="Quantity">
                    <text_node>
            <column name="quantity"/>
          </text_node>
                  </element_node>
              <element_node name="ExtendedPrice">
                    <text_node>
            <column name="price"/>
          </text_node>
                  </element_node>
              <element_node name="Tax">
                    <text_node>
            <column name="tax"/>
          </text_node>
                  </element_node>
        <element_node name="Shipment" multi-occurrence="YES">
                  <element_node name="ShipDate">
                     <text_node>
              <column name="date"/>
            </text_node>
                  </element_node>
                  <element_node name="ShipMode">
                     <text_node>
              <column name="mode"/>
            </text_node>
                  </element_node>
        </element_node> <!-- end Shipment -->
      </element_node> <!-- end Part -->
    </element_node> <!-- end Order -->
          </root_node>
```

11. Vérifiez que vous avez placé une balise de fin </root_node> après la dernière balise </element_node>.
12. Vérifiez que vous avez placé une balise de fin </Xcollection> après la dernière balise </root_node>.
13. Vérifiez que vous avez placé une balise de fin </DAD> après la balise </Xcollection>.
14. Sauvegardez le fichier sous le nom `getstart_xcollection.dad`.

Vous pouvez comparer le fichier que vous venez de créer au fichier exemple `c:\dxx\samples\dad\getstart_xcollection.dad`. Il s'agit d'une copie de travail du fichier DAD nécessaire à la composition du document XML. Le fichier exemple contient peut-être des instructions relatives aux chemins d'accès (path) qui doivent être adaptées à votre environnement pour pouvoir s'exécuter.

Dans votre application, si vous faites fréquemment appel à une collection XML pour composer des documents, vous pouvez l'activer afin de définir son nom. Lorsque vous activez une collection, vous l'enregistrez dans la table

XML_USAGE. Vous pouvez ainsi améliorer les performances d'exécution en indiquant le nom de collection (plutôt que le nom de fichier DAD) lors de l'appel de procédures mémorisées. Dans ce cours, vous n'allez pas activer la collection. Pour plus d'informations sur l'activation des collections, reportez-vous à la section «Activation de collections XML» à la page 115.

Composition du document XML

La présente tâche consiste à composer le document XML indiqué dans le fichier DAD à l'aide de la procédure mémorisée dxxGenXML(). Cette procédure mémorisée renvoie le document sous le type UDT XMLVARCHAR.

Pour composer le document XML, procédez comme suit :

1. A partir de la fenêtre de commande DB2, entrez la commande suivante pour exécuter la procédure mémorisée :

   ```
   getstart_stp.cmd
   ```

   Le document XML a été composé et est stocké dans la table RESULT_TAB.

   Vous trouverez dans les fichiers suivants des exemples de procédures mémorisées applicables à la présente tâche :

   - `c:\dxx\samples\c\tests2x.sqc` indique comment appeler la procédure mémorisée à l'aide d'instructions SQL imbriquées et génère le fichier exécutable `texts2x` utilisé par `getstart_stp.cmd`.
   - `c:\dxx\samples\cli\sql2xml.c` indique comment appeler la procédure mémorisée via l'interface CLI (Call Level Interface).
2. Exportez le document XML de la table vers un fichier à l'aide de la fonction Content () (fonction de récupération de l'Extension XML) :

   ```
   DB2 CONNECT TO SALES_DB

   DB2 SELECT db2xml.Content(db2xml.xmlvarchar(doc),
      'c:\dxx\samples\cmd\getstart.xml') FROM RESULT_TAB
   ```

   Vous pouvez également exporter le fichier en exécutant la commande suivante :

   ```
   getstart_exportXML.cmd
   ```

   Ce cours indique comment obtenir un ou plusieurs documents XML à l'aide de la fonction ensemble de résultats de la procédure mémorisée de DB2, permettant d'extraire de chaque document. Vous pouvez exporter chaque ligne du document vers un fichier, ce qui constitue la méthode la plus simple. Pour consulter des méthodes d'extraction plus efficaces, reportez-vous aux exemples CLI contenus dans le répertoire `c:\dxx\samples\cli`.

Nettoyage de l'environnement du tutoriel

Pour nettoyer l'environnement du tutoriel, vous pouvez exécuter le fichier `getstart_clean.cmd`. Ce fichier :

- désactive la colonne XML ORDER,
- retire les tables créées dans le tutoriel,
- supprime la DTD de la table de référence des DTD.

Ce fichier de commande ne désactive ni ne retire la base de données SALES_DB qui reste disponible pour une utilisation avec l'Extension XML. Il est possible que vous receviez des messages d'erreur si vous n'avez pas terminé les deux cours de ce chapitre. Vous pouvez les ignorer.

Pour nettoyer l'environnement du tutoriel, procédez comme suit :

1. A partir de la fenêtre de commande DB2, lancez la commande suivante :
   ```
   getstart_clean.cmd
   ```
2. Pour désactiver la base de données, vous pouvez exécuter la commande Extension XML suivante à partir de la fenêtre de commande DB2 :
   ```
   dxxadm disable_db SALES_DB
   ```

   Cette commande retire les tables de contrôle d'administration DTD_REF et XML_USAGE. Elle retire également les types UDT et les fonctions UDF fournis par l'Extension XML.
3. Pour retirer la base de données, vous pouvez exécuter la commande Extension XML suivante à partir de la fenêtre de commande DB2 :
   ```
   db2 drop database SALES_DB
   ```

   Cette commande retire SALES_DB.

Partie 2. Administration

Cette partie indique comment effectuer les tâches d'administration DB2 Extension XML.

Chapitre 3. Préparation de l'utilisation de l'Extension XML : administration

Ce chapitre décrit les exigences liées à la configuration et à la planification de l'Extension XML.

Exigences liées à la configuration

Les sections suivantes décrivent les exigences liées à la configuration de l'Extension XML.

Logiciels requis

L'Extension XML est disponible pour AIX, Windows NT et Sun Solaris.

Logiciels requis : DB2 Universal Database version 7.1 ou suivante.

Logiciels facultatifs :

- Pour les recherches structurelles : DB2 UDB Extension Texte version 7.1 ou suivante.
- Pour l'outil d'administration de l'Extension XML :
  - Pilote DB2 UDB JDBC (disponible avec DB2 UDB version 7.1 ou suivante).
  - JDK 1.1.7 ou JRE 1.1.7 (disponible avec le Centre de contrôle DB2 UDB).
  - JFC 1.1 avec Swing 1.1 (disponible avec le Centre de contrôle DB2 UDB).

Exigences liées à l'installation

Pour effectuer les tâches suivantes, reportez-vous au fichier README applicable à votre système d'exploitation :

- Définition des accès de l'Extension XML à votre base de données DB2 UDB.

  Pour des raisons de sécurité, vous devez définir les accès de l'Extension XML à chaque base de données. Pour plus d'informations sur la procédure de définition des accès, reportez-vous à la section «Avant de commencer» à la page 217. Vous pouvez également consulter l'exemple

  DXX_INSTALL\samples\cmd\getstart_prep.cmd
- Affichage des instructions de configuration sous UNIX.
- Création d'une base de données pour les accès XML.

Droits d'accès requis

Vous devez disposer des droits d'accès DB2ADM pour pouvoir exécuter les tâches d'administration.

Outils d'administration

L'Extension XML comporte trois outils d'administration : un assistant d'administration, une commande d'administration et des *procédures mémorisées*.

- L'assistant d'administration, que nous vous recommandons, vous aide par l'affichage d'invites au cours des tâches d'administration. Son utilisation est décrite au «Chapitre 4. Administration des données XML» à la page 71.
- La commande d'administration, **dxxadm**, comporte des options pour les différentes tâches d'administration. Son utilisation est décrite au «Chapitre 4. Administration des données XML» à la page 71 et au «Chapitre 7. Commande d'administration de l'Extension XML : **dxxadm**» à la page 161.
- Les procédures mémorisées d'administration comprennent également des options pour les diverses tâches d'administration. Elles sont décrites dans la section «Procédures mémorisées d'administration» à la page 217.

Planification des tâches d'administration

Lorsque vous planifiez une application qui utilise des documents XML, vous devez d'abord prendre des décisions quant aux points suivants :

- Composition de documents XML à partir de données de la base de données.
- Stockage de documents XML pré-existants et option de stockage (documents XML intégralement contenus dans une colonne ou décomposés en données DB2 classiques).

Une fois ces décisions prises, vous pouvez planifier les autres tâches d'administration :

- Validation des documents XML.
- Indexation des données de colonne XML pour optimiser les opérations de recherche et d'extraction.
- Schéma de mappage de la structure du document XML vers des tables relationnelles DB2.

Le mode d'utilisation de l'Extension XML varie en fonction des besoins de votre application. Comme indiqué au «Chapitre 1. Présentation de l'Extension XML» à la page 3, vous pouvez composer des documents XML à partir de données DB2 existantes. Vous pouvez également stocker des documents XML dans DB2, sous forme de documents entiers ou de données DB2. Chaque méthode d'accès et de stockage a ses exigences en matière de planification. Ces points relatifs à la planification sont traités dans les sections suivantes.

Choix d'une méthode d'accès et de stockage

L'Extension XML comporte deux méthodes d'accès et de stockage permettant d'utiliser DB2 comme référentiel XML : la colonne XML et la collection XML.

Vous devez d'abord choisir la méthode qui correspond le mieux aux besoins de votre application pour l'accès aux données XML et leur manipulation.

Colonne XML
Stocke et extrait des documents XML entiers en tant que données de colonne DB2. Les données XML sont représentées par une colonne XML.

Collection XML
Décompose des documents XML en une collection de tables relationnelles, ou compose des documents XML à partir de celle-ci.

Le type de votre application détermine la méthode d'accès et de stockage à utiliser, ainsi que le mode de structuration des données XML. Les scénarios ci-après décrivent les situations où chaque méthode d'accès et de stockage est la mieux adaptée.

Utilisation des colonnes XML

Utilisez les colonnes XML dans les cas suivants :

- Vous disposez de documents XML existants ou provenant d'une source externe et vous préférez les stocker au format natif XML. Vous voulez les stocker dans DB2 pour le maintien de leur intégrité et à des fins d'archivage et de contrôle.
- Les documents XML sont généralement lus, mais non mis à jour.
- Vous voulez stocker, dans le système de fichiers local ou éloigné, des documents XML externes à DB2 avec le type de données nom de fichier. Vous souhaitez également effectuer des opérations de gestion et de recherche sous DB2.
- Vous devez classer les recherches en fonction de valeurs d'éléments ou d'attributs XML et identifier les éléments ou les attributs qui seront souvent utilisés comme arguments de recherche.
- Les documents sont dotés d'éléments avec de grands blocs de texte et vous voulez lancer des recherches structurelles à l'aide de DB2 Extension Texte en conservant l'intégrité des documents.

Utilisation des collections XML

Utilisez les collections XML dans les cas suivants :

- Vous disposez de tables relationnelles contenant des données et vous voulez composer des documents XML conformément à une certaine DTD.
- Vous possédez des documents XML à stocker avec des collections de données qui mappent vers des tables relationnelles.
- Vous voulez créer des vues diverses des données relationnelles avec des schémas de mappage différents.
- Des documents XML proviennent d'autres sources de données. Ce sont les données qui vous intéressent, et non les balises. Vous voulez stocker des

données pures dans la base de données. Vous voulez stocker les données dans des tables existantes ou nouvelles.

- Un petit sous-ensemble de documents XML doit être fréquemment actualisé et les performances de mise à jour sont essentielles.
- Vous avez besoin de stocker des documents XML entrants entiers, mais de n'en extraire qu'un sous-ensemble dans la plupart des cas.
- Vos documents XML dépassent 2 gigaoctets et vous devez les décomposer.

Le fichier DAD (définition d'accès au document) permet d'associer des données XML à des tables DB2 par ces deux méthodes d'accès et de stockage. La figure 7 illustre comment la définition DAD précise les méthodes d'accès et de stockage.

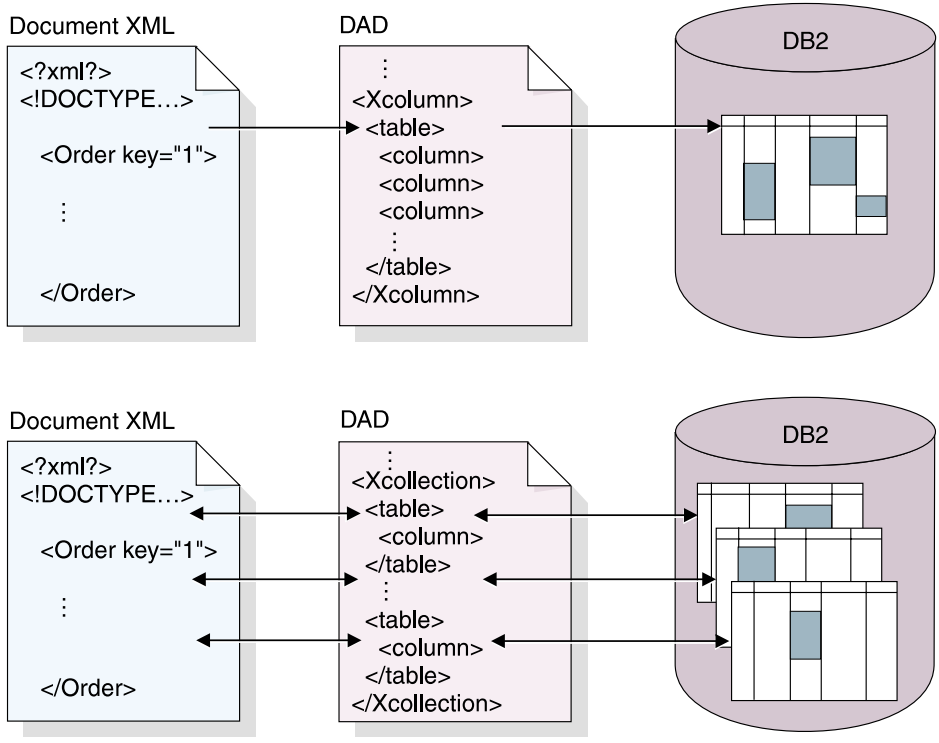


Figure 7. Le fichier DAD mappe la structure du document XML vers DB2 et indique la méthode d'accès et de stockage choisie.

Le fichier DAD constitue une partie importante de l'administration de l'Extension XML. Il précise l'emplacement des fichiers clés tels que la DTD (définition du type de document) et indique la relation existant entre la

structure des documents XML et les données DB2. Plus important, il définit les méthodes d'accès et de stockage utilisées dans votre application.

Planification des colonnes XML

Les sections ci-après décrivent les tâches de planification liées aux colonnes XML.

Validation

Une fois que vous avez choisi une méthode d'accès et de stockage, vous pouvez *valider* vos données. Pour vérifier la validité d'un document XML et permettre des recherches structurées sur les données XML, vous devez utiliser une DTD. La DTD est stockée dans le référentiel des DTD ou dans un système de fichiers auquel le serveur DB2 a accès.

Vous pouvez valider des documents dans une même colonne XML à l'aide de DTD différentes. En d'autres termes, des documents peuvent être dotés d'une structure, d'éléments et d'attributs similaires, mais faire appel à des DTD distinctes. Pour utiliser plusieurs DTD, procédez comme suit :

- Dans la définition DOCTYPE, l'ID système du document XML doit désigner le fichier DTD par son chemin complet.
- Dans le fichier DAD, vous devez indiquer la valeur YES pour l'option de validation.
- Au moins une DTD doit figurer dans la table DTD_REF. Toutes les DTD peuvent être stockées dans cette table.
- Les DTD doivent présenter une structure commune, comportant des différences uniquement au niveau des sous-éléments.
- Le fichier DAD doit contenir des éléments ou des attributs communs à toutes les DTD utilisées par les documents de la colonne.

Important : Prenez votre décision en matière de validation avant d'insérer des données XML dans DB2. L'Extension XML n'accepte pas la validation de données déjà insérées dans DB2.

Remarques :

- Il est recommandé de valider les données XML à l'aide d'une DTD, sauf si vous stockez les documents XML à des fins d'archivage. Pour cette validation, vous avez besoin d'une DTD dans le référentiel de l'Extension XML. Pour apprendre à insérer une DTD dans le référentiel, reportez-vous à la section «Insertion d'une DTD dans le référentiel des DTD» à la page 76.
- Vous n'avez pas besoin de DTD pour stocker ou archiver des documents XML.
- La validation des données XML peut avoir un léger impact sur les performances.

- Vous pouvez avoir recours à plusieurs DTD, mais vous ne pouvez indexer que des éléments et des attributs communs.
- Si vous choisissez de ne pas valider un document, la DTD spécifiée par le document XML n'est pas traitée. Il est important que les DTD soient traitées en vue de la résolution des valeurs par défaut des entités et attributs, même lorsque le traitement de fragments de documents ne peut être validé.

Types XML définis par l'utilisateur (UDT)

Vous stockez dans une colonne XML un document XML en lui attribuant un type UDT (défini par l'utilisateur). Pour obtenir la liste des UDT disponibles, reportez-vous au tableau 4.

Tableau 4. Types de données UDT Extension XML

Colonne UDT	Type de la source de données	Description
XMLVARCHAR	VARCHAR(*varchar_len*)	Stocke un document XML entier en tant que VARCHAR dans DB2.
XMLCLOB	CLOB(*clob_len*)	Stocke un document XML entier en tant qu'objet *CLOB* dans DB2.
XMLFILE	VARCHAR(1024)	Stocke le nom de fichier d'un document XML dans DB2 et le document XML dans un fichier local pour le serveur DB2.

Tables annexes

Lors de la planification des tables annexes, vous devez déterminer leur mode d'organisation, le nombre de tables à créer et la génération ou non d'une vue par défaut pour celles-ci. Ces décisions dépendent en partie de plusieurs facteurs : la répétition ou non d'éléments ou d'attributs et les exigences relatives aux performances de recherche.

Occurrences multiples: Lorsqu'un document comporte des chemins d'emplacement à occurrences multiples, l'Extension XML ajoute une colonne DXX_SEQNO de type INTEGER dans chaque table pour y conserver l'ordre des éléments répétés. DXX_SEQNO permet d'extraire une liste d'éléments utilisant le même ordre que le document XML d'origine, par l'insertion de la clause ORDER BY DXX_SEQNO dans une requête SQL.

Vues par défaut et performances des requêtes: Lorsque vous activez une colonne XML, vous pouvez indiquer une vue par défaut accessible en lecture uniquement, réalisant la jointure de la table d'application et des tables annexes sous un ID unique, ROOT ID. Avec la vue par défaut, vous pouvez

recherchez des documents XML par l'interrogation de tables annexes. Par exemple, soient la table d'application SALES_TAB et les tables annexes ORDER_TAB, PART_TAB et SHIP_TAB :

```
SELECT sales_person FROM sales_order_view
   WHERE price > 2500.00
```

L'instruction SQL renvoie le nom des vendeurs figurant dans SALES_TAB et comportant dans la colonne ORDER des commandes dont le prix (PRICE) est supérieur à 2500,00.

L'interrogation de la vue par défaut présente l'avantage de fournir une vue virtuelle unique de la table d'application et des tables annexes. Cependant, le coût de la requête est proportionnel au nombre de tables annexes créées. Par conséquent, la création d'une vue par défaut n'est recommandée que lorsque le nombre total de colonnes de tables annexes est peu élevé. Les applications peuvent créer leurs propres vues, par la jointure des colonnes essentielles de tables annexes.

Indexation des données de colonnes XML

L'indexation ou non du document de colonne XML est une décision importante en matière de planification. Vous devez la prendre en fonction de la fréquence d'accès aux données et de l'importance des performances de recherche structurelle.

Lorsque vous utilisez des colonnes XML contenant des documents XML entiers, vous pouvez créer des tables annexes dotées de colonnes comportant des valeurs d'éléments ou d'attributs XML, puis définir des index pour ces colonnes. Vous devez déterminer les éléments et les attributs concernés par l'index.

L'indexation de colonne XML permet d'indexer, via le support d'index DB2 natif provenant du moteur de base de données, des données de type général (INTEGER, DECIMAL ou DATE) fréquemment interrogées. L'Extension XML extrait les valeurs d'éléments ou d'attributs XML des documents XML, puis les stocke dans les *tables annexes*, permettant de créer des index pour ces dernières.

Vous pouvez désigner chaque colonne de table annexe par un chemin d'emplacement qui identifie un élément ou un attribut XML et un type de données SQL. La figure 8 à la page 54 représente une colonne XML associée à des tables annexes.

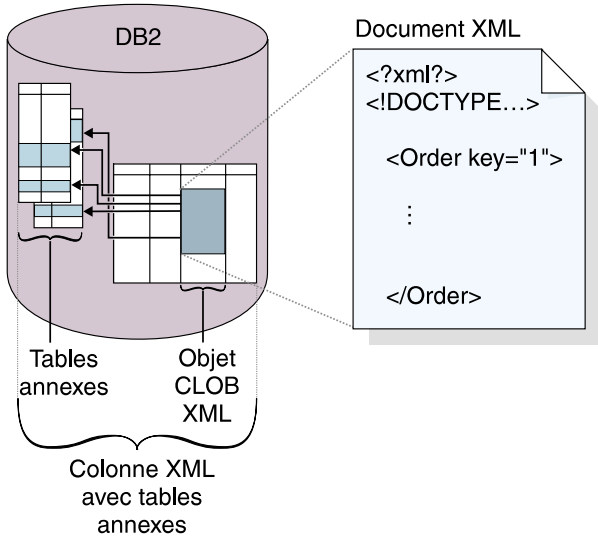


Figure 8. Colonne XML associée à des tables annexes

L'Extension XML remplit automatiquement la table annexe lorsque vous stockez des documents XML dans la colonne XML.

Pour accélérer les recherches, créez des index pour ces colonnes avec la méthode d'*indexation en arbre B* de DB2. Les méthodes d'indexation utilisées varient en fonction des systèmes d'exploitation et sont supportées par l'Extension XML.

Remarques :

- Pour les éléments ou les attributs XML à *occurrences multiples*, vous devez créer une table annexe distincte pour chacun d'eux, en raison de la structure complexe des documents XML.

  Par exemple, vous pouvez créer un index pour `/Order/Part/ExtendedPrice` et lui attribuer le type de données REAL. Dans ce cas, l'Extension XML stocke la valeur `/Order/Part/ExtendedPrice` dans la colonne PRICE d'une table annexe.
- Vous pouvez créer plusieurs index pour une colonne XML. A partir de l'exemple précédent, vous pouvez créer deux colonnes dans deux tables annexes, l'une pour `ExtendedPrice` et l'autre pour `ShipDate`.
- Vous pouvez associer toutes les tables annexes à la table d'application à l'aide de ROOT ID, du nom de colonne de la clé primaire appartenant à la table d'application et d'un identificateur unique. Vous pouvez sélectionner la clé primaire de la table d'application comme ROOT ID (ID racine), même si elle ne peut pas servir de clé composée. Nous vous recommandons cette méthode.

Si la clé primaire unique n'existe pas dans la table d'application, ou que vous ne voulez pas l'utiliser pour une raison quelconque, l'Extension XML modifie la table d'application pour ajouter la colonne DXXROOT_ID qui contient un ID unique créé lors de l'insertion. Toutes les tables annexes comportent une colonne DXXROOT_ID associée à l'ID unique. Si la clé primaire correspond à la valeur ROOT ID, toutes les tables annexes ont une colonne dont le nom et le type sont identiques à ceux de la colonne de clé primaire de la table d'application et les valeurs des clés primaires sont stockées.

- Si vous activez une colonne XML pour DB2 Extension Texte, vous pouvez également utiliser la fonction de recherche structurelle de l'Extension Texte. L'Extension Texte est dotée d'un support de *recherche par section* qui élargit les possibilités de recherche intégrale classique. En effet, les arguments de recherche sont comparés dans le contexte du document désigné par un chemin d'emplacement. La méthode d'*indexation structurelle* peut être utilisée avec la méthode d'indexation de l'Extension XML basée sur des types de données SQL généraux.

Chemin d'emplacement

Un *chemin d'emplacement* est une séquence de *balises XML* identifiant un attribut ou un élément XML. L'Extension XML utilise le chemin d'emplacement dans les cas suivants :

- Lors de l'extraction de fonctions UDF pour identifier les éléments et les attributs à extraire.
- Pour désigner le fichier de mappage entre un attribut ou un élément XML et une colonne DB2, lors de la définition du mode d'indexation dans la DAD pour les colonnes XML.
- Lors de la recherche structurelle par l'Extension Texte.

La figure 9 à la page 56 donne un exemple de chemin d'emplacement et de ses relations par rapport à la structure du document XML.

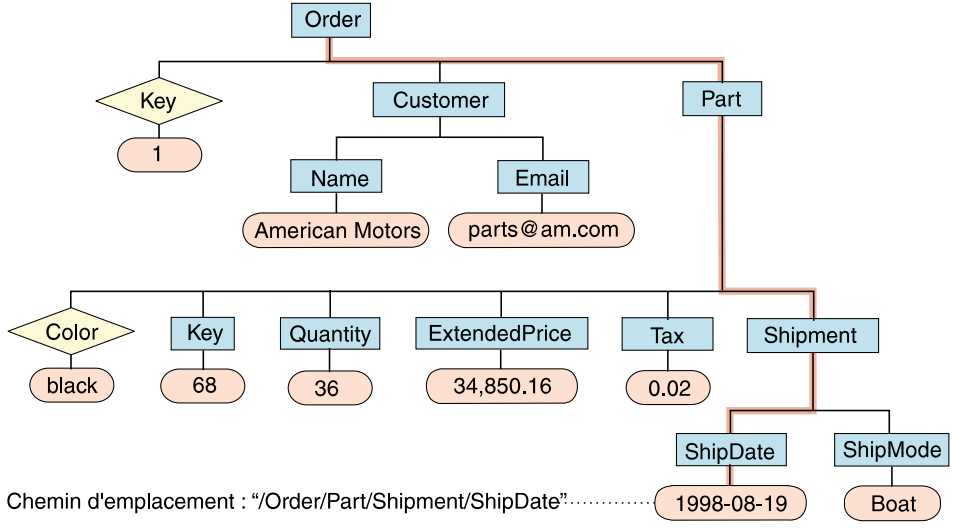


Figure 9. Stockage de documents sous forme de documents XML structurés dans une colonne de table DB2

Syntaxe du chemin d'emplacement: La liste suivante décrit la syntaxe de chemin d'emplacement prise en charge par l'Extension XML : Un chemin avec une barre oblique simple (/) indique que le contexte est constitué du document entier.

1. / Représente l'*élément racine* XML.

2. */tag1*
Représente l'élément *tag1* sous la racine.

3. */tag1/tag2/..../tagn*
Représente *tagn* comme élément enfant de la chaîne décroissante allant de la racine, *tag1*, *tag2*, jusqu'à *tagn*-1.

4. *//tagn*
Représente l'élément *tagn*. La double barre oblique (//) indique zéro ou d'autres balises obligatoires.

5. */tag1//tagn*
Représente *tagn*, élément enfant de *tag1* sous la racine. La double barre oblique (//) indique zéro ou d'autres balises obligatoires.

6. */tag1/tag2/@attr1*
Représente l'attribut *attr1* de l'élément *tag2*, qui est l'enfant de l'élément *tag1* sous la racine.

7. */tag1/tag2[@attr1="5"]*
Représente l'élément *tag2* dont l'attribut *attr1* a la valeur 5. *tag2* est l'élément enfant de *tag1* sous la racine.

8. */tag1/tag2[@attr1="5"]/.../tagn*

Représente *tagn*, élément enfant de la chaîne décroissante allant de la racine, *tag1*, *tag2*, jusqu'à *tagn*-1, où l'attribut *attr1* de *tag2* a la valeur 5.

Caractères génériques : Vous pouvez substituer un astérisque (pour une chaîne quelconque) à un élément du chemin d'emplacement.

Chemin d'emplacement simple: Le *chemin d'emplacement simple* est la syntaxe de chemin d'emplacement utilisée pour spécifier les éléments et attributs des tables annexes, définie dans le fichier DAD de la colonne XML. Le chemin d'emplacement simple est une séquence de noms de types d'éléments connectés par une seule barre oblique (/). Les valeurs d'attributs sont indiquées entre crochets à la suite du type d'élément correspondant. Le tableau 5 décrit les syntaxes relatives au chemin d'emplacement simple.

Tableau 5. Syntaxe de chemin d'emplacement simple

Objet	Chemin d'emplacement	Description
Elément XML	*/tag1/tag2/..../tagn-1/tagn*	Contenu d'élément identifié par l'élément *tagn* et ses parents
Attribut XML	*/tag_1/tag_2/..../tag_n-1/tag_n/@attr1*	Attribut *attr1* de l'élément identifié par *tagn* et ses parents

Restrictions de l'Extension XML: L'Extension XML comporte des restrictions quant à l'utilisation de chemin d'emplacement lors de la définition de l'élément ou de l'attribut dans la DAD. L'Extension XML utilisant un mappage de type biunivoque entre un élément ou un attribut, et une colonne DB2, cela limite les règles de syntaxe autorisées dans le fichier DAD et dans les fonctions. Le tableau 6, décrit les restrictions relatives au chemin d'emplacement. Les numéros figurant dans la colonne "Chemin d'emplacement pris en charge" désignent les syntaxes indiquées à la section «Syntaxe du chemin d'emplacement» à la page 56.

Tableau 6. Restrictions de l' Extension XML relatives au chemin d'emplacement

Utilisation du chemin d'emplacement	Chemin d'emplacement pris en charge
Elément du mappage DAD de colonne XML pour les tables annexes	3, 6 (chemin d'emplacement simple décrit au tableau 5)
Fonctions UDF d'extraction	1-8[1]
Fonction UDF de mise à jour (Update)	1-8[1]
Fonctions UDF de recherche de l'Extension Texte	1-8

Tableau 6. Restrictions de l' Extension XML relatives au chemin d'emplacement (suite)

Utilisation du chemin d'emplacement	Chemin d'emplacement pris en charge

[1] Notez que les fonctions UDF d'extraction et de mise à jour prennent en charge les chemins d'emplacement qui comportent des prédicats avec des attributs, mais sans éléments.

Fichier DAD

Pour les colonnes XML, le fichier DAD indique essentiellement le mode d'indexation des documents qu'elles contiennent. Le fichier DAD est un document au format XML résidant sur le client. Si vous validez des documents XML avec une DTD, vous pouvez relier le fichier DAD à celle-ci. Le fichier DAD est associé au type de données CLOB.Il peut contenir jusqu'à 100 ko de données.

Le fichier DAD de colonne XML comporte un en-tête XML, indique son chemin d'accès et celui de la DTD sur le client. Il fournit également une mappe des données XML à stocker dans des tables annexes pour indexation.

Pour indiquer la méthode d'accès et de stockage de colonne XML, vous insérez la balise ci-après dans le fichier DAD :

<Xcolumn>

Indique que les données XML doivent être stockées et extraites en tant que documents XML entiers dans des colonnes DB2 activées pour XML.

Une colonne activée pour XML est de type UDT. Les applications peuvent inclure la colonne dans toute *table utilisateur*. Les données de la colonne XML sont accessibles par des instructions SQL et par les fonctions UDF de l'Extension XML.

Vous pouvez créer et mettre à jour le fichier DAD à l'aide de l'assistant d'administration de l'Extension XML ou d'un éditeur.

Planification des collections XML

Lors de la planification des collections XML, vous devez prendre en compte certains points pour la composition de documents à partir de données DB2 et pour la décomposition de documents XML en données DB2. Les sections suivantes traitent de la planification de collections XML et contiennent des remarques sur les opérations de composition et de décomposition.

Validation

Une fois la méthode d'accès et de stockage choisie, vous pouvez valider vos données XML à l'aide d'une DTD. La DTD permet de vérifier la validité d'un document XML et de lancer des recherches structurées sur les données XML. Elle est stockée dans le référentiel des DTD.

Recommandation : Validez les données XML à l'aide d'une DTD. Pour cette validation, vous avez besoin d'une DTD dans le référentiel de l'Extension XML. Pour apprendre à insérer une DTD dans le référentiel, reportez-vous à la section «Insertion d'une DTD dans le référentiel des DTD» à la page 76. Les exigences liées à la DTD diffèrent selon que vous composez ou décomposez des documents XML.

- Pour la composition, vous ne pouvez valider que des documents XML générés à l'aide d'une seule DTD. La DTD à utiliser est indiquée dans le fichier DAD.
- Pour la décomposition, vous pouvez valider des documents à l'aide de DTD différentes. En d'autres termes, vous pouvez décomposer des documents à l'aide du même fichier DAD, mais de DTD distinctes. Pour utiliser plusieurs DTD, procédez comme suit :
  - Au moins une DTD doit figurer dans la table DTD_REF. Toutes les DTD peuvent être stockées dans cette table.
  - Les DTD doivent présenter une structure commune, comportant des différences uniquement au niveau des sous-éléments.
  - Dans le fichier DAD, vous devez indiquer l'option de validation.
  - L'ID système du document XML doit désigner le fichier DTD par son chemin complet.
  - Le fichier DAD contient la spécification relative au mode de décomposition du document. Par conséquent, vous ne pouvez indiquer pour la décomposition que des éléments et des attributs communs. Les éléments et les attributs spécifiques à une DTD ne peuvent pas être décomposés.

Important : Prenez votre décision en matière de validation avant d'insérer des données XML dans DB2. L'Extension XML n'accepte pas la validation de données déjà insérées dans DB2.

Remarques :

- Vous devez utiliser une DTD lorsque vous adoptez le format d'échange XML.
- La validation des données XML peut avoir un léger impact sur les performances.
- Vous ne pouvez décomposer que des éléments et des attributs communs lorsque vous utilisez plusieurs DTD.
- Vous pouvez décomposer tous les éléments et attributs lorsque vous utilisez une seule DTD.
- Vous ne pouvez utiliser qu'une seule DTD pour la composition.

Fichier DAD

Pour les collections XML, le fichier DAD mappe la structure du document XML vers les tables DB2 utilisées pour les opérations de composition ou de décomposition.

Par exemple, si un élément <Tax> est présent dans votre document XML, vous devez peut-être le mapper vers la colonne TAX. Vous définissez dans le fichier DAD la relation entre les données XML et les données relationnelles.

Le fichier DAD est indiqué lors de l'activation d'une collection, ou de l'utilisation du fichier DAD avec des *procédures mémorisées* de collection XML. Le fichier DAD est un document au format XML résidant sur le client. Si vous validez des documents XML avec une DTD, vous pouvez relier le fichier DAD à celle-ci. Utilisé comme paramètre d'entrée dans les procédures mémorisées de l'Extension XML, le fichier DAD est associé au type de données CLOB. Il peut contenir jusqu'à 100 ko de données.

Pour indiquer la méthode d'accès et de stockage de collection XML, vous insérez la balise ci-après dans le fichier DAD :

<Xcollection>
: Indique que les données XML doivent être décomposées à partir de documents XML en une collection de tables relationnelles, ou qu'elles doivent être composées en documents XML à partir de celle-ci.

  Une collection XML correspond à un nom virtuel attribué à un ensemble de tables relationnelles contenant des données XML. Les applications peuvent activer une collection XML de tables utilisateur. Ces tables utilisateur peuvent être des tables contenant des données de gestion existantes ou des tables récemment créées par l'Extension XML. Les données de collection XML sont accessibles principalement à l'aide des procédures mémorisées de l'Extension XML.

Le fichier DAD définit l'arborescence du document XML à l'aide des types de noeuds suivants :

root_node (noeud racine)
: Indique l'élément racine du document.

element_node (noeud d'élément)
: Identifie un élément, qui peut être l'élément racine ou un élément enfant.

text_node (noeud de texte)
: Représente le texte CDATA d'un élément.

attribute_node (noeud d'attribut)
: Représente un attribut d'élément.

La figure 10 représente un fragment du mappage utilisé dans un fichier DAD. Les noeuds mappent le contenu du document XML vers les colonnes d'une table relationnelle.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dtd\dad.dtd">
<DAD>
  ...
<Xcollection>
<SQL_stmt>
       ...
</SQL_stmt>
<prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE DAD SYSTEM "c:\dxx\sample\dtd\getstart.dtd"</doctype>
<root_node>
   <element_node name="Order">         --> Identifies the element <Order>
     <attribute_node name="key">       --> Identifies the attribute "key"
        <column name="order_key"/>     --> Defines the name of the column, "order_key",
                                           to which the element and attribute are mappe
     </attribute_node>
     <element_node name="Customer">    --> Identifies a child element of <Order> as
                                           <Customer>
        <text_node>                    --> Specifies the CDATA text for the element
                                           <Customer>
          <column name="customer">     --> Defines the name of the column, "customer",
                                           to which the child element is mapped
        </text_node>
     </element_node>
        ...
   </element_node>

      ...
<root_node>
</Xcollection>
</DAD>
```

Figure 10. Définitions de noeuds

Dans cet exemple, les deux premières colonnes de l'instruction SQL contiennent des éléments et des attributs mappés vers ces dernières.

L'Extension XML prend également en charge les instructions de traitement des feuilles de style, via l'élément <stylesheet>. Cet élément doit se trouver dans le noeud racine du fichier DAD, avec les instructions DOCTYPE et PROLOG définies pour le document XML. Par exemple :

```
<Xcollection>
...
<prolog>...</prolog>
<doctype>...</doctype>
<stylesheet>?xml-stylesheet type="text/css" href="order.css"?</stylesheet>
<root_node>...</root_node>
...

</Xcollection>
```

Vous pouvez créer et mettre à jour le fichier DAD à l'aide de l'assistant d'administration de l'Extension XML ou d'un éditeur.L'élément <stylesheet> n'est pas pris en charge par l'assistant d'administration de l'Extension XML.

Schémas de mappage pour collections XML

Si vous utilisez une collection XML, vous devez sélectionner un *schéma de mappage* qui définit le mode de représentation des données XML dans une base de données relationnelle. Les collections XML devant correspondre à une structure hiérarchique utilisée dans les documents XML à structure relationnelle, vous devez comprendre le mode de comparaison de ces deux structures. La figure 11 à la page 63 indique comment la structure hiérarchique peut être mappée vers des colonnes de table relationnelle.

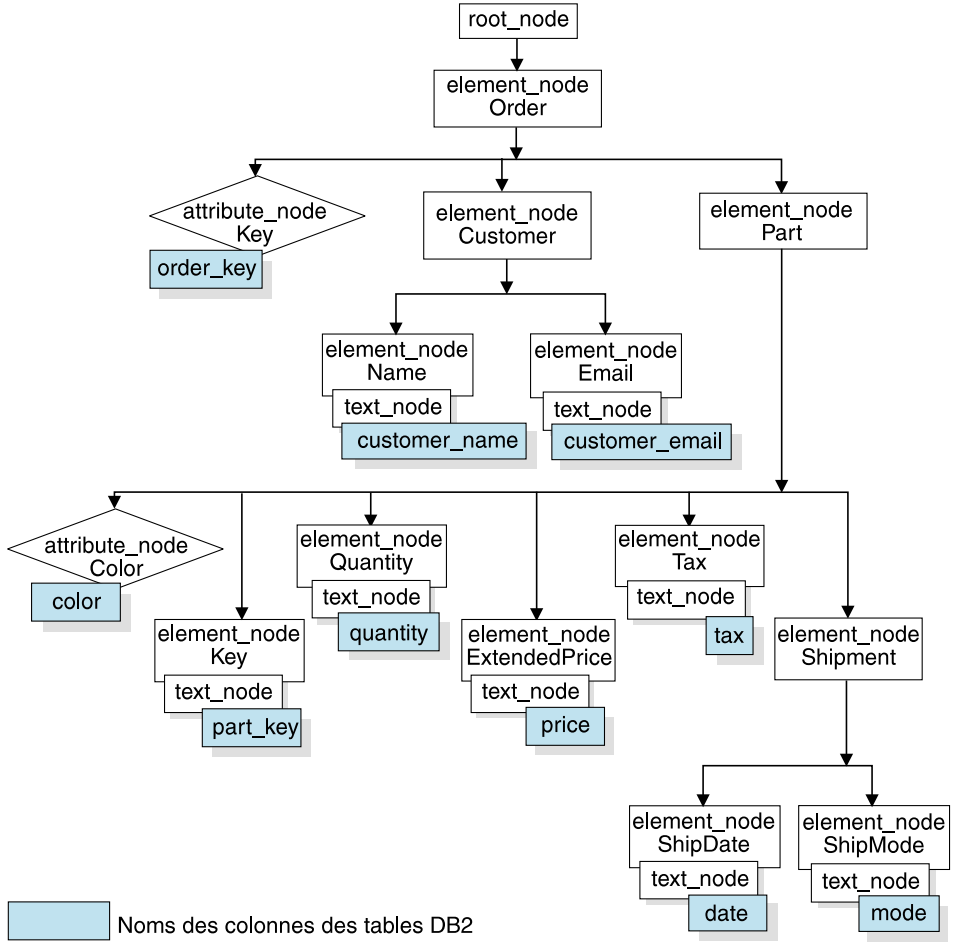


Figure 11. Structure de document XML mappée vers des colonnes de table relationnelle

L'Extension XML utilise le schéma de mappage lorsqu'il compose ou décompose des documents XML stockés dans plusieurs tables relationnelles. Il comporte un assistant qui vous aide à créer le fichier DAD. Cependant, vous devez au préalable choisir le schéma de mappage des données XML vers la collection XML.

Types de schémas de mappage: Le schéma de mappage choisi est précisé dans l'élément <Xcollection> du fichier DAD. L'Extension XML comporte deux types de schémas de mappage : le *mappage SQL* et le *mappage de noeud de base de données relationnelle (RDB_node)*. Les deux méthodes définissent la hiérarchie du document XML à partir du modèle XSLT.

Mappage SQL

Permet le mappage simple et direct à partir de données relationnelles vers des documents XML, à l'aide d'une seule instruction SQL et du *modèle de données XSLT*. Le mappage SQL est utilisé pour la composition et non pour la décomposition. Il est défini avec l'élément SQL_stmt dans le fichier DAD. Cet élément SQL_stmt contient une instruction SQL valide. Il mappe les colonnes de la clause SELECT vers les éléments ou les attributs XML du document XML. Lorsqu'ils sont indiqués pour la composition de documents XML, les noms de colonnes figurant dans la clause SELECT de l'instruction SQL permettent de définir la valeur d'un noeud d'attribut (*attribute_node)* ou le contenu d'un noeud de texte (*text_node*). La clause FROM définit les tables qui contiennent les données. La clause WHERE indique la *condition* de recherche et de *jointure.*

Le mappage SQL permet aux utilisateurs DB2 de mapper les données à l'aide de SQL. Dans ce mode, vous pouvez réaliser la jointure de toutes les tables en une seule instruction SELECT pour constituer une requête. Si une seule instruction SQL ne vous suffit pas, servez-vous du mappage de noeud RDB. Pour lier toutes les tables entre elles, nous vous recommandons de leur appliquer une relation de type *clé primaire - clé associée.*

Mappage de noeud RDB

Définit l'emplacement d'un contenu élémentaire XML ou d'une valeur d'attribut XML, pour que l'Extension XML puisse déterminer l'emplacement de stockage ou de récupération des données XML.

Le noeud de base de données relationnelle (*RDB_node*) contient une ou plusieurs définitions de noeud applicables aux tables, aux colonnes et aux conditions facultatives. Les tables et les colonnes permettent de définir le mode de stockage des données XML dans la base de données. La condition indique soit les critères de sélection des données XML, soit le mode de jointure des tables d'une collection XML.

Pour définir un schéma de mappage, vous créez un élément <Xcollection>. La figure 12 à la page 65, représente un fragment d'exemple de fichier DAD avec mappage SQL de collection XML, permettant de composer un ensemble de documents XML à partir de trois tables relationnelles.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dtd\dad.dtd">
<DAD>
  <dtdid>c:\dxx\samples\dad\getstart.dtd</dtdid>
   <validation>YES</validation>
<Xcollection>
<SQL_stmt>
       SELECT o.order_key, customer, p.part_key, quantity, price, tax, date,
              mode, comment
         FROM order_tab o, part_tab p,
               table(select substr(char(timestamp(generate_unique())),
               as ship_id, date, mode, from ship_tab) as s
        WHERE p.price > 2500.00 and s.date > "1996-06-01" AND
              p.order_key = o.order_key and s.part_key = p.part_key
</SQL_stmt>
      <prolog>?xml version="1.0"?</prolog>
    <doctype>!DOCTYPE DAD SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
      <root_node>
   <element_node name="Order">
  <attribute_node name="key">
<column_name="order_key"/>
  </attribute_node>
  <element_node name="Customer">
  <text_node>
  <column name="customer"/>
  </text_node>
  <element_node>

...

  </element_node><!-end Part->
   </element_node><!-end Order->
      </root_node>
</Xcollection>
</DAD>
```

Figure 12. Schéma de mappage SQL

L'Extension XML comporte plusieurs procédures mémorisées pour la gestion des données dans une collection XML. Ces procédures mémorisées supportent les deux types de mappage, mais exigent que le fichier DAD respecte les règles décrites à la section «Exigences liées aux schémas de mappage».

Exigences liées aux schémas de mappage: Les sections ci-après décrivent les exigences liées à chaque schéma de mappage pour collections XML.

Exigences liées au mappage SQL

Pour le schéma de mappage SQL, vous devez indiquer l'élément SQL_stmt dans l'élément <Xcollection> du fichier DAD. L'élément SQL_stmt doit contenir une seule instruction SQL pouvant réaliser la jointure de plusieurs tables relationnelles à l'aide du *prédicat* de requête. En outre, les clauses suivantes sont obligatoires :

- **Clause SELECT**
  - Permet de s'assurer que le nom de colonne est unique. Si deux tables contiennent le même nom de colonne, créez un nom d'alias pour l'une d'elles à l'aide du mot clé AS.
  - Regroupez les colonnes d'une même table et adoptez le niveau hiérarchique logique des tables relationnelles. Vous rassemblez les tables par niveau d'importance, en fonction de leur mappage vers la structure hiérarchique de votre document XML. Dans la clause SELECT, les colonnes des tables de niveau supérieur doivent précéder celles des tables de niveau inférieur. L'exemple suivant illustre la relation hiérarchique existant entre les tables :

    ```
    SELECT o.order_key, customer, p.part_key, quantity, price, tax,
           ship_id, date, mode
    ```

    Dans cet exemple, les éléments order_key et customer de la table ORDER_TAB ont le niveau hiérarchique le plus élevé car ils se situent à un niveau supérieur dans l'arborescence du document XML. Les éléments ship_id, date et mode de la table SHIP_TAB se trouvent au niveau hiérarchique inférieur.
  - Utilisez une clé candidate composée d'une seule colonne au début de chaque niveau. Si une telle clé n'est pas disponible, la requête doit en générer une pour la table à l'aide d'une expression de table et de la fonction intégrée generate_unique(). Dans l'exemple précédent, o.order_key et part_key sont respectivement les clés primaires des tables ORDER_TAB et PART_TAB. Elles figurent au début de leur groupe de colonnes à sélectionner. Une clé primaire (ship_id) dans ce cas) doit être générée pour la table SHIP_TAB qui n'en comporte pas. C'est la première colonne du groupe de la table SHIP_TAB. Générez la colonne de clé primaire à l'aide de la clause FROM, comme indiqué dans l'exemple ci-après.
- **Clause FROM**
  - A l'aide d'une expression de table et de la fonction intégrée generate_unique(), générez une clé primaire unique pour les tables qui n'en comportent pas. Par exemple :

    ```
    FROM order_tab as o, part_tab as p,
           table(select substr(char(timestamp(generate_unique())),16) as
           ship_id, date, mode from ship_tab) as s
    ```

Dans cet exemple, la fonction generate_unique() est transtypée en type de données CHAR TIMESTAMP et reçoit l'alias `ship_id`.

– Si nécessaire, servez-vous d'un nom d'alias pour distinguer une colonne. Par exemple, vous pouvez utiliser `o` pour ORDER_TAB, `p` pour PART_TAB et `s` pour SHIP_TAB.

- **Clause WHERE**
  – Indiquez une clé primaire et une clé associée comme condition de jointure reliant les tables entre elles dans la collection. Par exemple :

```
WHERE p.price > 2500.00 AND s.date > "1996-06-01" AND
      p.order_key = o.order_key AND s.part_key = p.part_key
```

  – Indiquez toute autre condition de recherche dans le prédicat. Vous pouvez utiliser n'importe quel prédicat valide.
- **Clause ORDER BY**
  – Définissez la clause ORDER BY à la fin de SQL_stmt.
  – Assurez-vous que les noms de colonnes sont identiques à ceux indiqués dans la clause SELECT.
  – Indiquez les noms de colonnes ou les identificateurs désignant de façon unique les entités dans la structure entité-relation de la base de données. Vous pouvez générer un identificateur à l'aide d'une expression de table et de la fonction intégrée generate_unique ou d'une fonction UDF (définie par l'utilisateur).
  – Conservez l'ordre hiérarchique descendant dans lequel les entités sont rangées. La colonne indiquée dans la clause ORDER BY doit être la première colonne affichée pour chaque entité. Le maintien de cet ordre permet de s'assurer que les documents XML à générer ne contiennent aucune clé en double.
  – Ne qualifiez pas les colonnes dans la clause ORDER BY à l'aide d'un nom de schéma ou de table.

Même si SQL_stmt doit respecter les exigences précédentes, il constitue un élément puissant car il permet d'indiquer n'importe quel prédicat dans la clause WHERE, dans la mesure où l'expression figurant dans ce prédicat utilise les colonnes des tables.

Exigences liées au mappage du noeud RDB

Pour ce schéma de mappage, n'utilisez pas l'élément SQL_stmt dans l'élément <Xcollection> du fichier DAD. Employez plutôt l'élément RDB_node dans chaque noeud d'élément (*element_node*) supérieur, ainsi que pour chaque noeud d'attribut (attribute_node) et noeud de texte (text_node).

- **Noeud RDB pour noeud d'élément supérieur**

  Le noeud d'élément (*element_node*) supérieur indiqué dans le fichier DAD représente l'élément racine du document XML. Pour indiquer un noeud RDB pour le noeud d'élément supérieur, procédez comme suit :

  – Indiquez toutes les tables associées aux documents XML. Par exemple, le mappage suivant indique trois tables dans le noeud RDB du noeud d'élément <Order>, qui constitue le noeud d'élément supérieur :

```
<element_node name="Order">
            <RDB_node>
            <table name="order_tab"/>
            <table name="part_tab"/>
               <table name="ship_tab"/>
               <condition>
          order_tab.order_key = part_tab.order_key AND
            part_tab.part_key = ship_tab.part_key
               </condition>
            </RDB_node>
```

    L'élément Condition peut être vide ou manquant s'il n'y a qu'une seule table dans la collection.

  – Dans la cas d'une opération de décomposition, ou de l'activation de la collection XML indiquée par le fichier DAD, vous devez indiquer une clé primaire pour chaque table. La clé primaire peut correspondre à une ou plusieurs colonnes, constituant la clé composée. Pour indiquer la clé primaire, ajoutez un attribut key à l'élément table du noeud RDB. Lorsqu'il existe une clé composée, l'attribut key est indiqué par les noms de colonnes de clé séparés par un espace. Par exemple :

```
<table name="part_tab" key="part_key, price"/>
```

    Les informations de décomposition sont ignorées lors de la composition d'un document.

  – Avec l'attribut orderBy, vous pouvez recomposer des documents XML qui contiennent des éléments ou des attributs à occurrences multiples, en rétablissant leur structure d'origine. A l'aide de cet attribut, vous pouvez désigner la colonne qui servira de clé permettant de préserver l'ordre du document. L'attribut orderBy, facultatif, fait partie de l'élément table du fichier DAD.

  Vous devez explicitement indiquer le nom de table et le nom de colonne.

- **Noeud RDB pour chaque noeud d'attribut et noeud de texte**

  Dans ce schéma de mappage, les données résident sur le noeud d'attribut (attribute_node) et sur le noeud de texte (text_node) pour

chaque noeud d'élément (element_node). Par conséquent, l'Extension XML doit connaître l'emplacement de base de données dans lequel il doit rechercher les données. Vous devez indiquer un noeud RDB pour chaque noeud d'attribut et noeud de texte, en communiquant à la procédure mémorisée la table, la colonne et la condition de requête à utiliser pour l'extraction de données. Les valeurs de table et de colonne sont obligatoires, alors que la valeur de condition est facultative.

– Indiquez le nom de la table qui contient les données de colonne. Le nom de table doit être inclus dans le noeud RDB du noeud d'élément supérieur. Dans cet exemple, pour le noeud de texte de l'élément <Price>, la table a pour valeur PART_TAB.

```
<element_node name="Price">
                  <text_node>
                     <RDB_node>
                     <table name="part_tab"/>
                        <column name="price"/>
                        <condition>
                           price > 2500.00
                        </condition>
                     </RDB_node>
                  </text_node>
</element_node>
```

– Indiquez le nom de la colonne qui contient les données relatives au texte de l'élément. Dans l'exemple précédent, la colonne s'appelle PRICE.

– Indiquez une condition de requête si vous voulez qu'elle permette de générer des documents XML. Dans l'exemple précédent, la condition est `price > 2500.00`. Seules les données satisfaisant à cette condition sont incluses dans les documents XML générés. La condition doit correspondre à une clause WHERE valide.

– Dans le cas d'une opération de décomposition, , ou de l'activation de la collection XML indiquée par le fichier DAD, vous devez indiquer le type de colonne pour chaque noeud d'attribut et noeud de texte. Ainsi, vous vous assurez que le type de données est correct pour chaque colonne lors de la création de tables pendant l'activation d'une collection XML. Pour indiquer le type de colonne, ajoutez l'attribut type à l'élément colonne. Par exemple :

  ```
  <column name="order_key" type="integer"/>
  ```

  Les informations de décomposition sont ignorées lors de la composition d'un document.

Dans le cadre du mappage du noeud RDB, vous n'avez pas besoin d'entrer des instructions SQL. Cependant, l'insertion de conditions de requête complexes dans l'élément RDB_node peut s'avérer plus difficile. Par exemple, l'expression ou l'opération *union* est peut-être moins puissante que l'approche SQL-XML.

Exigences liées à la taille des tables pour la décomposition

A partir du mappage du noeud RDB, l'opération de décomposition indique comment un document XML est décomposé dans des tables DB2 par l'extraction de valeurs d'éléments et d'attributs qui sont ensuite insérées dans des lignes de la table. Les valeurs de chaque document XML sont stockées dans une ou plusieurs tables DB2. Chaque table peut comporter jusqu'à 1024 lignes par document décomposé.

Par exemple, si un document XML est décomposé en cinq tables, chacune de ces cinq tables peut comporter jusqu'à 1024 lignes pour ce document. Si la table concerne plusieurs documents, elle peut contenir jusqu'à 1024 lignes pour chacun d'eux. Par exemple, si la table contient 20 documents, elle peut comporter 20480 lignes, soit 1024 lignes x 20 documents.

L'utilisation d'éléments à occurrences multiples (c'est-à-dire d'éléments dont les chemins d'emplacement peuvent apparaître plusieurs fois dans la structure XML) a une incidence sur le nombre de lignes. Par exemple, un document contenant un élément <Part> qui apparaît 20 fois, peut être décomposé en 20 lignes dans une table. Tenez compte de cette restriction relative à la taille des tables lorsque vous utilisez des éléments à occurrences multiples.

Chapitre 4. Administration des données XML

Les tâches d'administration de l'Extension XML consistent à activer la base de données et des colonnes de table pour XML et à mapper des données XML pour vers des structures relationnelles DB2. L'Extension XML met à votre disposition plusieurs outils d'administration que vous utilisez en fonction de vos besoin : développement d'une application en vue de l'exécution de tâches d'administration ou simple utilisation d'un assistant. Les outils disponibles sont les suivants :

- Assistant d'administration de l'Extension XML
- Commande **dxxadm**
- Procédures mémorisées d'administration de l'Extension XML

Ce chapitre décrit les tâches d'administration associées à l'assistant d'administration et à la commande **dxxadm**. Les procédures mémorisées d'administration sont décrites à la section «Procédures mémorisées d'administration» à la page 217.

Pour pouvoir exécuter les tâches décrites dans ce chapitre, vous devez être familiarisé avec les tâches de planification et les concepts décrits à la section «Planification des tâches d'administration» à la page 48.

Le présent chapitre se compose des sections suivantes :

1. «Démarrage de l'assistant d'administration»
2. «Activation d'une base de données pour XML» à la page 75
3. «Insertion d'une DTD dans le référentiel des DTD» à la page 76
4. «Définition de colonnes ou de collections XML» à la page 78
5. «Utilisation des colonnes XML» à la page 78
6. «Utilisation des collections XML» à la page 90

Démarrage de l'assistant d'administration

Cette section vous explique comment configurer et appeler l'assistant d'administration de l'Extension XML.

Configuration de l'assistant d'administration

Avant de commencer, exécutez les étapes d'installation et de configuration de l'assistant d'administration qui figurent dans le fichier readme de votre système d'exploitation. Vous devez également avoir exécuté l'instruction Bind et inclus les logiciels requis dans les instructions CLASSPATH.

- Les instructions Bind sont fournies dans les fichiers Readme de l'assistant et dans le fichier exemple de mise en route :

  /dxx_install/samples/cmd/getstart_prep.cmd

- L'instruction CLASSPATH doit se présenter de la manière suivante (les passages à la ligne ont uniquement pour rôle d'améliorer la présentation) :

```
.;C:\java\db2java.zip;C:\java\runtime.zip;C:\java\sqlj.zip;
C:\dxx\dxxadmin\dxxadmin.jar;C:\dxx\dxxadmin\dxxadmin.cmd;
C:\dxx\dxxadmin\html\dxxahelp*.htm;C:\java\jdk\lib\classes.zip;
C:\java\swingall.jar
```

  Important : Le chemin d'accès associé à l'assistant ne doit pas comporter d'espaces. Si vous avez effectué l'installation de IBM DB2 Universal Database version 7.1 proposée par défaut, SQLLIB\java se situe sous le répertoire Program Files. *Copiez* alors le code Java dans un chemin d'accès plus court. Ne déplacez pas le code Java et ne modifiez pas CLASSPATH ; le Centre de contrôle ne reconnaît que le CLASSPATH spécifié lors de l'installation.

L'assistant d'administration de l'Extension XML utilise un fichier de classe. Son nom complet est le suivant :

com.ibm.dxx.admin.Admin.

Modifiez ce fichier pour que votre système appelle l'assistant.

- Pour l'appeler via le JDK, tapez :

  java -classpath *classpath* com.ibm.dxx.admin.Admin

- Pour l'appeler via le JRE, tapez :

  jre -classpath *classpath* com.ibm.dxx.admin.Admin

  où *classpath* désigne l'un des éléments suivants :

  – La variable d'environnement %CLASSPATH% qui indique l'emplacement des fichiers de classe de l'assistant d'administration. Lorsque vous utilisez cette option, votre variable système CLASSPATH doit pointer sur le répertoire *dxx_install*/dxxadmin, qui contient les fichiers suivants : dxxadmin.jar, xml4j.jar et db2java.zip. Par exemple :

    java -classpath %CLASSPATH% com.ibm.dxx.admin.Admin

  – Une variable de substitution de la variable d'environnement %CLASSPATH%, qui pointe sur les fichiers du répertoire *dxx_install*/dxxadmin, à partir duquel vous exécutez l'assistant d'administration de l'Extension XML. Par exemple :

```
java -classpath dxxadmin.jar;xml4j.jar;db2java.zip com.ibm.dxx.admin.Admin url=jdbc:db2:mydb
   userid=db2xml password=db2xml driver=COM.ibm.db2.jdbc.app.DB2Driver
```

Vous pouvez facultativement spécifier les paramètres suivants lors de l'exécution :

url Chemin d'accès URL complet à la source de données IBM DB2 UDB à laquelle une connexion doit s'effectuer. Par exemple : jdbc:db2://dxx.stl.ibm.com:8080/guidb. Désigné par «Adresse» dans l'assistant.

userid ID utilisateur permettant d'accéder à la source de données précédemment indiquée. Par exemple : db2guest.

password
Mot de passe associé à l'ID utilisateur. Par exemple : guest.

driver Nom du pilote JDBC associé à l'URL précédemment mentionné. Par défaut : COM.ibm.db2.jdbc.net.DB2Driver. Désigné par «Pilote JDBC» dans l'assistant.

Pour plus d'informations sur ces valeurs, reportez-vous à la section «Appel de l'assistant d'administration».

Appel de l'assistant d'administration

Procédure :

1. Appelez l'assistant.

   Sous Windows NT :

   A partir du bureau, cliquez deux fois sur l'icône de l'assistant d'administration de l'Extension XML.

   Sous AIX, Sun Solaris et Linux :

   Exécutez le fichier `dxxadmin`.

   La fenêtre de connexion à l'assistant d'administration s'affiche.

Lorsque vous appelez l'assistant d'administration de l'Extension XML, la fenêtre de connexion s'affiche. Connectez-vous à la base de données à utiliser pour les données XML. L'Extension XML se connecte à l'instance en cours.

2. Dans la zone **Adresse**, entrez l'adresse JDBC qualifiée complète de la source de données IBM DB2 UDB à laquelle vous voulez vous connecter. La syntaxe de cette adresse est la suivante :

   Pour les configurations autonomes (recommandé) :

   `jdbc:db2:`*`database_name`*

   Où :

   database_name
   Nom de la base de données à utiliser pour la connexion et le stockage de documents XML.

Par exemple :

```
jdbc:db2:sales_db
```

Pour les configurations de réseau :

```
jdbc:db2://server_name:port_number/database_name
```

Où :

server_name
Nom du serveur où réside l'Extension XML.

port_number
Numéro de port à utiliser pour la connexion au serveur. Pour le connaître, tapez la commande suivante sur la ligne de commande DB2 du poste serveur :

```
db2jstrt port#
```

Les utilisateurs de Windows NT peuvent vérifier le numéro de port dans le fichier suivant : \winnt\system32\driver\etc\services.

database_name
Nom de la base de données à utiliser pour la connexion et le stockage de documents XML.

Par exemple :

```
jdbc:db2://host1.ibm.com:8080/sales_db
```

3. Dans les zones **ID util** et **Mot de passe**, tapez ou confirmez l'ID utilisateur et le mot de passe DB2 d'accès à la base de données souhaitée.
4. Dans la zone **Pilote JDBC**, confirmez le nom du pilote JDBC correspondant à l'adresse indiquée à l'aide des valeurs suivantes :

   Pour les configurations autonomes (par défaut et recommandé) :

   ```
   COM.ibm.db2.jdbc.app.DB2DRIVER
   ```

   Pour les configurations de réseau :

   ```
   COM.ibm.db2.jdbc.net.DB2DRIVER
   ```

5. Cliquez sur **Fin** pour vous connecter à l'assistant et passer au tableau de bord.

Le tableau de bord permet d'accéder à cinq assistants d'administration. Avec ces assistants, vous pouvez effectuer les tâches suivantes :

- Activation d'une base de données pour XML
- Insertion d'une DTD dans le référentiel des DTD
- Création ou édition d'un fichier DAD pour les :
  - Colonnes XML

  - Collections XML
- Utilisation des colonnes XML
- Utilisation des collections XML

Activation d'une base de données pour XML

Pour stocker ou extraire des documents XML à partir de DB2 avec l'Extension XML, vous activez une base de données pour XML. L'Extension XML active la base de données à laquelle vous êtes connecté, en utilisant l'instance en cours.

Lorsque vous activez une base de données pour XML, l'Extension XML exécute les opérations suivantes :

- Création de tous les types UDT et de toutes les fonctions UDF
- Création et peuplement des tables de contrôle avec les métadonnées requises par l'Extension XML
- Création du schéma db2xml et affectation des privilèges nécessaires

  Le nom complet d'une fonction XML se présente sous la forme *nom-schéma.nom-fonction*, où *nom-schéma* est un identificateur permettant un regroupement logique des *objets* SQL. Vous pouvez utiliser ce nom complet chaque fois que vous faites référence à des types UDT ou à des fonctions UDF. Vous pouvez également omettre le nom de schéma. Dans ce cas, DB2 utilise le chemin d'accès à la fonction pour déterminer la fonction ou le type de données recherché.

A l'aide de l'assistant d'administration

Pour activer une base de données pour les données XML, procédez comme suit :

1. Configurez et démarrez l'assistant d'administration. Pour plus d'informations, reportez-vous à la section «Démarrage de l'assistant d'administration» à la page 71.
2. A partir du tableau de bord, cliquez sur **Activation de la base de données** pour activer la base de données en cours.

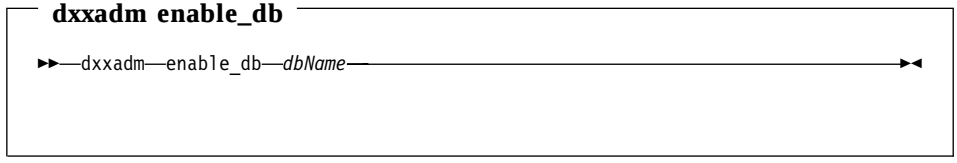
   Si une base de données est déjà activée, seule l'option **Désactivation de la base de données** est disponible.

   Une fois la base de données activée, vous revenez au tableau de bord.

A partir du shell de commandes DB2

Tapez **dxxadm** sur la ligne de commande en indiquant la base de données à activer.

Syntaxe :



Paramètres :

dbName
Nom de la base de données à activer.

Exemple : Activation d'une base de données existante appelée SALES_DB.

```
dxxadm enable_db SALES_DB
```

Insertion d'une DTD dans le référentiel des DTD

La définition ou déclaration du type de document (DTD) permet de valider des données XML dans une colonne ou une collection XML. Cette DTD valide la colonne XML et permet de définir les fichiers de définition d'accès à un document (DAD) utilisés pour la recherche structurelle XML et pour la composition et la décomposition des collections.

Toutes les DTD sont stockées dans le référentiel des DTD, une table DB2 appelée DTD_REF. Le nom du schéma associé est db2xml. Chaque DTD de la table DTD_REF possède un ID unique. L'Extension XML crée la table DTD_REF lorsque vous activez une base de données pour XML.

Pour plus d'informations sur l'utilisation des DTD, reportez-vous aux sections «Planification des colonnes XML» à la page 51 et «Planification des collections XML» à la page 58.

Vous pouvez insérer la DTD à partir du shell de commandes DB2 ou à l'aide de l'assistant d'administration.

A l'aide de l'assistant d'administration

Pour insérer une DTD, procédez comme suit :

1. Configurez et démarrez l'assistant d'administration. Pour plus d'informations, reportez-vous à la section «Démarrage de l'assistant d'administration» à la page 71.

2. A partir du tableau de bord, cliquez sur **Importation d'une DTD** pour importer un fichier DTD dans le référentiel des DTD de la base de données en cours. La fenêtre Importation d'une DTD s'affiche.
3. Tapez le nom du fichier DTD dans la zone **Nom de fichier DTD**, ou cliquez sur **...** pour parcourir la liste des DTD disponibles.
4. Tapez l'identificateur de la DTD dans la zone **ID DTD**.

   Cette valeur identifie la DTD et peut correspondre à son chemin dans le système local. Cet ID doit correspondre à la valeur indiquée dans le fichier DAD associé à l'élément <IDDTD>.
5. Vous pouvez également indiquer l'auteur de la DTD dans la zone **Auteur**.

   L'Extension XML affiche automatiquement le nom de l'auteur si ce dernier est indiqué dans la DTD.
6. Cliquez sur **Fin** pour insérer la DTD dans le référentiel des DTD (table DB2XML.DTD_REF) et revenir au tableau de bord.

A partir du shell de commandes DB2

Lancez l'instruction SQL INSERT pour la table DTD_REF en utilisant le schéma du tableau 7:

Tableau 7. Schéma de la table DTD_REF

Nom de colonne	Type de données	Description
DTDID	VARCHAR(128)	Clé primaire (unique et différente de NULL). La clé primaire permet d'identifier la DTD et doit être identique au SYSTEM ID de la ligne DOCTYPE de chaque document XML, en cas de validation. Lorsque la clé primaire est indiquée dans le fichier DAD, ce dernier doit être conforme au schéma défini par la DTD.
CONTENT	XMLCLOB	Contenu de la DTD.
USAGE_COUNT	INTEGER	Nombre de colonnes XML et de collections XML de la base de données qui utilisent cette DTD pour définir une DAD.
AUTHOR	VARCHAR(128)	Auteur de la DTD, information facultative destinée à l'utilisateur (en entrée).
CREATOR	VARCHAR(128)	ID utilisateur auteur de la première insertion.
UPDATOR	VARCHAR(128)	ID utilisateur auteur de la dernière mise à jour.

Par exemple :

```
DB2 INSERT into db2xml.dtd_ref values('c:\dxx\samples\dtd\getstart.dtd',
    db2xml.XMLClobFromFile('c:\dxx\samples\dtd\getstart.dtd'), 0, 'user1',
    'user1', 'user1')
```

Remarque importante concernant les collections XML : L'ID de DTD est un chemin indiquant l'emplacement de la DTD sur le système local. Cet ID doit correspondre à la valeur indiquée dans le fichier DAD associé à l'élément <IDDTD>.

Définition de colonnes ou de collections XML

Les sections suivantes expliquent comment configurer et définir la base de données pour les colonnes ou collections XML, et préparer les schémas de mappage de données nécessaires.

Ces sections sont les suivantes :

- «Utilisation des colonnes XML»
- «Utilisation des collections XML» à la page 90

Utilisation des colonnes XML

Pour configurer les colonnes XML, vous devez définir le fichier DAD afin de pouvoir accéder aux données XML et activer les colonnes pour les données XML d'une table XML. Lors de la création de la DAD, il est important de comprendre la syntaxe du chemin d'emplacement, car celui-ci permet de mapper les valeurs d'éléments et d'attributs à indexer par rapport aux tables DB2. Pour plus d'informations sur le chemin d'emplacement et sa syntaxe, reportez-vous à la section «Chemin d'emplacement» à la page 55.

Création ou édition du fichier DAD

Lorsque vous indiquez un fichier DAD, vous définissez les attributs et les éléments clés des données à examiner. L'Extension XML crée des tables annexes à partir de ces informations de sorte que vous puissiez indexer vos données pour les extraire rapidement. Pour obtenir des informations sur les tâches de planification à effectuer pour créer le fichier DAD, reportez-vous à la section «Fichier DAD» à la page 58.

Avant de commencer

- Comprenez la structure hiérarchique de vos données XML pour pouvoir définir les éléments et les attributs clés pour l'indexation et la recherche rapide.
- Préparez et insérez la DTD du document XML dans la table DTD_REF. Cette étape est obligatoire pour la validation.

A l'aide de l'assistant d'administration

Pour créer un fichier DAD :

1. Configurez et démarrez l'assistant d'administration. Pour plus d'informations, reportez-vous à la section «Démarrage de l'assistant d'administration» à la page 71.
2. A partir du tableau de bord, cliquez sur **Utilisation des fichiers DAD** pour éditer ou créer un fichier DAD XML. La fenêtre Spécification d'un fichier DAD s'affiche.
3. Vous pouvez éditer un fichier DAD existant ou en créer un nouveau.
   - **Pour éditer une DAD existante :**
     a. Cliquez sur **...** pour parcourir la liste des DAD disponibles dans le menu déroulant, ou tapez le nom de fichier DAD dans la zone **Nom de fichier**.
     b. Vérifiez que l'assistant reconnaît le fichier DAD indiqué.
        – Dans l'affirmative, le bouton **Suivant** devient disponible et Colonne XML s'affiche dans la zone **Type**.
        – Dans la négative, vous ne pouvez pas cliquer sur **Suivant**. Tapez un autre nom de fichier existant, ou cliquez sur **Ouverture** pour parcourir à nouveau la liste des DAD disponibles. Réessayez jusqu'à ce que le bouton **Suivant** devienne disponible.
     c. Cliquez sur **Suivant**.
   - **Pour créer un fichier DAD :**
     a. Laissez la zone **Nom de fichier** à blanc.
     b. Dans le menu **Type**, cliquez sur **Colonne XML**.
     c. Cliquez sur **Suivant**.
4. Dans la fenêtre Sélection de la validation, vous pouvez choisir de valider vos documents XML avec une DTD.
   - Pour effectuer la validation :
     a. Cliquez sur **Validation des documents XML avec la DTD**.
     b. Dans le menu **ID DTD**, sélectionnez la DTD à utiliser pour la validation.

     Si vous n'avez importé aucune DTD dans le référentiel des DTD de votre base de données, vous ne pouvez pas valider vos documents XML.
   - Cliquez sur **Pas de validation des documents XML avec la DTD** pour poursuivre l'opération sans valider vos documents XML.
5. Cliquez sur **Suivant**.

6. Vous pouvez ajouter, éditer ou retirer une table annexe dans la fenêtre Tables annexes.
   - **Pour ajouter une table annexe ou une colonne de table annexe :**

     Pour ajouter une table annexe, vous devez définir ses colonnes. Procédez comme suit pour chaque colonne de la table annexe.

     a. Remplissez les zones de la boîte d'options **Détails** dans la fenêtre Tables annexes.

        1) **Nom de la table** : Tapez le nom de la table qui contient la colonne. Par exemple :

           ```
           ORDER_SIDE_TAB
           ```

        2) **Nom de la colonne** : Tapez le nom de la colonne. Par exemple :

           ```
           CUSTOMER_NAME
           ```

        3) **Type** : Sélectionnez un type de colonne dans le menu. Par exemple :

           ```
           XMLVARCHAR
           ```

        4) **Longueur (type VARCHAR uniquement)** : Tapez le nombre maximal de caractères VARCHAR. Par exemple :

           ```
           30
           ```

        5) **Chemin d'accès** : Tapez le chemin de l'élément ou de l'attribut. Par exemple :

           ```
           /ORDER/CUSTOMER/NAME
           ```

           Pour consulter la syntaxe du chemin d'emplacement, reportez-vous à la section «Chemin d'emplacement» à la page 55.

        6) **Occurrences multiples** : Sélectionnez **Oui** ou **Non** dans le menu.

           Cette option indique si le chemin d'emplacement de cet élément ou attribut peut être utilisé plusieurs fois dans un même document.

           Important : Si vous cochez la case Occurrences multiples pour une colonne, vous ne pouvez indiquer qu'une seule colonne dans la table annexe qui la contient.

     b. Cliquez sur **Ajout** pour ajouter une colonne.

     c. Poursuivez les opérations d'ajout, d'édition ou de retrait de colonnes pour la table annexe, ou cliquez sur **Suivant**.

   - **Pour éditer une colonne de table annexe existante, procédez comme suit :**

     Pour mettre à jour une table annexe, modifiez les définitions des colonnes existantes.

     a. Dans la table annexe, cliquez sur le nom de colonne à éditer.

     b. Editez les zones de la boîte d'options **Détails**.

     c. Cliquez sur **Modification** pour sauvegarder vos modifications.

d. Poursuivez les opérations d'ajout, d'édition ou de retrait de colonnes pour chaque table annexe, ou cliquez sur **Suivant**.

- **Pour retirer une colonne de table annexe existante :**
  a. Dans la table annexe, cliquez sur la colonne à retirer.
  b. Cliquez sur **Retrait**.
  c. Poursuivez les opérations d'ajout, d'édition ou de retrait de colonnes de table annexe, ou cliquez sur **Suivant**.
- **Pour retirer une table annexe existante :**

  Pour retirer toute une table annexe, supprimez chacune de ses colonnes.
  a. Cliquez sur chaque colonne de la table annexe à retirer.
  b. Cliquez sur **Retrait**.
  c. Poursuivez les opérations d'ajout, d'édition ou de retrait de colonnes de tables annexes, ou cliquez sur **Suivant**.

7. Dans la zone **Nom de fichier** de la fenêtre Spécification d'une DAD, tapez un nom de fichier de sortie pour la DAD modifiée.
8. Cliquez sur **Fin** pour sauvegarder le fichier DAD et revenir au tableau de bord.

A partir du shell de commandes DB2

Le fichier DAD est un fichier XML qui peut être créé avec n'importe quel éditeur de texte.

Pour créer un fichier DAD :

1. Ouvrez un éditeur de texte.
2. Créez l'en-tête du fichier DAD en respectant la syntaxe suivante :

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "path\dtd\dad.dtd"> --> Chemin et nom de fichier de
                                              la DTD pour le fichier DAD
```

3. Insérez les balises <DAD></DAD>.
4. Dans la balise <DAD>, vous pouvez indiquer l'ID DTD qui associe le fichier DAD à la DTD du document XML pour la validation :

```
<dtdid>path\dtd_name.dtd</dtdid> --> Chemin et nom de fichier de la DTD
                                     pour l'application
```

   L'ID DTD est obligatoire pour la validation et doit être le même que celui utilisé pour l'insertion de la DTD dans le référentiel des DTD (table db2xml.DTD_REF).
5. Indiquez si vous voulez une validation (avec la DTD pour vérifier que le document XML est valide). Par exemple :

```
<validation>YES</validation>  --> Indiquez YES ou NO
```

Si vous répondez OUI, vous devez avoir indiqué un ID DTD à l'étape précédente et inséré une DTD dans la table DTD_REF.

6. A l'aide de l'élément <Xcolumn>, définissez la méthode d'accès et de stockage colonne XML.

```
<Xcolumn>
</Xcolumn>
```

7. Définissez chaque table annexe, ainsi que les éléments et les attributs essentiels à indexer pour la recherche structurelle. Suivez les étapes ci-après pour chaque table. Elles utilisent des exemples provenant d'un modèle de fichier DAD illustré à la section «Fichier DAD : colonne XML» à la page 279 :
   a. Insérez les balises <TABLE></TABLE> et l'attribut de nom.

```
<table name="order_tab">
</table>
```

   b. Après la balise <TABLE>, insérez une balise <COLUMN> et les attributs associés pour chaque colonne de la table :
      - ***name*** : Nom de la colonne
      - ***type*** : Type de la colonne
      - ***path*** : Chemin d'emplacement de l'élément ou de l'attribut. Pour consulter la syntaxe du chemin d'emplacement, reportez-vous à la section «Chemin d'emplacement» à la page 55.
      - ***multi_occurrence*** : Indication permettant de savoir si un élément ou un attribut peut être utilisé plusieurs fois dans un document

```
<table ...>
   <column name="order_key"
           type="integer"
           path="/Order/@key"
           multi_occurrence="NO"/>
   <column name="customer"
           type="varchar(50)"
           path="/Order/Customer/Name"
           multi_occurrence="NO"/>
</table>
```

8. Vérifiez que vous avez placé une balise de fin </TABLE> après la dernière définition de colonne.
9. Vérifiez que vous avez placé une balise de fin </Xcolumn> après la dernière balise </TABLE>.
10. Vérifiez que vous avez placé une balise de fin </DAD> après la balise </Xcolumn>.

Création ou modification d'une table XML

Pour pouvoir stocker des documents XML entiers dans une table, vous devez créer ou modifier celle-ci pour qu'elle contienne une colonne associée à un type UDT XML. Cette table est désignée *table XML*, c'est-à-dire qu'elle

contient des documents XML. Il peut s'agir d'une table modifiée ou nouvelle. Une fois que la table contient une colonne de type XML, vous pouvez activer cette colonne pour l'Extension XML.

Vous pouvez modifier une table existante à l'aide de l'assistant d'administration ou du shell de commandes DB2.

A l'aide de l'assistant d'administration

1. Configurez et démarrez l'assistant d'administration. Pour plus d'informations, reportez-vous à la section «Démarrage de l'assistant d'administration» à la page 71.
2. A partir du tableau de bord, cliquez sur **Utilisation des colonnes XML**. La fenêtre Sélection d'une tâche s'affiche.
3. Cliquez sur **Ajout d'une colonne XML**. La fenêtre Ajout d'une colonne XML s'affiche.
4. Tapez le nom de la table à modifier ou sélectionnez-le dans le menu déroulant **Nom de la table**. Par exemple :

   ```
   SALES_DB
   ```

5. Dans la zone **Nom de la colonne**, tapez le nom de la colonne à ajouter. Par exemple :

   ```
   ORDER
   ```

6. Dans le menu déroulant **Type de la colonne**, sélectionnez un type UDT pour la colonne. Par exemple :

   ```
   XMLVARCHAR
   ```

7. Cliquez sur **Fin** pour ajouter la colonne de type XML.

A partir du shell de commandes DB2

Créez ou modifiez une table dotée d'une colonne XML dans la clause colonne de l'instruction CREATE TABLE ou ALTER TABLE.

Exemple : Dans l'application SALES, vous stockez une commande client au format XML dans la colonne ORDER de la table d'application SALES_TAB. Cette table est également dotée des colonnes INVOICE_NUM et SALES_PERSON. S'agissant d'une commande peu volumineuse, vous la stockez sous le type de données XMLVARCHAR. La clé primaire est INVOICE_NUM. L'instruction CREATE TABLE ci-après crée la table avec une colonne de type XML :

```
CREATE TABLE sales_tab(
             invoice_num  char(6) NOT NULL PRIMARY KEY,
             sales_person     varchar(20),
             order            XMLVarchar);
```

Activation de colonnes XML

Pour pouvoir stocker un document XML dans une base de données DB2, vous devez activer une colonne pour l'Extension XML. Lorsque vous activez une

colonne, vous la préparez en vue de son indexation pour optimiser les recherches. Vous pouvez activer une colonne à l'aide de l'assistant d'administration de l'Extension XML ou du shell de commandes DB2. Le colonne doit être de type XML.

Lorsque DB2 Extension XML active une colonne XML, il exécute les tâches suivantes :

- Lecture du fichier DAD pour éventuellement :
  - valider le fichier DAD par rapport à la DTD correspondante,
  - extraire l'ID DTD de la table DTD_REF,
  - créer des tables annexes pour l'indexation de la colonne XML,
  - préparer la colonne pour qu'elle puisse contenir des données XML.
- Création éventuelle d'une *vue par défaut* de la table XML et des tables annexes.
- Définition d'une valeur ID ROOT, s'il n'en existe aucune.

Une fois la colonne XML activée, vous pouvez effectuer les tâches suivantes :

- Indexation des tables annexes.
- Insertion de documents XML dans la colonne XML.
- Lancement d'opérations d'interrogation, de mise à jour ou de recherche sur les documents XML de la colonne XML.

Avant de commencer

Générez une table XML en créant ou en modifiant une table DB2 dotée d'une colonne de type UDT XML.

A l'aide de l'assistant d'administration

Pour activer des colonnes XML :

1. Configurez et démarrez l'assistant d'administration. Pour plus d'informations, reportez-vous à la section «Démarrage de l'assistant d'administration» à la page 71.
2. A partir du tableau de bord, cliquez sur **Utilisation des colonnes XML** pour afficher les tâches liées aux colonnes de l'Extension XML. La fenêtre Sélection d'une tâche s'affiche.
3. Cliquez sur **Activation d'une colonne** puis sur **Suivant**, pour activer une colonne de table existant dans la base de données.
4. Dans la zone **Nom de la table**, sélectionnez la table qui contient la colonne XML. Par exemple :

   ```
   SALES_TAB
   ```

5. Dans la zone **Nom de la colonne**, sélectionnez la colonne à activer. Par exemple :

   ```
   ORDER
   ```

   Il doit s'agir d'une colonne existante de type XML.

6. Tapez le chemin et le nom du fichier DAD dans la zone **Nom de fichier DAD**, ou cliquez sur **...** pour parcourir la liste des DAD disponibles. Par exemple :

   ```
   c:\dxx\samples\dad\getstart.dad
   ```

7. Vous pouvez également indiquer dans la zone **Espace table** le nom d'un espace table existant.

   L'espace table contient les tables annexes générées par l'Extension XML. Les tables annexes sont créées dans l'espace table que vous précisez, ou dans l'espace table par défaut, si vous n'en indiquez aucun.

8. Vous pouvez également indiquer dans la zone **Vue par défaut** le nom de la vue par défaut.

   Si vous l'indiquez, la vue par défaut est automatiquement générée lors de la création de la colonne. Elle effectue la jointure de la table XML et de toutes ses tables annexes.

9. Vous pouvez également indiquer dans la zone **ID racine** (ROOT ID) le nom de colonne de la clé primaire figurant dans la table d'application. Cette méthode est recommandée.

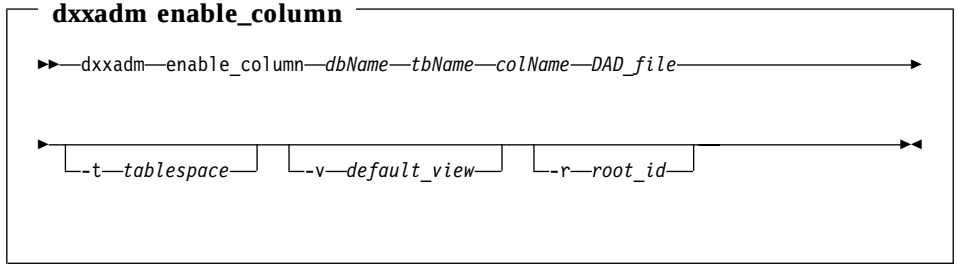
   L'Extension XML utilise la valeur ROOT ID en tant qu'identificateur unique pour associer toutes les tables annexes à la table d'application. Si vous n'indiquez aucun ID racine, l'Extension XML ajoute une colonne DXXROOT_ID dans la table d'application et génère un identificateur par défaut.

10. Cliquez sur **Fin** pour activer la colonne XML, créer les tables annexes et revenir au tableau de bord.
    - Si l'activation de la colonne aboutit, un message d'information l'indique.
    - Si l'activation de la colonne échoue, un message d'erreur s'affiche. Corrigez les valeurs dans la zone d'entrée jusqu'à l'aboutissement de l'opération.

A partir du shell de commandes DB2

Pour activer une colonne XML, entrez la commande suivante :

Syntaxe :



Paramètres :

dbName
Nom de la base de données.

tbName
Nom de la table contenant la colonne à activer.

colName
Nom de la colonne XML activée.

DAD_file
Nom du fichier contenant la DAD (définition d'accès au document).

tablespace
Espace table existant contenant les tables annexes générées par Extension XML. Si vous laissez cette zone à blanc, l'Extension XML utilise un espace table par défaut.

default_view
Facultatif. Nom de la vue par défaut créée par Extension XML pour réaliser la jointure de la table d'application et de toutes ses tables annexes.

root_id Facultatif. Nom de colonne de la clé primaire appartenant à la table d'application et identificateur unique associant toutes les tables annexes à cette table d'application. Nous vous recommandons d'indiquer la valeur ROOT ID. Si vous ne le faites pas, l'Extension XML ajoute une colonne DXXROOT_ID dans la table d'application et génère un identificateur par défaut.

Restriction : Si la table d'application comporte une colonne DXXROOT_ID, mais que celle-ci ne contient pas de valeur pour le paramètre *root_id*, vous devez définir cet identificateur racine sinon une erreur se produit.

Exemple : L'exemple ci-après illustre l'activation d'une colonne à partir du shell de commandes DB2. Le fichier DAD et le document XML se trouvent à l'«Annexe B. Exemples» à la page 277.

```
dxxadm enable_column SALES_DB sales_tab order getstart.dad
           -v sales_order_view -r invoice_num
```

Dans cet exemple, la colonne ORDER est activée dans la table SALES_DB.SALES_TAB. Soient un fichier DAD `getstart.dad`, une vue par défaut sales_order_view et une valeur ROOT ID INVOICE_NUM.

Le schéma de la table SALES_TAB est désormais le suivant :

Nom de colonne	INVOICE_NUM	SALES_PERSON	ORDER
Type de données	CHAR(6)	VARCHAR(20)	XMLVARCHAR

Les tables annexes suivantes sont créées en fonction de la définition DAD :

ORDER_SIDE_TAB :

Nom de colonne	ORDER_KEY	CUSTOMER	INVOICE_NUM
Type de données	INTEGER	VARCHAR(50)	CHAR(6)
Chemin d'accès	`/Order/@key`	`/Order/Customer/Name`	N/A

PART_SIDE_TAB:

Nom de colonne	PART_KEY	PRICE	INVOICE_NUM
Type de données	INTEGER	DOUBLE	CHAR(6)
Chemin d'accès	`/Order/Part/@key`	`/Order/Part/ExtendedPrice`	N/A

SHIP_SIDE_TAB:

Nom de colonne	DATE	INVOICE_NUM
Type de données	DATE	CHAR(6)
Chemin d'accès	`/Order/Part/Shipment/ShipDate`	N/A

Toutes les tables annexes possèdent une colonne INVOICE_NUM de même type car la valeur du paramètre ROOT ID est indiquée par la clé primaire INVOICE_NUM de la table d'application. Une fois la colonne activée, la valeur INVOICE_NUM est insérée dans les tables annexes. La définition du paramètre *default_view* lors de l'activation de la colonne XML ORDER entraîne

la création de la vue par défaut sales_order_view. Cette vue réalise la jointure des tables ci-dessus à l'aide de l'instruction suivante :

```
CREATE VIEW sales_order_view(invoice_num, sales_person, order,
                                 order_key, customer, part_key, price, date)
     AS
     SELECT sales_tab.invoice_num, sales_tab.sales_person, sales_tab.order,
            order_tab.order_key, order_tab.customer,
            part_tab.part_key, part_tab.price,
            ship_tab.date
     FROM sales_tab, order_tab, part_tab, ship_tab
     WHERE sales_tab.invoice_num = order_tab.invoice_num
               AND sales_tab.invoice_num = part_tab.invoice_num
               AND sales_tab.invoice_num = ship_tab.invoice_num
```

Si vous précisez un espace table dans la commande **enable_column**, les tables annexes sont créés dans celui-ci. Si vous n'indiquez aucun espace table, les tables annexes sont créées dans l'espace table par défaut.

Indexation des tables annexes

Une fois la colonne XML activée et les tables annexes créées, vous pouvez indexer ces dernières. Les tables annexes contiennent des données XML dans les colonnes que vous avez indiquées lors de la création du fichier DAD. L'indexation de ces tables permet d'améliorer les performances de recherche dans les documents XML.

Avant de commencer

- Créez un fichier DAD qui indique des tables annexes pour la structure du document XML.
- Activez la colonne XML à l'aide du fichier DAD, ce qui entraîne la création des tables annexes.

Commande DB2 CREATE INDEX

Utilisez la commande DB2 CREATE INDEX.

Exemple :

L'exemple suivant illustre l'indexation de quatre tables annexes :

```
DB2 CREATE INDEX KEY_IDX
      ON ORDER_SIDE_TAB(ORDER_KEY)

DB2 CREATE INDEX CUSTOMER_IDX
      ON ORDER_SIDE_TAB(CUSTOMER)

DB2 CREATE INDEX PRICE_IDX
      ON PART_SIDE_TAB(PRICE)

DB2 CREATE INDEX DATE_IDX
      ON SHIP_SIDE_TAB(DATE)
```

Désactivation de colonnes XML

Si vous devez mettre à jour une DAD de colonne XML ou que vous voulez supprimer la colonne XML ou la table qui la contient, désactivez la colonne XML. Une fois la colonne désactivée et l'opération terminée (par exemple, mise à jour du fichier DAD), vous pouvez la réactiver. Pour désactiver la colonne, vous faites appel à l'assistant d'administration de l'Extension XML ou au shell de commandes DB2.

Lorsque DB2 Extension XML active une colonne XML, il exécute les tâches suivantes :

- Suppression de l'entrée colonne de la table XML_USAGE.
- Retrait des tables annexes associées à cette colonne.

Important : Si vous retirez une table dotée d'une colonne XML sans désactiver celle-ci au préalable, l'Extension XML ne peut pas retirer les tables annexes associées à cette colonne XML, ce qui peut provoquer des résultats inattendus.

Avant de commencer

Assurez-vous que la colonne XML à désactiver existe dans la base de données DB2 en cours.

A l'aide de l'assistant d'administration

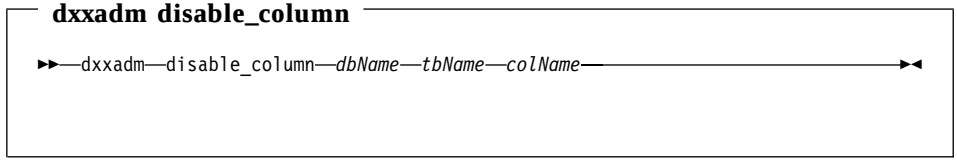
Pour désactiver des colonnes XML :

1. Configurez et démarrez l'assistant d'administration. Pour plus d'informations, reportez-vous à la section «Démarrage de l'assistant d'administration» à la page 71.
2. A partir du tableau de bord, cliquez sur **Utilisation des colonnes XML** pour afficher les tâches liées aux colonnes de l'Extension XML. La fenêtre Sélection d'une tâche s'affiche.
3. Cliquez sur **Désactivation d'une colonne** puis sur **Suivant**, pour désactiver une colonne de table existant dans la base de données.
4. Dans la zone **Nom de la table**, sélectionnez la table qui contient la colonne XML.
5. Dans la zone **Nom de la colonne**, sélectionnez la colonne à désactiver.
6. Cliquez sur **Fin**.
   - Si la désactivation de la colonne aboutit, un message d'information l'indique.
   - Si la désactivation de la colonne échoue, un message d'erreur s'affiche. Corrigez les valeurs dans la zone d'entrée jusqu'à l'aboutissement de l'opération.

A partir du shell de commandes DB2

Pour désactiver une colonne XML, entrez la commande suivante :

Syntaxe :



Paramètres :

dbName
Nom de la base de données.

tbName
Nom de la table contenant la colonne à désactiver.

colName
Nom de la colonne XML désactivée.

Exemple : L'exemple ci-après illustre la désactivation d'une colonne à partir du shell de commandes DB2. Le fichier DAD et le document XML se trouvent à l'«Annexe B. Exemples» à la page 277.

```
dxxadm disable_column SALES_DB sales_tab order
```

Dans cet exemple, la colonne ORDER est désactivée dans la table SALES_DB.SALES_TAB.

Lorsque la colonne XML est désactivée, les tables annexes sont retirées.

Utilisation des collections XML

Lorsque vous configurez des collections XML, vous avez besoin d'un schéma de mappage. Vous pouvez également activer la collection avec un nom virtuel associant les tables DB2 au fichier DAD.

L'activation de la collection XML n'est pas obligatoire mais optimise les performances.

Création ou édition du fichier DAD pour le schéma de mappage

Vous devez créer un fichier DAD lorsque vous utilisez des collections XML. Un fichier DAD définit la relation existant entre les données XML et plusieurs tables relationnelles. L'Extension XML exploite le fichier DAD pour :

- composer un document XML à partir de données relationnelles,

- décomposer un document XML en données relationnelles.

Il existe deux méthodes possibles pour mapper les données entre les tables XML et la table DB2 : le mappage SQL et le mappage du noeud RDB.

Mappage SQL
Fait appel à un élément d'instruction SQL en vue d'indiquer la requête SQL pour les tables et les colonnes contenant les données XML. Ne peut être utilisé que pour la composition de documents XML.

Mappage de noeud RDB
A recours à un élément spécifique à l'Extension XML, le noeud de base de données relationnelle (RDB_node), qui indique des tables, des colonnes, des conditions et l'ordre des données XML. Le mappage du noeud RDB supporte des mappages plus complexes que ceux fournis par une instruction SQL. Peut être utilisé pour la composition et la décomposition de documents XML.

Les deux méthodes de mappage utilisent le *modèle de données XPath*, décrit à la section «Fichier DAD» à la page 60.

Avant de commencer
- Mappez la relation existant entre les tables DB2 et le document XML. Au cours de cette étape, vous devez également mapper la hiérarchie du document XML et préciser le mode de mappage des données de ce document vers une table DB2.
- Si vous prévoyez de valider les documents XML, insérez dans la table de référence des DTD (db2xml.DTD_REF) la DTD du document XML en cours de composition ou de décomposition.

Composition de documents XML en mode mappage SQL
Choisissez le mappage SQL lorsque vous composez des documents XML et que vous voulez utiliser SQL.

A l'aide de l'assistant d'administration:

Pour créer une DAD de composition en mode mappage SQL, procédez comme suit :

Choisissez le mappage SQL lorsque vous composez un document XML et que vous voulez utiliser une instruction SQL pour définir la table et les colonnes à partir desquelles vous dérivez les données du document XML.

1. Configurez et démarrez l'assistant d'administration. Pour plus d'informations, reportez-vous à la section «Démarrage de l'assistant d'administration» à la page 71.

2. A partir du tableau de bord, cliquez sur **Utilisation des fichiers DAD**. La fenêtre Spécification d'une DAD s'affiche.
3. Vous pouvez éditer un fichier DAD existant ou en créer un nouveau.

   Pour créer un fichier DAD :

   a. Laissez la zone **Nom de fichier** à blanc.
   b. Dans le menu **Type**, sélectionnez **Mappage SQL de collection XML**.
   c. Cliquez sur **Suivant** pour ouvrir la fenêtre Sélection de la validation.

   Pour éditer un fichier DAD existant :

   a. Tapez le nom du fichier DAD dans la zone **Nom de fichier**, ou cliquez sur **...** pour parcourir la liste des DAD disponibles.
   b. Vérifiez que l'assistant reconnaît le fichier DAD indiqué.
      - Dans l'affirmative, le bouton **Suivant** devient disponible et Mappage SQL d'une collection XML s'affiche dans la zone **Type**.
      - Dans la négative, vous ne pouvez pas cliquer sur **Suivant**. Tapez un autre nom de fichier existant, ou cliquez sur **...** pour parcourir à nouveau la liste des DAD disponibles. Corrigez les valeurs dans la zone d'entrée jusqu'à ce que **Suivant** devienne disponible.
   c. Cliquez sur **Suivant** pour ouvrir la fenêtre Sélection de la validation.
4. Dans la fenêtre Sélection de la validation, vous pouvez choisir de valider vos documents XML avec une DTD.
   - Pour effectuer la validation :
     a. Cliquez sur **Validation des documents XML avec la DTD**.
     b. Dans le menu **ID DTD**, sélectionnez la DTD à utiliser pour la validation.

     Si vous n'avez importé aucune DTD dans le référentiel des DTD de votre base de données, vous ne pouvez pas valider vos documents XML.
   - Cliquez sur **Pas de validation des documents XML avec la DTD** pour poursuivre l'opération sans valider vos documents XML.
5. Cliquez sur **Suivant** pour ouvrir la fenêtre Spécification de texte.
6. Dans la zone **Prologue**, tapez le nom de prologue du document XML à composer.

   ```
   <?xml version="1.0"?>
   ```

   Si vous éditez une DAD existante, le prologue s'affiche automatiquement dans la zone **Prologue**.
7. Dans la zone **Doctype** de la fenêtre Spécification de texte, tapez le type du document XML en désignant la DTD associée. Par exemple :

   ```
   !DOCTYPE DAD SYSTEM "c:\dxx\samples\dtd\getstart.dtd"
   ```

Si vous éditez une DAD existante, le type de document s'affiche automatiquement dans la zone **Doctype**.

8. Cliquez sur **Suivant** pour ouvrir la fenêtre Spécification d'une instruction SQL.
9. Tapez une instruction SQL SELECT valide dans la zone **Instruction SQL**. Par exemple :

```
SELECT o.order_key, customer_name, customer_email, p.part_key, color, quantity,
  price, tax, ship_id, date, mode from order_tab o, part_tab p,
table (select substr(char(timestamp(generate_unique())),16)
  as ship_id, date, mode, part_key from ship_tab) s
      WHERE o.order_key = 1 and
                p.price > 20000 and
           p.order_key = o.order_key and
           s.part_key = p.part_key
      ORDER BY order_key, part_key, ship_id
```

Si vous éditez une DAD existante, l'instruction SQL s'affiche automatiquement dans la zone **Instruction SQL**.

10. Cliquez sur **Test SQL** pour vérifier la validité de l'instruction SQL.
    - Dans l'affirmative, l'exemple de résultats s'affiche dans la zone **Exemple de résultats**.
    - Dans la négative, cette zone contient un message d'erreur qui vous demande de corriger l'instruction SQL SELECT et de réessayer.
11. Cliquez sur **Suivant** pour ouvrir la fenêtre Mappage SQL.
12. Pour sélectionner un noeud d'élément ou d'attribut de départ pour le mappage, cliquez sur celui-ci dans la zone située à gauche de la fenêtre Mappage SQL.

    Mappez les éléments et les attributs du document XML vers les noeuds d'éléments et d'attributs correspondant aux données DB2. Ces noeuds indiquent le chemin des données XML aux données DB2.

    - **Pour ajouter le noeud racine (root) :**
      a. Sélectionnez l'icône **Root**.
      b. Cliquez sur **Nouvel élément** pour définir un nouveau noeud.
      c. Dans la boîte d'options **Détails**, indiquez **Elément** pour **Type du noeud**.
      d. Entrez le nom du noeud de niveau supérieur dans la zone **Nom du noeud**.
      e. Cliquez sur **Ajout** pour créer le noeud.

         Vous avez créé le noeud ou l'élément racine, parent de tous les autres noeuds d'éléments et d'attributs de la mappe. Vous pouvez à présent lui ajouter des éléments enfants et des attributs.

- **Pour ajouter un élément enfant ou un noeud d'attribut :**
  a. Pour ajouter un élément ou un attribut enfant, cliquez sur un noeud parent dans la zone située à gauche.

     Si vous n'avez sélectionné aucun noeud parent, l'option **Nouvel Elément** n'est pas disponible.
  b. Cliquez sur **Nouvel élément**.
  c. Sélectionnez un type de noeud dans la liste **Type du noeud** de la boîte d'options **Détails**.

     Le menu **Type du noeud** n'affiche que les types de noeuds valides pour ce point de la mappe :

     Elément (element)
     Représente un élément XML défini dans la DTD du document XML. Permet d'associer l'élément XML à une colonne dans une table DB2. Un noeud d'élément peut être doté de noeuds d'attributs, de noeuds d'éléments enfants ou de noeuds de texte. Un noeud de niveau inférieur est associé à un noeud de texte et à un nom de colonne dans l'arborescence.

     Attribut (attribute)
     Représente un attribut XML défini dans la DTD du document XML. Permet d'associer l'attribut XML à une colonne dans une table DB2. Un noeud d'attribut peut être doté d'un noeud de texte. Il est associé à un nom de colonne dans l'arborescence.

     Texte (text)
     Indique le contenu textuel d'un noeud d'élément ou d'attribut à mapper vers une table relationnelle. Un noeud de texte est associé à un nom de colonne dans l'arborescence.

     Table Indique le nom de table d'une valeur d'élément ou d'attribut à mapper vers une table relationnelle.

     Column
     Indique le nom de colonne d'une valeur d'élément ou d'attribut à mapper vers une table relationnelle.

     Condition
     Indique une condition pour la colonne.
  d. Renseignez la zone **Nom du noeud** dans la boîte d'options **Détails**. Par exemple :

     ```
     Order
     ```

e. Si vous avez indiqué le type de noeud **Attribut**, **Elément** ou **Texte** pour un élément de niveau inférieur, sélectionnez une colonne dans la zone **Colonne** de la boîte d'options **Détails**. Par exemple :

```
Customer_Name
```

Restriction : Vous ne pouvez pas créer de colonne à l'aide de l'assistant d'administration. Si vous indiquez le type de noeud **Colonne**, vous ne pouvez sélectionner qu'une colonne existant déjà dans votre base de données DB2.

f. Cliquez sur **Ajout** pour ajouter le noeud.

Vous pouvez modifier un noeud ultérieurement. Pour ce faire, cliquez sur celui-ci dans la zone située à gauche et apportez toutes les modifications souhaitées dans la boîte d'options **Détails**. Cliquez sur **Modification** pour mettre à jour l'élément.

Vous pouvez également ajouter des éléments enfants ou des attributs au noeud. Pour ce faire, mettez-le en évidence et répétez la procédure d'ajout.

g. Poursuivez l'édition de la mappe SQL, ou cliquez sur **Suivant** pour ouvrir la fenêtre Spécification d'une DAD.

- **Pour retirer un noeud :**
  a. Cliquez sur un noeud figurant dans la zone située à gauche.
  b. Cliquez sur **Retrait**.
  c. Poursuivez l'édition de la mappe SQL, ou cliquez sur **Suivant** pour ouvrir la fenêtre Spécification d'une DAD.

Si vous avez retiré un noeud de niveau inférieur, un autre élément, pouvant nécessiter la définition d'un nom de colonne, le remplace dans l'arborescence.

13. Dans la zone **Nom de fichier** de la fenêtre Spécification d'une DAD, tapez un nom de fichier de sortie pour la DAD modifiée.
14. Cliquez sur **Fin** pour revenir au tableau de bord.

A partir du shell de commandes DB2: Choisissez le mappage SQL lorsque vous composez des documents XML et que vous voulez utiliser SQL.

Le fichier DAD est un fichier XML qui peut être créé avec n'importe quel éditeur de texte. Les étapes suivantes utilisent des exemples extraits de l'annexe «Fichiers DAD» à la page 278. Pour obtenir de plus amples informations et prendre connaissance du contexte, reportez-vous à ces exemples.

1. Ouvrez un éditeur de texte.
2. Créez l'en-tête de la DAD :

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "path\dad.dtd"> --> Chemin et nom de fichier de
                                        la DTD pour le fichier DAD
```

3. Insérez les balises <DAD></DAD>.
4. Après la balise <DAD>, indiquez l'ID DTD qui associe le fichier DAD à la DTD du document XML.

   ```
   <dtdid>path\dtd_name.dtd> --> Chemin et nom de fichier de la DTD
   pour l'application
   ```

5. Indiquez si vous voulez une validation (avec la DTD pour vérifier que le document XML est valide). Par exemple :

   ```
   <validation>NO</validation>  --> Indiquez YES ou NO
   ```

6. A l'aide de l'élément <Xcollection>, définissez la méthode d'accès et de stockage collection XML. Les méthodes d'accès et de stockage indiquent que le contenu du document XML provient de données XML stockées dans des tables DB2.

   ```
   <Xcollection>
   </Xcollection>
   ```

7. Indiquez une ou plusieurs instructions SQL pour effectuer des opérations d'interrogation ou d'insertion de données sur les tables DB2. Pour consulter les instructions correspondantes, reportez-vous à la section «Exigences liées aux schémas de mappage» à la page 65. Par exemple, lancez une seule requête SQL comme suit :

   ```
    <SQL_stmt>
      SELECT o.order_key, customer_name, customer_email, p.part_key, color,
      quantity, price, tax, ship_id, date, mode from order_tab o, part_tab p,
   table (select substr(char(timestamp(generate_unique())),16)
     as ship_id, date, mode, part_key from ship_tab) s
         WHERE o.order_key = 1 and
                   p.price > 20000 and
                p.order_key = o.order_key and
                s.part_key = p.part_key
         ORDER BY order_key, part_key, ship_id
    </SQL_stmt>
   ```

8. Ajoutez les informations de prologue suivantes :

   ```
   <prolog>?xml version="1.0"?</prolog>
   ```

   Ce texte est obligatoire.

9. Ajoutez les balises <doctype></doctype>. Par exemple :

   ```
   <doctype>! DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
   ```

10. Définissez le noeud racine à l'aide des balises <root_node></root_node>. Dans root_node, indiquez les éléments et les attributs qui composent le document XML.
11. Mappez les éléments et les attributs du document XML vers les noeuds d'éléments et d'attributs correspondant aux données DB2. Ces noeuds indiquent le chemin des données XML aux données DB2.

a. Définissez une valeur <element_node> pour chaque élément du document XML mappant vers une colonne de table DB2.

```
<element_node name="name"></element_node>
```

Un noeud d'élément peut être doté des noeuds suivants :

- noeud d'attribut (attribute_node),
- noeud d'élément enfant (child element_node),
- noeud de texte (text_node).

b. Définissez une valeur <attribute_node> pour chaque attribut du document XML mappant vers une colonne de table DB2. Pour obtenir des exemples de DTD applicables au mappage SQL, reportez-vous au début de cette section. Pour consulter la DTD du fichier DAD et la syntaxe complète de ce dernier, reportez-vous à l'«Annexe A. DTD de fichier DAD» à la page 269.

Par exemple, vous avez besoin d'un attribut key pour l'élément <Order>. La valeur de key est stockée dans la colonne PART_KEY.

Fichier DAD : Dans le fichier DAD, créez un noeud d'attribut pour key et indiquez la table dans laquelle la valeur 1 doit être stockée.

```
<attribute_node name="key">
    <column name="part_key"/>
</attribute_node>
```

Document XML composé : La valeur de key est extraite de la colonne PART_KEY.

```
<Order key="1">
```

12. Créez un noeud de texte (<text_node>) pour chaque élément ou attribut dont le contenu sera dérivé d'une table DB2. Le noeud de texte comporte un élément <column> qui identifie la colonne source du contenu.

Par exemple, l'élément XML <Tax> peut être associé à une valeur qui doit être extraite de la colonne TAX :

Elément DAD :

```
<element_node name="Tax">
   <text_node>
         <column name="tax"/>
   </text_node>
</element_node>
```

Dans l'instruction SQL, le nom de colonne doit figurer au début du fichier DAD.

Document XML composé :

```
<Tax>0.02</Tax>
```

La valeur 0,02 sera dérivée de la colonne TAX.

13. Vérifiez que vous avez placé une balise de fin </root_node> après la dernière balise </element_node>.
14. Vérifiez que vous avez placé une balise de fin </Xcollection> après la dernière balise </root_node>.
15. Vérifiez que vous avez placé une balise de fin </DAD> après la balise </Xcollection>.

Composition de documents XML en mode mappage du noeud RDB

Choisissez le mappage de noeud RDB pour composer des documents XML à l'aide d'une structure compatible XML.

Cette méthode fait appel à <RDB_node> pour indiquer les tables DB2, la colonne et les conditions relatives à un noeud d'élément ou d'attribut. <RDB_node> utilise les éléments suivants :

- <table> : définit la table correspondant à l'élément.
- <column> : définit la colonne qui contient l'élément correspondant.
- <condition> : facultatif, indique une condition sur la colonne.

Les éléments enfants utilisés dans <RDB_node> varient en fonction du contexte du noeud et suivent les règles ci-après :

Type de noeud :	**Elément enfant RDB**		
	Table	**Colonne**	**Condition[1]**
Elément racine	O	N	O
Attribut	O	O	Facultatif
Texte	O	O	Facultatif

(1) Obligatoire avec plusieurs tables

A l'aide de l'assistant d'administration: Pour créer une DAD de composition en mode mappage du noeud RDB :

1. Configurez et démarrez l'assistant d'administration. Pour plus d'informations, reportez-vous à la section «Démarrage de l'assistant d'administration» à la page 71.
2. A partir du tableau de bord, cliquez sur **Utilisation des fichiers DAD**. La fenêtre Spécification d'une DAD s'affiche.
3. Vous pouvez éditer un fichier DAD existant ou en créer un nouveau.

   Pour éditer une DAD existante :

   a. Tapez le nom du fichier DAD dans la zone **Nom de fichier**, ou cliquez sur **...** pour parcourir la liste des DAD disponibles.
   b. Vérifiez que l'assistant reconnaît le fichier DAD indiqué.

- Dans l'affirmative, le bouton **Suivant** devient disponible et Mappage du noeud RDB d'une collection XML s'affiche dans la zone **Type**.
- Dans la négative, vous ne pouvez pas cliquer sur **Suivant**. Tapez un autre nom de fichier DAD existant dans la zone **Nom de fichier**, ou cliquez sur **...** pour parcourir à nouveau la liste des DAD disponibles. Réessayez jusqu'à ce que le bouton **Suivant** devienne disponible.

c. Cliquez sur **Suivant** pour ouvrir la fenêtre Sélection de la validation.

Pour créer un fichier DAD :

a. Laissez la zone **Nom de fichier** à blanc.

b. Sélectionnez Mappage du noeud RDB d'une collection XM dans le menu **Type**.

c. Cliquez sur **Suivant** pour ouvrir la fenêtre Sélection de la validation.

4. Dans la fenêtre Sélection de la validation, vous pouvez choisir de valider vos documents XML avec une DTD.
   - Pour effectuer la validation :
     a. Cliquez sur **Validation des documents XML avec la DTD**.
     b. Dans le menu **ID DTD**, sélectionnez la DTD à utiliser pour la validation.

     Si vous n'avez importé aucune DTD dans le référentiel des DTD de votre base de données, vous ne pouvez pas valider vos documents XML.
   - Cliquez sur **Pas de validation des documents XML avec la DTD** pour poursuivre l'opération sans valider vos documents XML.
5. Cliquez sur **Suivant** pour ouvrir la fenêtre Spécification de texte.
6. Tapez le nom de prologue dans la zone **Prologue** de la fenêtre Spécification de texte.

   ```
   <?xml version="1.0"?>
   ```

   Si vous éditez une DAD existante, le prologue s'affiche automatiquement dans la zone **Prologue**.
7. Dans la zone **Doctype** de la fenêtre Spécification de texte, entrez le type du document XML.

   Si vous éditez une DAD existante, le type de document s'affiche automatiquement dans la zone **Doctype**.
8. Cliquez sur **Suivant** pour ouvrir la fenêtre Mappage RDB.
9. Pour sélectionner un noeud d'élément ou d'attribut de départ pour le mappage, cliquez sur celui-ci dans la zone située à gauche de la fenêtre Mappage RDB.

Mappez les éléments et les attributs du document XML vers les noeuds d'éléments et d'attributs correspondant aux données DB2. Ces noeuds indiquent le chemin des données XML aux données DB2.

10. **Pour ajouter le noeud racine (root) :**
    a. Sélectionnez l'icône **Root**.
    b. Cliquez sur **Nouvel élément** pour définir un nouveau noeud.
    c. Dans la boîte d'options **Détails**, indiquez **Elément** pour **Type du noeud**.
    d. Entrez le nom du noeud de niveau supérieur dans la zone **Nom du noeud**.
    e. Cliquez sur **Ajout** pour créer le noeud.

       Vous avez créé le noeud ou l'élément racine, parent de tous les autres noeuds d'éléments et d'attributs de la mappe. Le noeud racine comporte des éléments enfants de table et une condition de jointure.
    f. Ajoutez des noeuds de tables pour chaque table faisant partie de la collection.
       1) Mettez le nom de noeud racine en évidence et sélectionnez **Nouvel élément**.
       2) Dans la boîte d'options **Détails**, indiquez **Table** pour **Type du noeud**.
       3) Sélectionnez le nom de la table dans la liste **Nom de la table**. La table doit déjà exister.
       4) Cliquez sur **Ajout** pour ajouter le noeud de table.
       5) Répétez ces étapes pour chaque table.
    g. Ajoutez une condition de jointure pour les noeuds de tables.
       1) Mettez le nom de noeud racine en évidence et sélectionnez **Nouvel élément**.
       2) Dans la boîte d'options **Détails**, indiquez **Condition** pour **Type du noeud**.
       3) Dans la zone **Condition**, entrez la condition de jointure en respectant la syntaxe suivante :

          ```
          table_name.table_column = table_name.table_column AND
          table_name.table_column = table_name.table_column ...
          ```
       4) Cliquez sur **Ajout** pour ajouter la condition.
11. **Pour ajouter un noeud d'élément ou d'attribut :**
    a. Pour ajouter un élément ou un attribut enfant, cliquez sur un noeud parent dans la zone située à gauche.
    b. Cliquez sur **Nouvel élément**. Si vous n'avez sélectionné aucun noeud parent, l'option **Nouvel Elément** n'est pas disponible.

c. Sélectionnez un type de noeud dans le menu **Type du noeud** de la boîte d'options **Détails**.

   Le menu **Type du noeud** n'affiche que les types de noeuds valides pour ce point de la mappe. **Elément** ou **attribut**.

d. Renseignez la zone **Nom du noeud**.

e. Cliquez sur **Ajout** pour ajouter le noeud.

f. **Pour mapper le contenu d'un noeud d'élément ou d'attribut vers une table relationnelle :**

   1) Indiquez un noeud de texte.
      a) Cliquez sur le noeud parent.
      b) Cliquez sur **Nouvel élément**.
      c) Dans la zone **Type du noeud**, sélectionnez **Texte**.
      d) Sélectionnez **Ajout** pour ajouter le noeud.
   2) Ajoutez un noeud de table.
      a) Sélectionnez le noeud de texte que vous venez de créer et cliquez sur **Nouvel élément**.
      b) Dans la zone **Type du noeud**, sélectionnez **Table** et indiquez un nom de table pour l'élément.
      c) Cliquez sur **Ajout** pour ajouter le noeud.
   3) Ajoutez un noeud de colonne.
      a) Resélectionnez le noeud de texte et cliquez sur **Nouvel élément**.
      b) Dans la zone **Type du noeud**, sélectionnez **Colonne** et indiquez un nom de colonne pour l'élément.
      c) Cliquez sur **Ajout** pour ajouter le noeud.

      Restriction : Vous ne pouvez pas créer de colonne à l'aide de l'assistant d'administration. Si vous indiquez le type de noeud Colonne, vous ne pouvez sélectionner qu'une colonne existant déjà dans votre base de données DB2.
   4) Vous pouvez ajouter une condition pour la colonne.
      a) Resélectionnez le noeud de texte et cliquez sur **Nouvel élément**.
      b) Dans la zone **Type du noeud**, sélectionnez **Condition** et indiquez une condition en respectant la syntaxe :

         operator LIKE|<|>|= *value*
      c) Cliquez sur **Ajout** pour ajouter le noeud.

g. Poursuivez l'édition de la mappe RDB, ou cliquez sur **Suivant** pour ouvrir la fenêtre Spécification d'une DAD.

12. **Pour retirer un noeud :**

a. Cliquez sur un noeud figurant dans la zone située à gauche.
b. Cliquez sur **Retrait**.
c. Poursuivez l'édition de la mappe du noeud RDB, ou cliquez sur **Suivant** pour ouvrir la fenêtre Spécification d'une DAD.

13. Dans la zone **Nom de fichier** de la fenêtre Spécification d'une DAD, tapez un nom de fichier de sortie pour la DAD modifiée.
14. Cliquez sur **Fin** pour retirer le noeud et revenir au tableau de bord.

A partir du shell de commandes DB2: Le fichier DAD est un fichier XML qui peut être créé avec n'importe quel éditeur de texte. Les étapes suivantes utilisent des exemples extraits de l'annexe «Fichiers DAD» à la page 278. Pour obtenir de plus amples informations et prendre connaissance du contexte, reportez-vous à ces exemples.

1. Ouvrez un éditeur de texte.
2. Créez l'en-tête de la DAD :

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "path\dad.dtd"> --> Chemin et nom de fichier de la DTD
pour le fichier DAD
```

3. Insérez les balises <DAD></DAD>.
4. Après la balise <DAD>, indiquez l'ID DTD qui associe le fichier DAD à la DTD du document XML.

```
<dtdid>path\dtd_name.dtd> --> Chemin et nom de fichier de la DTD
                                    pour l'application
```

5. Indiquez si vous voulez une validation (avec la DTD pour vérifier que le document XML est valide). Par exemple :

```
<validation>NO</validation>  --> Indiquez YES ou NO
```

6. A l'aide de l'élément <Xcollection>, définissez la méthode d'accès et de stockage collection XML. Les méthodes d'accès et de stockage indiquent que les données XML sont stockées dans une collection de tables DB2.

```
<Xcollection>
</Xcollection>
```

7. Ajoutez les informations de prologue suivantes :

```
<prolog>?xml version="1.0"?</prolog>
```

   Ce texte est obligatoire.

8. Ajoutez les balises <doctype></doctype>. Par exemple :

```
<doctype>! DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
```

9. Définissez le noeud racine à l'aide des balises <root_node>. Dans root_node, indiquez les éléments et les attributs qui composent le document XML.

10. Mappez les éléments et les attributs du document XML vers les noeuds d'éléments et d'attributs correspondant aux données DB2. Ces noeuds indiquent le chemin des données XML aux données DB2.
    a. Définissez un noeud d'élément racine (node element_node). Ce noeud d'élément contient :
        - un noeud RDB désignant des noeuds de tables associés à une condition de jointure pour indiquer la collection,
        - des éléments enfants,
        - des attributs.

        Pour indiquer les noeuds de tables et les conditions :

        1) Créez un élément de noeud RDB. Par exemple :

```
      <RDB_node>
</RDB_node>
```

        2) Définissez un noeud de table <table_node> pour chaque table contenant des données à inclure dans le document XML. Par exemple, soient trois tables, ORDER_TAB, PART_TAB et SHIP_TAB contenant des données de colonne à inclure dans le document. Créez un noeud de table pour chacune d'elles. Par exemple :

```
      <RDB_node>
<table name="ORDER_TAB">
<table name="PART_TAB">
<table name="SHIP_TAB"></RDB_node>
```

        3) Vous pouvez également indiquer une colonne de clé pour chaque table lorsque vous préparez l'activation de la collection. L'attribut key n'est normalement pas obligatoire pour la composition. Cependant, lorsque vous activez une collection, le fichier DAD utilisé doit supporter les opérations de composition et de décomposition. Par exemple :

```
      <RDB_node>
<table name="ORDER_TAB" key="order_key">
<table name="PART_TAB" key="part_key">
<table name="SHIP_TAB" key="date mode">
</RDB_node>
```

        4) Définissez une condition de jointure pour les tables de la collection. La syntaxe est la suivante :

```
expression = expression AND
expression = expression
```

        Par exemple :

```
      <RDB_node>
<table name="ORDER_TAB">
<table name="PART_TAB">
<table name="SHIP_TAB">
<condition>
```

```
                    order_tab.order_key = part_tab.order_key AND
                       part_tab.part_key = ship_tab.part_key
    </condition>
    </RDB_node>
```

b. Définissez une balise <element_node> pour chaque élément du document XML mappant vers une colonne de table DB2. Par exemple :

```
<element_node name="name">
</element_node>
```

Un noeud d'élément peut être associé à l'un des types d'éléments suivants :

- <text_node> : pour indiquer que le contenu de l'élément mappe vers une table dB2. Cet élément ne comporte aucun élément enfant.
- <attribute_node> : pour indiquer un attribut. Les noeuds d'attributs sont définis à l'étape suivante.

Le noeud de texte (text_node) comporte un noeud RDB (<RDB_node>) pour mapper le contenu de l'élément vers une table et une colonne DB2.

Les noeuds RDB sont utilisés pour des éléments de niveau inférieur dont le contenu doit être mappé vers une table DB2. Un noeud RDB comporte les éléments enfants suivants :

- <table> : définit la table correspondant à l'élément.
- <column> : définit la colonne qui contient l'élément correspondant et indique le type de colonne à l'aide de l'attribut type.
- <condition> : facultatif, indique une condition sur la colonne.

Par exemple, un élément XML <Tax> peut mapper vers une colonne TAX :

Document XML :

```
<Tax>0.02</Tax>
```

Dans ce cas, vous voulez stocker la valeur 0,02 dans la colonne TAX.

```
<element_node name="Tax">
   <text_node>
      <RDB_node>
         <table name="part_tab"/>
         <column name="tax"/>
      </RDB_node>
   </text_node>
</element_node>
```

Dans cet exemple, <RDB_node> indique que la valeur de l'élément <Tax> est une valeur texte et que les données sont stockées dans la colonne TAX de la table PART_TAB. Pour consulter des exemples de fichiers DAD avec mappage du noeud RDB, reportez-vous à la section «Fichiers DAD» à la page 278. Pour consulter la DTD du fichier DAD et la syntaxe complète de ce dernier, reportez-vous à l'«Annexe A. DTD de fichier DAD» à la page 269.

c. Vous pouvez également ajouter un attribut type à chaque élément <column> lorsque vous préparez l'activation de la collection. L'attribut type n'est normalement pas obligatoire pour la composition. Cependant, lorsque vous activez une collection, le fichier DAD utilisé doit supporter les opérations de composition et de décomposition. Par exemple :

```
<column name="tax" type="real"/>
```

d. Définissez une valeur <attribute_node> pour chaque attribut du document XML mappant vers une colonne de table DB2. Par exemple :

```
<attribute_node name="key">
</attribute_node>
```

Le noeud d'attribut (attribute_node) comporte un noeud RDB (<RDB_node>) pour mapper la valeur d'attribut vers une table et une colonne DB2. Un noeud RDB comporte les éléments enfants suivants :

- <table> : définit la table correspondant à l'élément.
- <column> : définit la colonne qui contient l'élément correspondant.
- <condition> : facultatif, indique une condition sur la colonne.

Par exemple, l'élément <Order> peut être associé à un attribut key. La valeur de key doit être stockée dans la colonne PART_KEY. Dans le fichier DAD, créez un noeud d'attribut (<attribute_node>) pour key et indiquez la table dans laquelle la valeur 1 doit être stockée.

Fichier DAD

```
<attribute_node name="key">
      <RDB_node>
    <table name="part_tab">
    <column name="part_key"/>
      <RDB_node>
</attribute_node>
```

Document XML composé :

```
    <Order key="1">
```

11. Vérifiez que vous avez placé une balise de fin </root_node> après la dernière balise </element_node>.

12. Vérifiez que vous avez placé une balise de fin </Xcollection> après la dernière balise </root_node>.
13. Vérifiez que vous avez placé une balise de fin </DAD> après la balise </Xcollection>.

Spécification d'une feuille de style pour le document XML

Lors de la composition de documents, l'Extension XML prend également en charge les instructions de traitement des feuilles de style, via l'élément <stylesheet>. Ces instructions de traitement doivent se trouver dans l'élément racine <Xcollection>, situé à côté des instructions <doctype> et <prolog> définies pour la structure du document XML. Par exemple :

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dtd\dad.dtd">
<DAD>
<SQL_stmt>
       ...
</SQL_stmt>
<Xcollection>
...
<prolog>...</prolog>
<doctype>...</doctype>
<stylesheet>?xml-stylesheet type="text/css" href="order.css"?</stylesheet>
<root_node>...</root_node>
...

</Xcollection>
...
</DAD>
```

Décomposition de documents XML en mode mappage du noeud RDB

Choisissez le mappage du noeud RDB pour décomposer des documents XML. Cette méthode fait appel à <RDB_node> pour indiquer les tables DB2, la colonne et les conditions relatives à un noeud d'élément ou d'attribut. <RDB_node> utilise les éléments suivants :

- <table> : définit la table correspondant à l'élément.
- <column> : définit la colonne qui contient l'élément correspondant.
- <condition> : facultatif, indique une condition sur la colonne.

Les éléments enfants utilisés dans <RDB_node> varient en fonction du contexte du noeud et suivent les règles ci-après :

Type de noeud :	**Elément enfant RDB**		
	Table	**Colonne**	**Condition**[1]
Elément racine	O	N	O
Attribut	O	O	Facultatif

Type de noeud :	Elément enfant RDB		
	Table	**Colonne**	**Condition**[1]
Texte	O	O	Facultatif
(1) Obligatoire avec plusieurs tables			

A l'aide de l'assistant d'administration: Pour créer une DAD de décomposition en mode mappage du noeud RDB, procédez comme suit :

1. Configurez et démarrez l'assistant d'administration. Pour plus d'informations, reportez-vous à la section «Démarrage de l'assistant d'administration» à la page 71.
2. A partir du tableau de bord, cliquez sur **Utilisation des fichiers DAD**. La fenêtre Spécification d'une DAD s'affiche.
3. Vous pouvez éditer un fichier DAD existant ou en créer un nouveau.

   Pour éditer une DAD existante :

   a. Tapez le nom du fichier DAD dans la zone **Nom de fichier**, ou cliquez sur **...** pour parcourir la liste des DAD disponibles.

   b. Vérifiez que l'assistant reconnaît le fichier DAD indiqué.

      - Dans l'affirmative, le bouton **Suivant** devient disponible et Mappage du noeud RDB d'une collection XML s'affiche dans la zone **Type**.
      - Dans la négative, vous ne pouvez pas cliquer sur **Suivant**. Tapez un autre nom de fichier DAD existant dans la zone **Nom de fichier**, ou cliquez sur **...** pour parcourir à nouveau la liste des DAD disponibles. Réessayez jusqu'à ce que le bouton **Suivant** devienne disponible.

   c. Cliquez sur **Suivant** pour ouvrir la fenêtre Sélection de la validation.

   Pour créer un fichier DAD :

   a. Laissez la zone **Nom de fichier** à blanc.

   b. Sélectionnez Mappage du noeud RDB d'une collection XM dans le menu **Type**.

   c. Cliquez sur **Suivant** pour ouvrir la fenêtre Sélection de la validation.
4. Dans la fenêtre Sélection de la validation, vous pouvez choisir de valider vos documents XML avec une DTD.
   - Pour effectuer la validation :

     a. Cliquez sur **Validation des documents XML avec la DTD**.

     b. Dans le menu **ID DTD**, sélectionnez la DTD à utiliser pour la validation.

Si vous n'avez importé aucune DTD dans le référentiel des DTD de votre base de données, vous ne pouvez pas valider vos documents XML.

- Cliquez sur **Pas de validation des documents XML avec la DTD** pour poursuivre l'opération sans valider vos documents XML.

5. Cliquez sur **Suivant** pour ouvrir la fenêtre Spécification de texte.
6. Si vous décomposez uniquement un document XML, ignorez la zone **Prologue**. Si vous utilisez le fichier DAD pour des opérations de composition et de décomposition, tapez le nom de prologue dans la zone **Prologue** de la fenêtre Spécification de texte. Le prologue est facultatif lorsque vous décomposez des documents XML en données DB2.

   ```
   <?xml version="1.0"?>
   ```

   Si vous éditez une DAD existante, le prologue s'affiche automatiquement dans la zone **Prologue**.
7. Si vous décomposez uniquement un document XML, ignorez la zone **Doctype**. Si vous utilisez le fichier DAD pour des opérations de composition et de décomposition, entrez le type de document XML dans la zone **Doctype**.

   Si vous éditez une DAD existante, le type de document s'affiche automatiquement dans la zone **Doctype**.
8. Cliquez sur **Suivant** pour ouvrir la fenêtre Mappage RDB.
9. Pour sélectionner un noeud d'élément ou d'attribut de départ pour le mappage, cliquez sur celui-ci dans la zone située à gauche de la fenêtre Mappage RDB.

   Mappez les éléments et les attributs du document XML vers les noeuds d'éléments et d'attributs correspondant aux données DB2. Ces noeuds indiquent le chemin des données XML aux données DB2.
10. **Pour ajouter le noeud racine (root) :**
    a. Sélectionnez l'icône **Root**.
    b. Cliquez sur **Nouvel élément** pour définir un nouveau noeud.
    c. Dans la boîte d'options **Détails**, indiquez **Elément** pour **Type du noeud**.
    d. Entrez le nom du noeud de niveau supérieur dans la zone **Nom du noeud**.
    e. Cliquez sur **Ajout** pour créer le noeud.

       Vous avez créé le noeud ou l'élément racine, parent de tous les autres noeuds d'éléments et d'attributs de la mappe. Le noeud racine comporte des éléments enfants de table et une condition de jointure.
    f. Ajoutez des noeuds de tables pour chaque table faisant partie de la collection.

1) Mettez le nom de noeud racine en évidence et sélectionnez **Nouvel élément**.
2) Dans la boîte d'options **Détails**, indiquez **Table** pour **Type du noeud**.
3) Sélectionnez le nom de la table dans la liste **Nom de la table**. La table doit déjà exister.
4) Dans la zone **Clé de la table**, indiquez une colonne de clé pour la table.
5) Cliquez sur **Ajout** pour ajouter le noeud de table.
6) Répétez ces étapes pour chaque table.

g. Ajoutez une condition de jointure pour les noeuds de tables.
1) Mettez le nom de noeud racine en évidence et sélectionnez **Nouvel élément**.
2) Dans la boîte d'options **Détails**, indiquez **Condition** pour **Type du noeud**.
3) Dans la zone **Condition**, entrez la condition de jointure en respectant la syntaxe suivante :

```
table_name.table_column = table_name.table_column AND
table_name.table_column = table_name.table_column ...
```

4) Cliquez sur **Ajout** pour ajouter la condition.

Vous pouvez à présent lui ajouter des éléments enfants et des attributs.

11. **Pour ajouter un noeud d'élément ou d'attribut :**
a. Pour ajouter un élément ou un attribut enfant, cliquez sur un noeud parent dans la zone située à gauche.

Si vous n'avez sélectionné aucun noeud parent, l'option **Nouveau** n'est pas disponible.

b. Cliquez sur **Nouvel élément**.
c. Sélectionnez un type de noeud dans le menu **Type du noeud** de la boîte d'options **Détails**.

Le menu **Type du noeud** n'affiche que les types de noeuds valides pour ce point de la mappe. **Elément** ou **attribut**.

d. Renseignez la zone **Nom du noeud**.
e. Cliquez sur **Ajout** pour ajouter le noeud.
f. **Pour mapper le contenu d'un noeud d'élément ou d'attribut vers une table relationnelle :**
1) Indiquez un noeud de texte.
a) Cliquez sur le noeud parent.
b) Cliquez sur **Nouvel élément**.
c) Dans la zone **Type du noeud**, sélectionnez **Texte**.

d) Sélectionnez **Ajout** pour ajouter le noeud.

2) Ajoutez un noeud de table.
   a) Sélectionnez le noeud de texte que vous venez de créer et cliquez sur **Nouvel élément**.
   b) Dans la zone **Type du noeud**, sélectionnez **Table** et indiquez un nom de table pour l'élément.
   c) Cliquez sur **Ajout** pour ajouter le noeud.
3) Ajoutez un noeud de colonne.
   a) Resélectionnez le noeud de texte et cliquez sur **Nouvel élément**.
   b) Dans la zone **Type du noeud**, sélectionnez **Colonne** et indiquez un nom de colonne pour l'élément.
   c) Dans la zone **Type**, indiquez le type de base à affecter à la colonne pour que celle-ci puisse stocker les données non balisées.
   d) Cliquez sur **Ajout** pour ajouter le noeud.

   Restriction : Vous ne pouvez pas créer de colonne à l'aide de l'assistant d'administration. Si vous indiquez le type de noeud Colonne, vous ne pouvez sélectionner qu'une colonne existant déjà dans votre base de données DB2.
4) Vous pouvez ajouter une condition pour la colonne.
   a) Resélectionnez le noeud de texte et cliquez sur **Nouvel élément**.
   b) Dans la zone **Type du noeud**, sélectionnez **Condition** et indiquez une condition en respectant la syntaxe :

      operator LIKE|<|>|= *value*
   c) Cliquez sur **Ajout** pour ajouter le noeud.

Pour modifier un noeud, sélectionnez-le, éditez les zones dans le groupe d'options **Détails**, puis cliquez sur **Modification**.

g. Poursuivez l'édition de la mappe RDB, ou cliquez sur **Suivant** pour ouvrir la fenêtre Spécification d'une DAD.

12. **Pour retirer un noeud :**
    a. Cliquez sur un noeud figurant dans la zone située à gauche.
    b. Cliquez sur **Retrait**.
    c. Poursuivez l'édition de la mappe du noeud RDB, ou cliquez sur **Suivant** pour ouvrir la fenêtre Spécification d'une DAD.
13. Dans la zone **Nom de fichier** de la fenêtre Spécification d'une DAD, tapez un nom de fichier de sortie pour la DAD modifiée.
14. Cliquez sur **Fin** pour retirer le noeud et revenir au tableau de bord.

A partir du shell de commandes DB2: Le fichier DAD est un fichier XML qui peut être créé avec n'importe quel éditeur de texte. Les étapes suivantes utilisent des exemples extraits de l'annexe «Fichiers DAD» à la page 278. Pour obtenir de plus amples informations et prendre connaissance du contexte, reportez-vous à ces exemples.

1. Ouvrez un éditeur de texte.
2. Créez l'en-tête de la DAD :

   ```
   <?xml version="1.0"?>
   <!DOCTYPE DAD SYSTEM "path\dad.dtd"> --> Chemin et nom de fichier de la DTD
   pour le fichier DAD
   ```

3. Insérez les balises <DAD></DAD>.
4. Après la balise <DAD>, indiquez l'ID DTD qui associe le fichier DAD à la DTD du document XML.

   ```
   <dtdid>path\dtd_name.dtd> --> Chemin et nom de fichier de la DTD
                                 pour l'application
   ```

5. Indiquez si vous voulez une validation (avec la DTD pour vérifier que le document XML est valide). Par exemple :

   ```
   <validation>NO</validation>  --> Indiquez YES ou NO
   ```

6. A l'aide de l'élément <Xcollection>, définissez la méthode d'accès et de stockage collection XML. Les méthodes d'accès et de stockage indiquent que les données XML sont stockées dans une collection de tables DB2.

   ```
   <Xcollection>
   </Xcollection>
   ```

7. Ajoutez les informations de prologue suivantes :

   ```
   <prolog>?xml version="1.0"?</prolog>
   ```

   Ce texte est obligatoire.
8. Ajoutez les balises <doctype></doctype>. Par exemple :

   ```
   <doctype>! DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
   ```

9. Définissez le noeud racine à l'aide des balises <root_node></root_node>. Dans root_node, indiquez les éléments et les attributs qui composent le document XML.
10. Après la balise <root_node>, mappez les éléments et les attributs du document XML vers les noeuds d'éléments et d'attributs correspondant aux données DB2. Ces noeuds indiquent le chemin des données XML aux données DB2.
    a. Définissez un noeud d'élément racine de niveau supérieur. Ce noeud d'élément contient :
       - des noeuds de tables associés à une condition de jointure pour indiquer la collection,
       - des éléments enfants,
       - des attributs.

Pour indiquer les noeuds de tables et les conditions :

1) Créez un élément de noeud RDB. Par exemple :

```
      <RDB_node>
</RDB_node>
```

2) Définissez un noeud de table <table_node> pour chaque table contenant des données à inclure dans le document XML. Par exemple, soient trois tables, ORDER_TAB, PART_TAB et SHIP_TAB contenant des données de colonne à inclure dans le document. Créez un noeud de table pour chacune d'elles. Par exemple :

```
      <RDB_node>
<table name="ORDER_TAB">
<table name="PART_TAB">
<table name="SHIP_TAB"></RDB_node>
```

3) Définissez une condition de jointure pour les tables de la collection. La syntaxe est la suivante :

```
expression = expression AND
expression = expression ...
```

Par exemple :

```
      <RDB_node>
<table name="ORDER_TAB">
<table name="PART_TAB">
<table name="SHIP_TAB">
<condition>
                order_tab.order_key = part_tab.order_key AND
                    part_tab.part_key = ship_tab.part_key
</condition>
</RDB_node>
```

4) Indiquez une clé primaire pour chaque table. La clé primaire correspond à une ou plusieurs colonnes, constituant la clé composée. Pour indiquer la clé primaire, ajoutez une clé d'attribut à l'élément table du noeud RDB. L'exemple suivant définit une clé primaire pour chaque table du noeud RDB, dans l'ordre du noeud d'élément racine :

```
<element_node name="Order">
      <RDB_node>
         <table name="order_tab" key="order_key"/>
         <table name="part_tab" key="part_key price"/>
         <table name="ship_tab" key="date mode"/>
                  <condition>
                order_tab.order_key = part_tab.order_key AND
                    part_tab.part_key = ship_tab.part_key
                  </condition>
      <RDB_node>
```

Les informations de décomposition sont ignorées lors de la composition d'un document XML.

L'attribut key n'est normalement pas obligatoire pour la décomposition. Cependant, lorsque vous activez une collection, le fichier DAD utilisé doit supporter les opérations de composition et de décomposition.

b. Définissez une balise <element_node> pour chaque élément du document XML mappant vers une colonne de table DB2. Par exemple :

```
<element_node name="name">
</element_node>
```

Un noeud d'élément peut être associé à l'un des types d'éléments suivants :

- <text_node> : pour indiquer que le contenu de l'élément mappe vers une table dB2. Dans ce cas, cet élément ne comporte aucun élément enfant.
- <attribute_node> : pour indiquer un attribut. Les noeuds d'attributs sont définis à l'étape suivante.
- éléments enfants

Le noeud de texte (text_node) comporte un noeud RDB (RDB_node) pour mapper le contenu de l'élément vers une table et une colonne DB2.

Les noeuds RDB sont utilisés pour des éléments de niveau inférieur dont le contenu doit être mappé vers une table DB2. Un noeud RDB comporte les éléments enfants suivants :

- <table> : définit la table correspondant à l'élément.
- <column> : définit la colonne qui contient l'élément correspondant.
- <condition> : facultatif, indique une condition sur la colonne.

Par exemple, soit un élément XML <Tax> pour lequel vous voulez stocker le contenu non balisé dans la colonne TAX :

Document XML :

```
<Tax>0.02</Tax>
```

Dans ce cas, vous voulez stocker la valeur `0,02` dans la colonne TAX.

Dans le fichier DAD, vous indiquez un noeud RDB (<RDB_node>) pour mapper l'élément XML vers la table et la colonne DB2.

Fichier DAD :

```
<element_node name="Tax">
   <text_node>
      <RDB_node>
```

```
        <table name="part_tab"/>
        <column name="tax"/>
      </RDB_node>
   </text_node>
</element_node>
```

<RDB_node> indique que la valeur de l'élément <Tax> est une valeur texte et que les données sont stockées dans la colonne TAX de la table PART_TAB.

c. Définissez une valeur <attribute_node> pour chaque attribut du document XML mappant vers une colonne de table DB2. Par exemple :

```
<attribute_node name="key">
</attribute_node>
```

Le noeud d'attribut (attribute_node) comporte un noeud RDB (RDB_node) pour mapper la valeur d'attribut vers une table et une colonne DB2. Un noeud RDB comporte les éléments enfants suivants :

- <table> : définit la table correspondant à l'élément.
- <column> : définit la colonne qui contient l'élément correspondant.
- <condition> : facultatif, indique une condition sur la colonne.

Par exemple, l'élément <Order> peut être associé à un attribut key. La valeur de key doit être stockée dans la colonne PART_KEY.

Document XML :

```
<Order key="1">
```

Dans le fichier DAD, créez un noeud d'attribut pour key et indiquez la table dans laquelle la valeur 1 doit être stockée.

Fichier DAD :

```
<attribute_node name="key">
      <RDB_node>
    <table name="part_tab">
    <column name="part_key"/>
      <RDB_node>
</attribute_node>
```

11. Indiquez le type de colonne associé au noeud RDB pour chaque noeud d'attribut (attribute_node) et noeud de texte (text_node). Ainsi, vous vous assurez que le type de données est correct pour chaque colonne de stockage des données non balisées. Pour indiquer le type de colonne, ajoutez le type d'attribut à l'élément colonne. Dans l'exemple ci-après, vous attribuez la valeur INTEGER au type de colonne :

```
<attribute_node name="key">
      <RDB_node>
                    <table name="order_tab"/>
        <column name="order_key" type="integer"/>
      </RDB_node>
</attribute_node>
```

12. Vérifiez que vous avez placé une balise de fin </root_node> après la dernière balise </element_node>.
13. Vérifiez que vous avez placé une balise de fin </Xcollection> après la dernière balise </root_node>.
14. Vérifiez que vous avez placé une balise de fin </DAD> après la balise </Xcollection>.

Activation de collections XML

L'activation d'une collection XML lance l'analyse du fichier DAD pour identifier les tables et les colonnes liées aux documents XML. Elle déclenche également l'enregistrement des données de contrôle dans la table XML_USAGE. L'activation d'une collection XML est facultative dans les cas suivants :

- Décomposition d'un document XML et stockage des données dans de nouvelles tables DB2.
- Composition d'un document XML à partir de données existant dans plusieurs tables DB2.

Si vous utilisez le même fichier DAD pour la composition et la décomposition, vous pouvez activer la collection pour ces deux tâches.

Pour activer une collection XML, vous pouvez utiliser l'assistant d'administration, la commande **dxxadm** avec l'option enable_collection, ou la procédure mémorisée dxxEnableCollection().

A l'aide de l'assistant d'administration

Pour activer une collection XML :

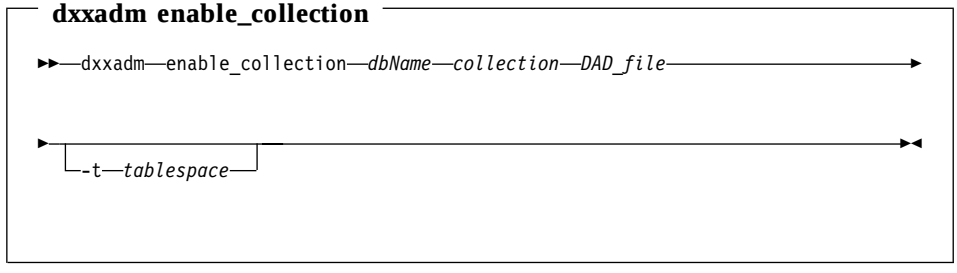
1. Configurez et démarrez l'assistant d'administration. Pour plus d'informations, reportez-vous à la section «Démarrage de l'assistant d'administration» à la page 71.
2. A partir du tableau de bord, cliquez sur **Utilisation des collections XML**. La fenêtre Sélection d'une tâche s'affiche.
3. Cliquez sur **Activation d'une collection** puis sur **Suivant**. La fenêtre Activation d'une collection s'affiche.
4. Dans la zone **Nom de la collection** du menu déroulant, sélectionnez le nom de la collection à activer.
5. Tapez le nom du fichier DAD dans la zone **Nom de fichier DAD**, ou cliquez sur **...** pour parcourir la liste des DAD disponibles.
6. Vous pouvez également indiquer dans la zone **Espace table** le nom d'un espace table existant.

   Cet espace table contiendra les nouvelles tables DB2 générées pour les opérations de décomposition.
7. Cliquez sur **Fin** pour activer la collection et revenir au tableau de bord.
   - Si l'activation de la collection aboutit, un message d'information l'indique.
   - Si l'activation de la collection échoue, un message d'erreur s'affiche. Réessayez jusqu'à l'aboutissement de l'opération.

A partir du shell de commandes DB2

Pour activer une collection XML, entrez la commande **dxxadm** :

Syntaxe :



Paramètres :

dbName
Nom de la base de données.

collection
Nom de la collection XML. Cette valeur est utilisée comme paramètre dans les procédures mémorisées de collection XML.

DAD_file
Nom du fichier contenant la DAD (définition d'accès au document).

tablespace
Espace table existant contenant les nouvelles tables DB2 générées pour la décomposition. Si vous laissez cette zone à blanc, l'Extension XML utilise un espace table par défaut.

Exemple : L'exemple ci-après illustre l'activation de la collection sales_ord dans la base de données SALES_DB, à partir du shell de commandes DB2. Le fichier DAD utilise le mappage SQL et est décrit à la section «Fichier DAD : collection XML (mappage SQL)» à la page 279.

```
dxxadm enable_collection SALES_DB sales_ord getstart.dad
```

Une fois la collection XML activée, vous pouvez composer ou décomposer des documents XML à l'aide des procédures mémorisées de l'Extension XML.

Désactivation de collections XML

Lorsque vous désactivez une collection XML, vous retirez l'enregistrement correspondant de la table XML_USAGE qui identifie les tables et les colonnes de cette collection. Cette opération ne retire aucune table de données. Vous désactivez une collection lorsque vous voulez mettre à jour la DAD et que vous avez besoin de réactiver ou de retirer la collection.

Pour désactiver une collection XML, vous pouvez utiliser l'assistant d'administration, la commande **dxxadm** avec l'option disable_collection, ou la procédure mémorisée dxxDisableCollection().

A l'aide de l'assistant d'administration

Pour désactiver une collection XML :

1. Configurez et démarrez l'assistant d'administration. Pour plus d'informations, reportez-vous à la section «Démarrage de l'assistant d'administration» à la page 71.
2. A partir du tableau de bord, cliquez sur **Utilisation des collections XML** pour afficher les tâches liées aux collections de l'Extension XML. La fenêtre Sélection d'une tâche s'affiche.
3. Cliquez sur **Désactivation d'une collection XML** puis sur **Suivant** pour désactiver une collection XML. La fenêtre Désactivation d'une collection s'affiche.
4. Dans la zone **Nom de la collection**, tapez le nom de la collection à désactiver.
5. Cliquez sur **Fin** pour désactiver la collection et revenir au tableau de bord.

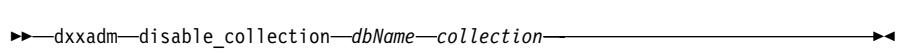
- Si la désactivation de la collection aboutit, un message d'information l'indique.
- Si la désactivation de la collection échoue, un message d'erreur s'affiche. Réessayez jusqu'à l'aboutissement de l'opération.

A partir du shell de commandes DB2

Pour désactiver une collection XML, entrez la commande **dxxadm** :

Syntaxe :

dxxadm disable_collection



Paramètres :

dbName
Nom de la base de données.

collection
Nom de la collection XML. Cette valeur est utilisée comme paramètre dans les procédures mémorisées de collection XML.

Exemple :

```
dxxadm disable_collection SALES_DB sales_ord
```

Désactivation d'une base de données pour XML

Vous désactivez la base de données lorsque vous voulez nettoyer votre environnement DB2 Extension XML et retirer les types UDT, les fonctions UDF, les procédures mémorisées et les tables de gestion DB2 Extension XML.L'LExtension XML désactive la base de données à laquelle vous êtes connecté, en prenant l'instance en cours.

Lorsque vous désactivez une base de données pour XML, DB2 Extension XML exécute les opérations suivantes :

- Suppression de tous les types UDT et de toutes les fonctions UDF.
- Suppression des tables de contrôle contenant les métadonnées pour l'Extension XML.
- Suppression du schéma db2xml.

Avant de commencer

Désactivez les colonnes et les collections XML de la base de données à désactiver.

A l'aide de l'assistant d'administration

Pour désactiver une base de données pour données XML :

1. Configurez et démarrez l'assistant d'administration. Pour plus d'informations, reportez-vous à la section «Démarrage de l'assistant d'administration» à la page 71.
2. A partir du tableau de bord, cliquez sur **Désactivation de la base de données** pour désactiver la base de données en cours.

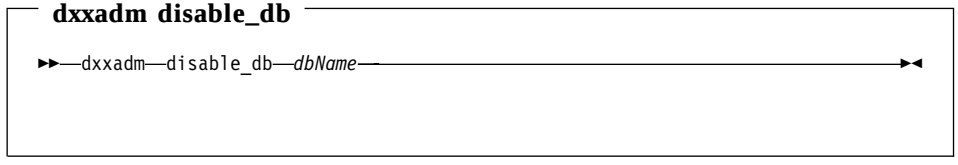
   Si aucune base de données n'est déjà activée, seule l'option **Activation de la base de données** est disponible.

   Une fois la base de données désactivée, vous revenez au tableau de bord.

A partir du shell de commandes DB2

Tapez **dxxadm** sur la ligne de commande en indiquant la base de données à désactiver.

Syntaxe :



Paramètres :

dbName
Nom de la base de données à désactiver.

Exemple : Désactivation d'une base de données existante appelée SALES_DB.

```
dxxadm disable_db SALES_DB
```

Partie 3. Programmation

Cette partie décrit des techniques de programmation pour la gestion des données XML.

Chapitre 5. Gestion des données de colonne XML

Lors de l'utilisation de colonnes XML, vous stockez tout un document XML en tant que données de colonne. Par cette méthode d'accès et d'archivage, le document XML conserve son intégrité, mais reste disponible pour des opérations d'indexation, de recherche, d'extraction ou de mise à jour de données. Une colonne XML contient, en tant que données de colonne, des documents XML dans leur format natif DB2. Lorsque vous avez activé une base de données pour XML, les types UDT (définis par l'utilisateur) suivants sont mis à votre disposition :

XMLCLOB
Contenu de document XML stocké sous forme d'objet CLOB (character large object) dans DB2.

XMLVARCHAR
Contenu de document XML stocké sous forme de caractères VARCHAR dans DB2.

XMLFile
Document XML stocké dans un fichier d'un système de fichiers local.

Vous pouvez créer ou modifier des tables d'application dont les types de données de colonne sont des types UDT XML. Il s'agit des tables XML. Pour apprendre à créer ou à modifier une table XML, reportez-vous à la section «Création ou modification d'une table XML» à la page 82.

Après l'activation d'une colonne XML, vous pouvez gérer son contenu. Après la création d'une colonne XML, vous pouvez effectuer les tâches de gestion suivantes :

- Stockage de documents XML dans DB2
- Extraction de données ou de documents XML dans DB2
- Mise à jour de documents XML
- Suppression de données ou de documents XML

Pour effectuer ces tâches, vous pouvez recourir à deux méthodes :

- Les *fonctions de transtypage par défaut*, qui convertissent le type de base SQL en type UDT XML.
- Les fonctions UDF (définies par l'utilisateur) fournies par l'Extension XML.

Le présent manuel décrit les deux méthodes pour chaque tâche.

Noms des types UDT et des fonctions UDF

Le nom complet d'une fonction DB2 se présente sous la forme *nom-schéma.nom-fonction*, où *nom-schéma* est un identificateur permettant un regroupement logique des objets SQL. Le nom de schéma pour les fonctions UDF de l'Extension XML est db2xml. Ce nom sert également de qualificatif pour les types UDT de l'Extension XML. Le présent manuel ne fait référence qu'au nom de fonction.

Le chemin des fonctions est une liste ordonnée de noms de schémas. DB2 suit leur ordre d'apparition dans la liste pour résoudre les références aux fonctions UDF et aux types UDT. Vous pouvez préciser le chemin des fonctions par l'instruction SQL SET CURRENT FUNCTION PATH. Cette dernière définit le chemin des fonctions dans le registre spécial CURRENT FUNCTION PATH.

Pour l'Extension XML, il est recommandé d'ajouter le schéma db2xml au chemin des fonctions. Cela vous permet d'entrer des noms UDF et UDT Extension XML sans devoir leur attribuer le préfixe `db2xml`. L'exemple ci-après indique comment ajouter le schéma db2xml au chemin des fonctions :

```
SET CURRENT FUNCTION PATH = db2xml, CURRENT FUNCTION PATH
```

Important : Si vous vous connectez sous l'ID utilisateur db2xml, db2xml est automatiquement défini comme premier schéma. Dans ce cas, ne définissez pas `db2xml` comme premier schéma dans le chemin des fonctions. Sinon, une condition d'erreur est générée car votre chemin des fonctions commence par deux schémas db2xml.

Stockage des données

L'Extension XML permet d'insérer dans une colonne XML des documents XML en l'état. Si vous définissez des tables annexes, l'Extension XML les met à jour automatiquement. Lorsque vous stockez directement un document XML, l'Extension XML mémorise son type de base en tant que type XML.

Tâche en cours :

1. Assurez-vous d'avoir créé ou mis à jour le fichier DAD.
2. Déterminez le type de données à utiliser lors du stockage du document.
3. Choisissez une méthode de stockage des données dans la table DB2 (fonctions de transtypage ou fonctions UDF).
4. Indiquez une instruction SQL INSERT qui identifie la table et la colonne XML cible pour le document XML.

L'Extension XML comporte deux méthodes de stockage des documents XML : les fonctions de transtypage par défaut et les fonctions UDF d'archivage. Le tableau 8 à la page 125 indique la pertinence d'utilisation de chaque méthode.

Tableau 8. Fonctions d'archivage de l'Extension XML

Type de base	Type dans DB2		
	XMLVARCHAR	**XMLCLOB**	**XMLFILE**
VARCHAR	XMLVARCHAR()	Non applicable	XMLFileFromVarchar()
CLOB	Non applicable	XMLCLOB()	XMLFileFromCLOB()
FILE	XMLVarcharFromFile()	XMLCLOBFromFile()	XMLFILE

Utilisation d'une fonction de transtypage par défaut

Pour chaque type UDT, il existe une *fonction de transtypage par défaut* permettant de transtyper le type de base SQL en type UDT. Vous pouvez insérer des données par l'ajout dans la clause VALUES de fonctions de transtypage fournies par l'Extension XML. Le tableau 9 répertorie les fonctions de transtypage fournies :

Tableau 9. Fonctions de transtypage par défaut de l'Extension XML

Transtypage utilisé dans la clause SELECT	Type de données renvoyé	Description
XMLVARCHAR(VARCHAR)	XMLVARCHAR	Données d'entrée de la mémoire tampon VARCHAR
XMLCLOB(CLOB)	XMLCLOB	Données d'entrée de la mémoire tampon CLOB ou releveur de coordonnées CLOB
XMLFILE(VARCHAR)	XMLFILE	Nom du fichier de stockage uniquement

Exemple : L'instruction ci-après insère dans le type XMLVARCHAR un type VARCHAR transtypé :

```
INSERT INTO sales_tab
VALUES('123456', 'Sriram Srinivasan', db2xml.XMLVarchar(:xml_buff))
```

Utilisation d'une fonction UDF d'archivage :

Pour chaque type UDT Extension XML, il existe une fonction UDF d'archivage qui permet d'importer des données dans DB2, à partir d'une ressource autre que son type de base. Par exemple, pour importer un document XML en tant qu'objet XMLCLOB dans DB2, vous pouvez utiliser la fonction XMLCLOBFromFile().

Le tableau 10 à la page 126 répertorie les fonctions d'archivage fournies par l'Extension XML.

Tableau 10. Fonctions UDF d'archivage de l'Extension XML

Fonction UDF d'archivage	Type de données renvoyé	Description
XMLVarcharFromFile()	XMLVARCHAR	Lit un document XML à partir d'un fichier de serveur et renvoie une valeur de type XMLVARCHAR.
XMLCLOBFromFile()	XMLCLOB	Lit un document XML à partir d'un fichier de serveur et renvoie une valeur de type XMLCLOB.
XMLFileFromVarchar()	XMLFILE	Lit un document XML dans la mémoire en tant que VARCHAR, l'écrit dans un fichier externe et renvoie une valeur de type XMLFILE correspondant au nom du fichier.
XMLFileFromCLOB()	XMLFILE	Lit un document XML dans la mémoire en tant que CLOB ou releveur de coordonnées CLOB, l'écrit dans un fichier externe et renvoie une valeur de type XMLFILE correspondant au nom du fichier.

Exemple : L'instruction ci-après stocke un enregistrement en tant qu'objet XMLCLOB dans une table XML à l'aide de la fonction XMLCLOBFromFile().

```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
    VALUES( '1234', 'Sriram Srinivasan,
      XMLCLOBFromFile('c:\dxx\samples\cmd\getstart.xml'))
```

Dans l'exemple précédent, l'objet XML est importé du fichier `c:\dxx\samples\cmd\getstart.xml` vers la colonne ORDER de la table SALES_TAB.

Extraction de données

L'Extension XML permet d'extraire tout un document, ou des contenus élémentaires et des valeurs d'attributs. Lors de l'extraction directe d'une colonne XML, l'Extension XML renvoie le type de colonne UDT. Pour plus d'informations sur l'extraction de données, reportez-vous aux sections suivantes :

- «Extraction d'un document entier»
- «Extraction de contenus élémentaires et de valeurs d'attributs» à la page 129

L'Extension XML comporte deux méthodes d'extraction des données : les fonctions de transtypage par défaut et les fonctions UDF Content() multi-référencées. Le tableau 11 indique la pertinence d'utilisation de chaque méthode.

Tableau 11. Fonctions de récupération de l'Extension XML

Type XML	**Type extrait de DB2**		
	VARCHAR	**CLOB**	**FILE**
XMLVARCHAR	VARCHAR	Non applicable	Content()
XMLCLOB	Non applicable	XMLCLOB	Content()
XMLFILE	Non applicable	Content()	FILE

Extraction d'un document entier

Tâche en cours :

1. Assurez-vous que vous avez stocké le document XML dans une table XML et déterminez les données à extraire.
2. Choisissez une méthode pour l'extraction de données et leur importation dans la table DB2 (fonctions de transtypage ou fonctions UDF).
3. Si vous utilisez la fonction UDF Content() multi-référencée, déterminez le type des données à exporter et à importer.
4. Indiquez une instruction SQL qui identifie la table et la colonne XML source pour l'extraction du document XML.

L'Extension XML comporte deux méthodes d'extraction des données :

Utilisation d'une fonction de transtypage par défaut

La fonction de transtypage par défaut fournie par DB2 pour les types UDT permet de convertir un type UDT XML en type de base SQL, puis de l'utiliser. Vous pouvez extraire des données par l'ajout dans l'instruction SELECT de fonctions de transtypage fournies par l'Extension XML. Le tableau 12 répertorie les fonctions de transtypage fournies :

Tableau 12. Fonctions de transtypage par défaut de l'Extension XML

Transtypage utilisé dans la clause SELECT	**Type de données renvoyé**	**Description**
varchar(XMLVARCHAR)	VARCHAR	Document XML dans VARCHAR
clob(XMLCLOB)	CLOB	Document XML dans CLOB

Tableau 12. Fonctions de transtypage par défaut de l'Extension XML (suite)

Transtypage utilisé dans la clause SELECT	Type de données renvoyé	Description
varchar(XMLFile)	VARCHAR	Nom de fichier XML dans VARCHAR

Exemple : Dans l'exemple suivant, les données XMLVARCHAR sont extraites, puis stockées en mémoire en tant que données de type VARCHAR :

```
EXEC SQL SELECT db2xml.varchar(order) from sales_tab
```

Utilisation de la fonction UDF Content() multi-référencée

La fonction UDF Content() permet d'extraire le contenu d'un document d'une mémoire externe vers la mémoire interne, ou de l'exporter de la mémoire interne vers un *fichier externe* figurant sur le serveur DB2.

Par exemple, vous pouvez stocker votre document XML en fichier XMLFILE et l'utiliser dans la mémoire interne. Dans ce cas, vous avez recours à la fonction UDF Content() qui accepte des données d'entrée de type XMLFILE et renvoie un objet CLOB.

La fonction UDF Content() offre deux modes d'extraction, selon le type de données indiqué :

Extraction d'un document de la mémoire externe vers la mémoire interne

La fonction Content() permet d'extraire un document XML stocké en tant que fichier externe et de l'exporter vers une mémoire tampon ou un releveur de coordonnées CLOB. Respectez la syntaxe suivante pour la fonction, *xmlobj* représentant la colonne XML interrogée :

XMLFILE vers CLOB : Extrait des données d'un fichier et les exporte vers un releveur de coordonnées CLOB.

```
Content(xmlobj XMLFile)
```

Extraction d'un document de la mémoire interne vers un fichier externe

La fonction Content() permet également d'extraire un document XML stocké dans DB2 en tant qu'objet XMLCLOB et de l'exporter vers un fichier du système de fichiers résidant sur le serveur de bases de données. Elle renvoie le nom du fichier au format VARCHAR. Respectez la syntaxe suivante pour la fonction, *xmlobj* représentant la colonne XML interrogée et *filename*, le fichier externe. *XML type* peut correspondre au type XMLVARCHAR ou XMLCLOB.

Type XML **vers fichier externe :** Extrait le contenu XML stocké comme type XML et l'exporte vers un fichier externe.

```
Content(xmlobj XML type, filename varchar(512))
```

Où :

xmlobj Nom de la colonne XML à partir de laquelle le contenu XML est extrait ; *xmlobj* peut être de type XMLVARCHAR ou XMLCLOB.

filename
Nom du fichier dans lequel les données XML doivent être stockées.

Dans l'exemple ci-dessous, un petit segment de programme C comportant des instructions SQL *imbriquées* illustre la procédure d'extraction d'un document XML d'un fichier vers la mémoire interne. Cet exemple suppose que la colonne BOOK est de type XMLFILE.

```
EXEC SQL BEGIN DECLARE SECTION;
          SQL TYPE IS CLOB_LOCATOR xml_buff;
     EXEC SQL END DECLARE SECTION;
     EXEC SQL CONNECT TO SALES_DB
     EXEC SQL DECLARE c1 CURSOR FOR
         SELECT Content(order) from sales_tab
                   EXEC SQL OPEN c1;
     do {
        EXEC SQL FETCH c1 INTO :xml_buff;
        if (SQLCODE != 0) {
           break;
           }
        else {
          /* do whatever you need to do with the XML doc in buffer */
          }
        }
     EXEC SQL CLOSE c1;
     EXEC SQL CONNECT RESET;
```

Extraction de contenus élémentaires et de valeurs d'attributs

Vous pouvez extraire un *contenu élémentaire* ou une *valeur d'attribut* à partir d'un ou plusieurs documents XML (recherche dans un document ou dans une collection). L'Extension XML comporte des fonctions d'extraction définies par l'utilisateur que vous pouvez indiquer dans la clause SQL SELECT pour chaque type de données SQL.

L'extraction de contenus élémentaires et de valeurs d'attributs est particulièrement utile lorsque vous développez vos applications car vous pouvez accéder aux données XML en tant que données relationnelles. Par exemple, 1000 documents peuvent être stockés dans la colonne ORDER de la

table SALES_TAB. Pour extraire les noms de tous les clients ayant passé commande, utilisez l'instruction SQL ci-après en indiquant la fonction UDF d'extraction dans la clause SELECT :

```
SELECT extractVarchar(Order,
'/Order/Customer/Name') from sales_order_view
   WHERE price > 2500.00
```

Dans cet exemple, la fonction UDF d'extraction extrait de la colonne ORDER l'élément <customer> en tant que données de type VARCHAR. Le chemin d'emplacement est `/Order/Customer/Name` (pour consulter la syntaxe du chemin d'emplacement, reportez-vous à la section «Chemin d'emplacement» à la page 55). Vous pouvez également réduire le nombre de valeurs renvoyées à l'aide d'une clause WHERE pour ne retenir que le contenu de l'élément <customer> associé à un sous-élément <ExtendedPrice> ayant une valeur supérieure à 2500,00.

Pour extraire un contenu élémentaire ou des valeurs d'attributs : Utilisez les fonctions UDF d'extraction répertoriées dans le tableau 13 à la page 131 en respectant la syntaxe ci-après pour les *fonctions scalaires* ou les fonctions de table :

extract*retrieved_datatype*(*xmlobj*, *path*)

Où :

retrieved_datatype
: Type de données renvoyé par la fonction d'extraction. Il peut s'agir de l'un des types suivants :
  - INTEGER
  - SMALLINT
  - DOUBLE
  - REAL
  - CHAR
  - VARCHAR
  - CLOB
  - DATE
  - TIME
  - TIMESTAMP
  - FILE

xmlobj
: Nom de la colonne XML à partir de laquelle l'élément ou l'attribut doit être extrait. Cette colonne doit être définie pour l'un des types UDT XML suivants :
  - XMLVARCHAR
  - XMLCLOB as LOCATOR

- XMLFILE

path Chemin d'emplacement de l'élément ou de l'attribut dans le document XML (par exemple, `/Order/Customer/Name`). Pour consulter la syntaxe du chemin d'emplacement, reportez-vous à la section «Chemin d'emplacement» à la page 55.

Important : Notez que la fonction UDF d'extration prend en charge les chemins d'emplacement qui comportent des prédicats avec des attributs, mais sans éléments. Par exemple, le prédicat qui suit est pris en charge :

```
'/Order/Part[@color="black "]/ExtendedPrice'
```

Par contre, le prédicat qui suit n'est pas pris en charge :

```
'/Order/Part/Shipment/[Shipdate < "11/25/00"]'
```

Le tableau 13 répertorie les fonctions d'extraction (fonctions scalaires et fonctions de table) :

Tableau 13. Fonctions d'extraction de l'Extension XML

Fonction scalaire	Fonction de table	Nom de colonne renvoyé (fonction de table)	Type de données renvoyé
extractInteger()	extractIntegers()	returnedInteger	INTEGER
extractSmallint()	extractSmallints()	returnedSmallint	SMALLINT
extractDouble()	extractDoubles()	returnedDouble	DOUBLE
extractReal()	extractReals()	returnedReal	REAL
extractChar()	extractChars()	returnedChar	CHAR
extractVarchar()	extractVarchars()	returnedVarchar	VARCHAR
extractCLOB()	extractCLOBs()	returnedCLOB	CLOB
extractDate()	extractDates()	returnedDate	DATE
extractTime()	extractTimes()	returnedTime	TIME
extractTimestamp()	extractTimestamps()	returnedTimestamp	TIMESTAMP

Exemple de fonction scalaire :

Dans l'exemple suivant, une seule valeur est renvoyée lorsque l'attribut key est égal à 1. La valeur extraite est de type INTEGER et est automatiquement convertie en valeur de type DECIMAL.

```
CREATE TABLE t1(key decimal(3,2));
INSERT into t1 values
SELECT * from table(db2xml.extractInteger(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/[@key="1"]'));
SELECT * from t1;
```

Exemple de fonction de table :

Dans l'exemple suivant, chaque valeur de clé de SALES_ORDER est extraite sous forme de valeur de type INTEGER :

```
SELECT * from table(db2xml.extractIntegers(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/@key')) as x;
```

Mise à jour de données XML

L'Extension XML permet de mettre à jour tout un document XML en remplaçant les données de colonne XML, ou uniquement les valeurs des éléments ou des attributs précisés.

Tâche en cours :

1. Assurez-vous que vous avez stocké le document XML dans une table XML et déterminez les données à extraire.
2. Choisissez une méthode de mise à jour des données dans la table DB2 (fonctions de transtypage ou fonctions UDF).
3. Indiquez une instruction SQL qui identifie la table et la colonne XML à mettre à jour.

Important : Lorsque vous mettez à jour une colonne activée pour XML, l'Extension XML actualise automatiquement les tables annexes en conséquence. Toutefois, n'effectuez aucune mise à jour directe dans ces tables sans mettre à jour le document XML d'origine dans la colonne XML pour éviter des incohérences dans les données.

Pour mettre à jour un document XML :

Utilisez l'une des méthodes suivantes :

Utilisation d'une fonction de transtypage par défaut

Pour chaque type UDT, il existe une fonction de transtypage par défaut permettant de transtyper le type de base SQL en type UDT. Vous pouvez mettre à jour le document XML à l'aide des fonctions de transtypage disponibles par l'Extension XML. Le tableau 9 à la page 125, répertorie les fonctions de transtypage fournies. Il suppose que la colonne ORDER est associée à un type UDT différent fourni par l'Extension XML.

Exemple : Mettez à jour le type XMLVARCHAR à partir du type VARCHAR transtypé, en supposant que la variable hôte `xml_buf` est associée au type VARCHAR.

```
UPDATE sales_tab VALUES('123456', 'Sriram Srinivasan',
  db2xml.XMLVarchar(:xml_buff))
```

Utilisation d'une fonction UDF d'archivage

Pour chaque type UDT Extension XML, il existe une fonction UDF d'archivage qui permet d'importer des données dans DB2, à partir d'une ressource autre que son type de base. Une fonction UDF d'archivage permet de mettre à jour tout le document XML par son remplacement.

Exemple :L'exemple ci-après illustre la mise à jour d'un document XML avec la fonction XMLVarcharFromFile() :

```
UPDATE sales_tab
       set order = XMLVarcharFromFile('c:\dxx\samples\cmd\getstart.xml')
     WHERE sales_person = 'Sriram Srinivasan'
```

Dans l'exemple précédent, l'objet XML est mis à jour du fichier `c:\dxx\samples\cmd\getstart.xml` vers la colonne ORDER de la table SALES_TAB.

Pour obtenir la liste des fonctions d'archivage fournies par l'Extension XML, reportez-vous au tableau 10 à la page 126.

Pour mettre à jour certains éléments et attributs d'un document XML :

Précisez les modifications voulues une par une à l'aide de la fonction UDF Update(), plutôt que de mettre à jour tout le document. Avec la fonction UDF, vous indiquez un chemin d'emplacement et la valeur de l'élément ou de l'attribut représenté par le chemin à remplacer. (Pour consulter la syntaxe du chemin d'emplacement, reportez-vous à la section «Chemin d'emplacement» à la page 55 .) Vous n'avez pas besoin d'éditer le document XML : l'Extension XML effectue les modifications à votre place.

La fonction UDF Update met à jour la totalité du fichier XML et reconstruit le fichier en fonction des informations obtenues à partir de l'analyseur XML. Pour savoir comment cette fonction agit sur le document et pour obtenir des exemples de documents avant et après leur mise à jour, reportez-vous à la section «Mode de traitement du document XML par la fonction Update» à la page 207.

Syntaxe :

`Update(`*`xmlobj`*`, `*`path`*`, `*`value`*`)`

Où :

xmlobj Nom de la colonne XML pour laquelle la valeur de l'élément ou de l'attribut doit être mise à jour.

path Chemin d'emplacement de l'élément ou de l'attribut à mettre à jour. Pour consulter la syntaxe du chemin d'emplacement, reportez-vous à la section «Chemin d'emplacement» à la page 55. Pour plus

d'informations sur les occurrences multiples, reportez-vous à la section «Occurrences multiples» à la page 210.

value Valeur à mettre à jour.

Exemple : L'instruction ci-après remplace la valeur de l'élément <Customer> par la chaîne de caractères IBM, avec la fonction UDF Update() :

```
UPDATE sales_tab
       set order = Update(order, '/Order/Customer/Name', 'IBM')
     WHERE sales_person = 'Sriram Srinivasan'
```

Occurrences multiples :

Lorsqu'un chemin d'emplacement est fourni dans la fonction UDF Update(), le contenu de chaque élément ou attribut ayant un chemin similaire est mis à jour avec la valeur fournie. Cela signifie que si un document comporte plusieurs chemins d'emplacement, le fonction Update remplace les valeurs existantes par la valeur fournie dans le paramètre *value*.

Recherche dans des documents XML

La recherche dans des données XML et l'extraction de celles-ci sont deux opérations identiques. Elles renvoient des données qui seront manipulées ultérieurement. Les recherches sont exécutées avec la clause WHERE pour définir des prédicats comme critères d'extraction.

L'Extension XML comporte plusieurs méthodes de recherche dans des documents XML de colonne XML, en fonction des besoins de l'application. Il permet d'effectuer des recherches dans la structure du document et de renvoyer des résultats en fonction du contenu élémentaire et des valeurs d'attributs. Vous pouvez effectuer des recherches dans une vue de la colonne XML et de ses tables annexes, directement dans les tables annexes pour améliorer les performances, ou utiliser des fonctions UDF d'extraction avec des clauses WHERE. Vous pouvez également vous servir de l'Extension Texte DB2 pour rechercher une chaîne texte dans des données de colonne liées au contenu structurel.

Pour accélérer les recherches, l'Extension XML permet d'utiliser sur des colonnes de tables annexes des index qui contiennent des valeurs d'éléments ou d'attributs XML extraites de documents XML. Lorsque vous indiquez le type de données d'un élément ou d'un attribut, vous pouvez effectuer des recherches d'après un type de données général SQL ou des plages de valeurs. Ainsi, dans l'exemple en cours, vous pouvez rechercher toutes les commandes dont le prix total est supérieur à 2500,00.

Vous pouvez également effectuer des recherches structurelles ou intégrales à l'aide de DB2 UDB Extension Texte. Par exemple, la colonne RESUME

contient des CV au format XML. Vous voulez le nom de tous les candidats possédant des compétences Java. Avec l'Extension Texte DB2, vous recherchez dans les documents XML tous les CV dans lesquels l'élément <skill> contient la chaîne de caractères JAVA.

Les sections suivantes décrivent les méthodes de recherche :

- «Recherche en fonction de la structure du document XML»
- «Recherche structurelle avec l'Extension Texte» à la page 137

Recherche en fonction de la structure du document XML

Les fonctions de recherche de l'Extension XML permettent d'effectuer des recherches dans une colonne XML d'après la structure du document (éléments ou attributs). Pour ce faire, vous utilisez une instruction SELECT de plusieurs façons et vous obtenez en retour un *ensemble de résultats* basé sur les analogies avec les éléments ou les attributs du document. Vous pouvez utiliser les méthodes de recherche suivantes :

- Interrogation directe des tables annexes
- *Vue de jointure*
- Fonctions UDF d'extraction
- Eléments ou attributs à occurrences multiples

Ces méthodes sont décrites dans les sections suivantes et comportent des exemples utilisant le scénario ci-après. La table d'application SALES_TAB comporte une colonne XML intitulée ORDER. Cette colonne contient trois tables annexes, ORDER_SIDE_TAB, PART_SIDE_TAB et SHIP_SIDE_TAB. La vue par défaut sales_order_view, indiquée lors de l'activation de la colonne ORDER, réalise la jointure de ces tables par l'instruction CREATE VIEW :

```
CREATE VIEW sales_order_view(invoice_num, sales_person, order,
                             order_key, customer, part_key, price, date)
     AS
     SELECT sales_tab.invoice_num, sales_tab.sales_person, sales_tab.order,
            order_side_tab.order_key, order_side_tab.customer,
            part_side_tab.part_key, ship_side_tab.date
     FROM sales_tab, order_side_tab, part_side_tab, ship_side_tab
     WHERE sales_tab.invoice_num = order_side_tab.invoice_num
               AND sales_tab.invoice_num = part_side_tab.invoice_num
               AND sales_tab.invoice_num = ship_side_tab.invoice_num
```

Recherche avec interrogation directe des tables annexes

L'interrogation directe par sous-requêtes fournit les meilleures performances pour la recherche structurelle lorsque les tables annexes sont indexées. Les requêtes ou les sous-requêtes permettent d'effectuer des recherches correctes dans les tables annexes.

Exemple : L'instruction ci-après exécute une recherche directe dans une table annexe par une requête ou une sous-requête :

```
SELECT sales_person from sales_tab
            WHERE invoice_num in
     (SELECT invoice_num from part_side_tab
               WHERE price > 2500.00 )
```

Dans cet exemple, invoice_num représente la clé primaire de la table SALES_TAB.

Recherche à partir d'une vue de jointure

L'Extension XML peut créer une vue par défaut qui réalise une jointure de la table d'application et des tables annexes par un identificateur unique. A l'aide de cette vue par défaut ou de toute vue équivalente, vous pouvez effectuer des recherches dans les données de colonne et interroger les tables annexes. Cette méthode fournit une vue virtuelle unique de la table d'application et de ses tables annexes. Cependant, le coût de la requête est proportionnel au nombre de tables annexes créées.

Astuce : Lors de la création de votre propre vue, vous pouvez réaliser une jointure de tables à l'aide de l'identificateur racine *root_id* ou de l'identificateur DXXROOT_ID créé par l'Extension XML.

Exemple : L'instruction ci-après exécute une recherche dans une vue :

```
SELECT sales_person from sales_order_view
   WHERE price > 2500.00
```

L'instruction SQL renvoie les valeurs sales_person de la table sales_order_view pour lesquelles les commandes ont un prix supérieur à 2500,00.

Recherche à l'aide des fonctions UDF d'extraction

Les fonctions UDF d'extraction fournies par l'Extension XML vous permettent également d'effectuer des recherches sur des éléments et des attributs lorsque vous n'avez pas créé d'index ni de table annexe pour la table d'application. L'examen de données XML à l'aide des fonctions UDF d'extraction est très coûteux et ne doit être effectué qu'avec des clauses WHERE limitant le nombre de documents XML concernés.

Exemple : L'instruction ci-après exécute une recherche avec une fonction UDF d'extraction fournie par l'Extension XML :

```
SELECT sales_person from sales_tab
     WHERE extractVarchar(order, '/Order/Customer/Name')
    like '%IBM%'
AND invoice_num > 100
```

Dans cet exemple, la fonction UDF d'extraction extrait les éléments </Order/Customer/Name> contenant la valeur IBM.

Recherche sur des éléments ou des attributs à occurrences multiples

Lors des recherches sur des éléments ou des attributs à occurrences multiples, la clause DISTINCT permet d'éviter les valeurs en double.

Exemple : L'instruction ci-après exécute une recherche avec la clause DISTINCT :

```
SELECT sales_person from sales_tab
            WHERE invoice_num in
              (SELECT DISTINCT invoice_num from part_side_tab
               WHERE price > 2500.00 )
```

Dans cet exemple, le fichier DAD indique qu'il existe plusieurs occurrences de `/Order/Part/Price` et crée la table annexe PART_SIDE_TAB correspondante. Celle-ci peut contenir plusieurs lignes associées au même numéro de facture (invoice_num). La clause DISTINCT ne renvoie que des valeurs uniques.

Recherche structurelle avec l'Extension Texte

Lorsque Extension XML exécute une recherche sur la structure d'un document XML, elle parcourt les valeurs d'éléments et d'attributs converties en données de type général, mais n'examine pas le texte. DB2 UDB Extension Texte permet d'exécuter des recherches structurelles ou intégrales sur une colonne activée pour XML. L'Extension Texte accepte la recherche de documents XML à partir de DB2 UDB version 6.1.L'Extension Texte est disponible sous Windows, AIX et Sun Solaris.

Recherche structurelle

Examine les chaînes texte en fonction de l'arborescence du document XML. Par exemple, si la structure de votre document est `/Order/Customer/Name` et que vous voulez rechercher la chaîne de caractères «IBM» dans le sous-élément <Customer>, vous pouvez lancer une recherche structurelle. Le document peut contenir la chaîne IBM dans un sous-élément <Comment> ou dans un nom de produit, mais la recherche structurelle ne la recherche que dans les éléments indiqués. Dans cet exemple, vous n'obtenez que les documents pour lesquels la chaîne IBM est trouvée dans le sous-élément </Order/Customer/Name>. Ceux qui contiennent la chaîne IBM dans d'autres éléments ne sont pas renvoyés.

Recherche intégrale

Examine les chaînes texte dans toute la structure du document, indépendamment des éléments ou des attributs. Ainsi dans l'exemple précédent, tous les documents contenant la chaîne de caractères IBM sont renvoyés, quel que soit l'emplacement de l'occurrence trouvée.

Pour pouvoir utiliser la fonction recherche de l'Extension Texte, vous devez installer DB2 Extension Texte, puis activer votre base de données et vos tables conformément aux instructions ci-dessous. Pour apprendre à utiliser la fonction recherche de l'Extension Texte, reportez-vous au chapitre sur

l'exécution de recherches avec les fonctions UDF de l'Extension Texte, dans le manuel *DB2 UDB Extension Texte - Administration et programmation*.

Activation d'une colonne XML pour l'Extension Texte

Supposons que vous ayez préparé la base de données en vue de son utilisation avec XML. Suivez les étapes ci-après pour activer l'Extension Texte afin d'examiner le contenu d'une colonne activée pour XML. Soient la base de données SALES_DB, la table ORDER et deux colonnes XML XVARCHAR et XCLOB.

1. Dans le fichier `install.txt` figurant sur le CD des Extensions, consultez la procédure d'installation de l'Extension Texte.
2. Entrez la commande **txstart** à partir de l'un des emplacements suivants :
   - Sous les systèmes d'exploitation UNIX, entrez la commande à partir de l'invite du propriétaire d'instance.
   - Sous Windows NT, entrez la commande à partir de la fenêtre de commande où DB2INSTANCE est indiqué.
3. Ouvrez la fenêtre de ligne de commande de l'Extension Texte. Cette étape suppose que vous disposez d'une base de données SALES_DB et d'une table ORDER qui contient deux colonnes XML XVARCHAR et XCLOB. Vous devez exécuter les programmes exemples dans `dxx\samples\c`.
4. Connectez-vous à la base de données. A l'invite **db2tx**, tapez :

   ```
   'connect to SALES_DB'
   ```
5. Activez la base de données pour l'Extension Texte.

   A partir de l'invite **db2tx**, tapez :

   ```
   'enable database'
   ```
6. Pour activer les colonnes dans la table XML pour l'Extension Texte, définissez les types de données associés au document XML, la langue, les pages de codes, ainsi que d'autres informations relatives à la colonne.
   - Pour la colonne VARCHAR nommée XVARCHAR, tapez :

     ```
     'enable text column order xvarchar function db2xml.varchartovarchar handle
     varcharhandle ccsid 850 language us_english format xml indextype precise
         indexproperty sections_enabled
     documentmodel (Order) updateindex update'
     ```
   - Pour la colonne CLOB nommée XCLOB, tapez :

     ```
     'enable text column order xclob function db2xml.clob handle clobhandle
      ccsid 850 language us_english indextype precise updateindex update'
     ```
7. Vérifiez l'état de l'index.
   - Pour la colonne XVARCHAR, tapez : `get index status order handle varcharhandle`
   - Pour la colonne XCLOB, tapez : `get index status order handle clobhandle`

8. Définissez le modèle de document XML dans un fichier d'initialisation de modèles de documents desmodel.ini. Ce fichier se trouve dans /db2tx/txins000 sous UNIX et dans \instance\\db2tx\txins000 sous Windows NT. Par exemple, pour le fichier textmodel.ini :

```
;list of document models
[MODELS]
modelname=Order

; an 'Order' document model definition
; left side = section name identifier
; right side = section name tag

[Order]
Order = /Order
Order/Customer/Name = /Order/Customer/Name
Order/Customer/Email = /Order/Customer/Email
Order/Part/@color = /Order/Part/@color
Order/Part/Shipment/ShipMode = /Order/Part/Shipment/ShipMode
```

Recherche de texte avec l'Extension Texte

La fonction de recherche de l'Extension Texte est tout à fait compatible avec la fonction de recherche structurelle de l'Extension XML. La méthode recommandée consiste à créer une requête qui exécute une recherche sur les éléments ou les attributs du document et examine le contenu élémentaire ou les valeurs d'attributs avec l'Extension Texte.

Exemple : Les instructions ci-après examinent le texte d'un document XML avec l'Extension Texte. A l'invite DB2, tapez :

```
'connect to SALES_DB'
'select xvarchar from order where db2tx.contains(varcharhandle,
 'model Order section(Order/Customer/Name) "Motors"')=1'
'select xclob from order where db2tx.contains(clobhandle,
 'model Order section(Order/Customer/Name) "Motors"')=1'
```

L'Extension Texte contient () des fonctions de recherche UDF.

Cet exemple n'illustre pas toutes les étapes nécessaires pour exécuter des recherches dans des données de colonne avec l'Extension Texte. Pour plus d'informations, reportez-vous au chapitre sur l'exécution de recherches avec les fonctions UDF de l'Extension Texte, dans le manuel *DB2 UDB Extension Texte - Administration et programmation*.

Suppression de documents XML

Pour supprimer une ligne de documents XML dans une colonne XML, servez-vous de l'instruction SQL DELETE. Vous pouvez ajouter des clauses WHERE pour préciser les documents à supprimer.

Exemple : Les instructions ci-après suppriment tous les documents dans lesquels la valeur <ExtendedPrice> est supérieure à 2500,00.

```
DELETE from sales_tab
            WHERE invoice_num in
      (SELECT invoice_num from part_side_tab
                WHERE price > 2500.00 )
```

Limitations relatives à l'appel de fonctions à partir de JDBC

Lorsque vous utilisez des marqueurs de paramètres dans des fonctions, en raison d'une restriction JDBC, le marqueur de paramètre de la fonction doit être transtypé vers le type de données de la colonne dans laquelle les données renvoyées seront insérées. La logique de sélection de fonction ne connaît pas le type de données dans lequel l'argument devrait être transtypé. La référence ne peut alors pas être décodée.

JDBC ne peut alors résoudre le code suivant :

```
db2xml.XML default_casting_function(length)
```

Vous pouvez utiliser la spécification CAST pour fournir un type au marqueur de paramètre, tel que VARCHAR. La logique de sélection de fonction peut alors fonctionner :

```
db2xml.XML default_casting_function(CAST(? AS cast_type(length))
```

Exemple 1 : Dans l'exemple suivant, le marqueur de paramètre est transtypé en VARCHAR. Le paramètre transféré est un document XML, qui est transtypé en VARCHAR(1000) et inséré dans la colonne ORDER.

```
String query = "insert into sales_tab(invoice_num, sales_person, order) values
    (?,?,db2xml.XMLVarchar(cast (? as varchar(1000))))";
```

Exemple 2 : Dans l'exemple suivant, le marqueur de paramètre est transtypé en VARCHAR. Le paramètre transféré est un nom de fichier et son contenu est converti en données de type VARCHAR(1000) et inséré dans la colonne ORDER.

```
String query = "insert into sales_tab(invoice_num, sales_person, order) values
    (?,?,db2xml.XMLVarcharfromFILE(cast (? as varchar(1000))))";
```

Chapitre 6. Gestion des données de collection XML

Une collection XML est un ensemble de tables relationnelles contenant des données mappées vers des documents XML. Cette méthode d'accès et de stockage permet de composer un document XML à partir de données existantes, de le décomposer et d'utiliser XML comme méthode d'échange.

Les tables relationnelles peuvent être des tables nouvelles générées par l'Extension XML lors de la décomposition de documents XML, ou des tables existantes qui contiennent des données permettant de générer avec l'Extension XML des documents XML destinés aux applications. Dans ces tables, les données de colonne ne contiennent aucune balise XML, mais les contenus élémentaires et les valeurs d'attributs. Les procédures mémorisées servent de méthode d'accès et de stockage pour stocker, extraire, mettre à jour, examiner et supprimer des données de collection XML.

Les limites des paramètres utilisés par les procédures mémorisées des collections XML sont détaillées à l'«Annexe D. Les limites de l'Extension XML» à la page 297.

Vous pouvez augmenter les tailles des paramètres CLOB dans les résultats des procédures mémorisées. Pour cela, reportez-vous à la section «Augmentation des limites du paramètre CLOB» à la page 216.

Pour obtenir des informations vous permettant de gérer votre collection XML, reportez-vous aux sections suivantes :

- «Composition de documents XML à partir de données DB2»
- «Décomposition de documents XML en données DB2» à la page 149

Composition de documents XML à partir de données DB2

La composition consiste à générer un ensemble de documents XML à partir de données relationnelles figurant dans une collection XML. Vous pouvez composer des documents XML à l'aide de procédures mémorisées. Dans ce cas, vous devez créer un fichier DAD qui définit le mappage entre le document XML et la structure des tables DB2. Les procédures mémorisées composent le document XML conformément au fichier DAD. Pour apprendre à créer un fichier DAD, reportez-vous à la section «Planification des collections XML» à la page 58.

Avant de commencer

- Mappez la structure du document XML vers les tables relationnelles où résident les contenus élémentaires et les valeurs d'attributs.
- Sélectionnez un mode de mappage : SQL ou noeud RDB.
- Préparez le fichier DAD. Pour plus d'informations, reportez-vous à la section «Planification des collections XML» à la page 58.
- Vous pouvez également activer la collection XML.

Composition du document XML

L'Extension XML comporte deux procédures mémorisées, dxxGenXML() et dxxRetrieveXML(), pour composer des documents XML.

dxxGenXML()

Cette procédure mémorisée est utilisée pour les applications qui exécutent des mises à jour occasionnelles ou qui ne veulent pas se charger de l'administration des données XML. Elle ne nécessite pas l'activation d'une collection, mais a recours à un fichier DAD.

Elle construit des documents XML à partir des tables de collection XML indiquées par l'élément <Xcollection> dans le fichier DAD. Cette procédure mémorisée insère chaque document XML sous forme de ligne dans une *table de résultats*. Vous pouvez également ouvrir un curseur dans la table de résultats et extraire l'ensemble de résultats. La table de résultats, qui doit toujours être créée par l'application, comporte une seule colonne de type VARCHAR, CLOB, XMLVARCHAR ou XMLCLOB.

En outre, si vous attribuez la valeur YES à l'élément de validation figurant dans le fichier DAD, l'Extension XML ajoute la colonne DXX_VALID de type INTEGER et insère la valeur 1 ou 0, selon que le document XML est valide ou non .

La procédure mémorisée dxxGenXML() permet également de préciser le nombre maximal de lignes à générer dans la table de résultats dans le but de réduire les temps de traitement. Elle renvoie le nombre réel de ligne composant la table, ainsi que des codes et messages retour.

La procédure de décomposition correspondante est dxxShredXML(). Cette dernière utilise également la DAD comme paramètre d'entrée et ne nécessite pas l'activation d'une collection XML.

Pour composer une collection XML : dxxGenXML()

Imbriquez un appel de procédure mémorisée dans votre application par la déclaration suivante :

```
dxxGenXML(CLOB(100K)     DAD,             /* input */
     char(resultTabName) resultTabName,   /* input */
          integer        overrideType,    /* input */
          varchar(1024)  override,        /* input */
```

```
            integer        maxRows,         /* input */
            integer        numRows,         /* output */
            long           returnCode,       /* output */
            varchar(1024)  returnMsg)        /* output */
```

Pour consulter la syntaxe complète et des exemples, reportez-vous à la section «dxxGenXML()» à la page 226.

Exemple : L'exemple ci-après illustre la composition d'un document XML :

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE is CLOB(100K) dad;              /* DAD */
     SQL TYPE is CLOB_FILE  dadFile;          /* dad file */
         char    result_tab[32];  /* name of the result table */
              char    override[2];      /* override, will set to NULL*/
              short   overrideType;     /* defined in dxx.h */
         short   max_row;         /* maximum number of rows */
         short   num_row;         /* actual number of rows */
         long    returnCode;      /* return error code */
         char                  returnMsg[1024];  /* error message text */
         short                 dad_ind;
         short   rtab_ind;
         short   ovtype_ind;
         short   ov_inde;
         short   maxrow_ind;
         short   numrow_ind;
         short                 returnCode_ind;
         short                 returnMsg_ind;

  EXEC SQL END DECLARE SECTION;

         /* create table */
         EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

     /* read data from a file to a CLOB */
     strcpy(dadfile.name,"c:\dxx\samples\dad\getstart_xcollection.dad");
     dadfile.name_length = strlen("c:\dxx\samples\dad\getstart_xcollection.dad");
     dadfile.file_options = SQL_FILE_READ;
     EXEC SQL VALUES (:dadfile) INTO :dad;
         strcpy(result_tab,"xml_order_tab");
     override[0] = '\0';
     overrideType = NO_OVERRIDE;
         max_row = 500;
         num_row = 0;
         returnCode = 0;
         msg_txt[0] = '\0';
         collection_ind = 0;
        dad_ind = 0;
         rtab_ind = 0;
     ov_ind = -1;
         ovtype_ind = 0;
         maxrow_ind = 0;
         numrow_ind = -1;
         returnCode_ind = -1;
         returnMsg_ind = -1;

         /* Call the store procedure */
     EXEC SQL CALL dxxGenXML(:dad:dad_ind,
                :result_tab:rtab_ind,
          :overrideType:ovtype_ind,:override:ov_ind,
                :max_row:maxrow_ind,:num_row:numrow_ind,
                      :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Une fois la procédure mémorisée appelée, la table de résultats contient 250 lignes car la requête SQL indiquée dans le fichier DAD a généré 250 documents XML.

dxxRetrieveXML()

Cette procédure mémorisée est utilisée pour les applications qui exécutent des mises à jour régulières. En raison de la répétition des mêmes tâches, il est important d'optimiser les performances. C'est ce que permet l'activation d'une collection XML et l'utilisation de son nom dans la procédure mémorisée.

La procédure mémorisée dxxRetrieveXML() fonctionne comme dxxGenXML(), excepté qu'elle utilise une collection XML activée, et non un fichier DAD. Lorsqu'une collection XML est activée, un fichier DAD est stocké dans la table XML_USAGE. L'Extension XML récupère le fichier DAD puis, à partir de cette action, dxxRetrieveXML() se comporte comme dxxGenXML().

dxxRetrieveXML() permet d'utiliser le même fichier DAD pour les opérations de composition et de décomposition. Cette procédure mémorisée permet également de récupérer des documents XML décomposés.

La procédure de décomposition correspondante est dxxInsertXML(). Cette dernière utilise également une collection XML activée.

Pour composer une collection XML : dxxRetrieveXML()

Imbriquez un appel de procédure mémorisée dans votre application par la déclaration suivante :

```
dxxRetrieveXML(char(collectionName) collectionName,  /* input */
               char(resultTabName) resultTabName,   /* input */
               integer        overrideType,    /* input */
               varchar(1024)  override,        /* input */
               integer        maxRows,         /* input */
               integer        numRows,         /* output */
             long           returnCode,       /* output */
             varchar(1024)  returnMsg)        /* output */
```

Pour consulter la syntaxe complète et des exemples, reportez-vous à la section «dxxRetrieveXML()» à la page 230.

Exemple : Voici un exemple d'appel de la procédure dxxRetrieveXML(). Soit une table de résultats XML_ORDER_TAB, contenant une colonne de type XMLVARCHAR.

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;
```

```
        char    collection[32];  /* dad buffer */
        char    result_tab[32];  /* name of the result table */
             char    override[2];      /* override, will set to NULL*/
             short   overrideType;     /* defined in dxx.h */
        short   max_row;         /* maximum number of rows */
        short   num_row;         /* actual number of rows */
        long    returnCode;      /* return error code */
        char                  returnMsg[1024];  /* error message text */
        short   dadbuf_ind;
        short   rtab_ind;
        short   ovtype_ind;
        short   ov_inde;
        short   maxrow_ind;
        short   numrow_ind;
        short                 returnCode_ind;
        short                 returnMsg_ind;

EXEC SQL END DECLARE SECTION;

        /* create table */
             EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

        /* initialize host variable and indicators */
        strcpy(collection,"sales_ord");
        strcpy(result_tab,"xml_order_tab");
   override[0] = '\0';
   overrideType = NO_OVERRIDE;
        max_row = 500;
        num_row = 0;
        returnCode = 0;
        msg_txt[0] = '\0';
        collection_ind = 0;
        rtab_ind = 0;
   ov_ind = -1;
        ovtype_ind = 0;
        maxrow_ind = 0;
        numrow_ind = -1;
        returnCode_ind = -1;
        returnMsg_ind = -1;

        /* Call the store procedure */
        EXEC SQL CALL dxxRetrieve(:collection:collection_ind;
               :result_tab:rtab_ind,
         :overrideType:ovtype_ind,:override:ov_ind,
               :max_row:maxrow_ind,:num_row:numrow_ind,
                     :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Remplacement dynamique de valeurs dans le fichier DAD

Pour les requêtes dynamiques, deux paramètres facultatifs permettent de remplacer des conditions dans le fichier DAD : *override* et *overrideType*. En fonction des données d'entrée du paramètre *overrideType*, l'application peut remplacer dans le fichier DAD les valeurs de la balise <SQL_stmt> par le mode de mappage SQL ou les conditions RDB_node par le mode de mappage du noeud RDB.

Ces paramètres sont associés aux valeurs et aux règles suivantes :

overrideType
Paramètre d'entrée (IN) obligatoire, précisant le type du paramètre *override*. *overrideType* peut prendre les valeurs suivantes :

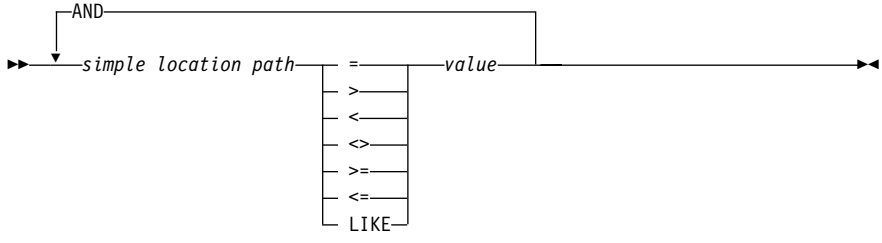
NO_OVERRIDE
Indique de ne remplacer aucune condition dans le fichier DAD.

SQL_OVERRIDE
Indique de remplacer une condition du fichier DAD par une instruction SQL.

XML_OVERRIDE
Indique de remplacer une condition du fichier DAD par une condition basée sur XPath.

override
Paramètre d'entrée (IN) facultatif, précisant la condition override pour le fichier DAD. La valeur d'entrée dépend de la valeur du paramètre *overrideType*.

- NO_OVERRIDE : chaîne de type NULL.
- SQL_OVERRIDE : instruction SQL valide. ***Valeur obligatoire :*** avec SQL_OVERRIDE et une instruction SQL, vous devez utiliser le schéma de mappage SQL dans le fichier DAD. L'instruction SQL d'entrée remplace l'instruction SQL indiquée par l'élément <SQL_stmt> dans le fichier DAD.
- XML_OVERRIDE : chaîne d'une ou plusieurs expressions. ***Valeur obligatoire :*** avec XML_OVERRIDE et une expression, vous devez utiliser le schéma de mappage du noeud RDB dans le fichier DAD. L'expression XML d'entrée remplace la condition RDB_node indiquée dans le fichier DAD. La syntaxe de cette expression est la suivante :



Où :

simple location path
Chemin d'emplacement simple utilisant la syntaxe définie par XPATH. Pour plus d'informations sur cette syntaxe, reportez-vous au tableau 5 à la page 57.

opérateurs
Un espace peut séparer l'opérateur des autres parties de l'expression.

value
Valeur numérique ou chaîne simple placée entre guillemets.

Des espaces facultatifs peuvent figurer avant et après les opérations. Les espaces avant et après l'opérateur LIKE sont obligatoires.

Lorsque XML_OVERRIDE est indiqué, la condition RDB_node de l'élément text_node ou attribute_node correspondant au chemin d'emplacement simple est remplacée par l'expression précisée.

XML_OVERRIDE n'est pas complètement compatible avec XPath. Le chemin d'emplacement simple permet uniquement d'identifier l'élément ou l'attribut mappé vers une colonne.

Exemples :

Les exemples ci-après illustrent le remplacement dynamique à l'aide de SQL_OVERRIDE et XML_OVERRIDE. La plupart des exemples de procédures mémorisées présentés dans ce manuel utilisent NO_OVERRIDE.

Exemple : Procédure mémorisée utilisant SQL_OVERRIDE.

```
include "dxx.h"
include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;
         char    collection[32];  /* dad buffer */
         char    result_tab[32];  /* name of the result table */
         char    override[256];   /* override, SQL_stmt */
         short   overrideType;      /* defined in dxx.h */
         short   max_row;         /* maximum number of rows */
         short   num_row;         /* actual number of rows */
         long    returnCode;      /* return error code */
         char                  returnMsg[1024];  /* error message text */
         short   rtab_ind;
         short   ovtype_ind;
         short   ov_inde;
         short   maxrow_ind;
         short   numrow_ind;
         short                 returnCode_ind;
         short                 returnMsg_ind;

  EXEC SQL END DECLARE SECTION;

         /* create table */
```

```
EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

/* initialize host variable and indicators */
strcpy(collection,"sales_ord");
strcpy(result_tab,"xml_order_tab");
sprintf(override,"%s %s %s %s %s %s %s",
          "SELECT o.order_key, customer, p.part_key, quantity, price,",
          "tax, ship_id, date, mode ",
          "FROM order_tab o, part_tab p,",
          "table(select substr(char(timestamp(generate_unique())),16",
          "as ship_id,date,mode from ship_tab)as s",
          "WHERE p.price > 50.00 and s.date >'1998-12-01' AND",
          "p.order_key = o.order_key and s.part_key = p.part_key");
overrideType = SQL_OVERRIDE;
max_row = 500;
num_row = 0;
returnCode = 0;
msg_txt[0] = '\0';
collection_ind = 0;
rtab_ind = 0;
ov_ind = 0;
ovtype_ind = 0;
maxrow_ind = 0;
numrow_ind = -1;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Call the store procedure */
EXEC SQL CALL dxxRetrieve(:collection:collection_ind;
       :result_tab:rtab_ind,
       :overrideType:ovtype_ind,:override:ov_ind,
       :max_row:maxrow_ind,:num_row:numrow_ind,
             :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Dans cet exemple, l'élément <xcollection> du fichier DAD doit être associé à un élément <SQL_stmt>. Le paramètre *override* remplace la valeur de l'élément <SQL_stmt>, le nouveau prix devant être supérieur à 50,00 et la nouvelle date postérieure au 1er décembre 1998 (1998-12-01).

Exemple : Procédure mémorisée utilisant XML_OVERRIDE.

```
include "dxx.h"
include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;
         char    collection[32];  /* dad buffer */
         char    result_tab[32];  /* name of the result table */
         char    override[256];   /* override, SQL_stmt */
         short   overrideType;    /* defined in dxx.h */
         short   max_row;         /* maximum number of rows */
         short   num_row;         /* actual number of rows */
         long    returnCode;      /* return error code */
```

```
        char                  returnMsg[1024];  /* error message text */
        short   dadbuf_ind;
        short   rtab_ind;
        short   ovtype_ind;
        short   ov_inde;
        short   maxrow_ind;
        short   numrow_ind;
        short                 returnCode_ind;
        short                 returnMsg_ind;

EXEC SQL END DECLARE SECTION;

        /* create table */
        EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

        /* initialize host variable and indicators */
        strcpy(collection,"sales_ord");
        strcpy(result_tab,"xml_order_tab");
        sprintf(override,"%s %s",
                  "/Order/Part/Price > 50.00 AND ",
                  "Order/Part/Shipment/ShipDate > '1998-12-01'");
       overrideType = XML_OVERRIDE;
        max_row = 500;
        num_row = 0;
        returnCode = 0;
        msg_txt[0] = '\0';
        collection_ind = 0;
        rtab_ind = 0;
        ov_ind = 0;
        ovtype_ind = 0;
        maxrow_ind = 0;
        numrow_ind = -1;
        returnCode_ind = -1;
        returnMsg_ind = -1;

        /* Call the store procedure */
        EXEC SQL CALL dxxRetrieve(:collection:collection_ind;
               :result_tab:rtab_ind,
               :overrideType:ovtype_ind,:override:ov_ind,
               :max_row:maxrow_ind,:num_row:numrow_ind,
                     :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Dans cet exemple, l'élément <collection> du fichier DAD est associé à un noeud RDB pour le noeud d'élément racine. La valeur du paramètre *override* est basée sur du contenu XML. L'Extension XML substitue la colonne DB2 mappée au chemin d'emplacement simple.

Décomposition de documents XML en données DB2

La décomposition d'un document XML consiste à décomposer les données d'un document XML pour les stocker dans des tables relationnelles. Pour ce faire, l'Extension XML fournit des procédures mémorisées.Pour pouvoir les utiliser, vous devez créer un fichier DAD qui définit le mappage entre le

document XML et la structure des tables DB2. Les procédures mémorisées décomposent le document XML conformément au fichier DAD. Pour apprendre à créer un fichier DAD, reportez-vous à la section «Planification des collections XML» à la page 58.

Activation d'une collection XML pour la décomposition

En général, vous devez d'abord activer une collection XML pour pouvoir utiliser les procédures mémorisées. Vous devez le faire dans les cas suivants :

- Lorsque vous décomposez des documents XML en nouvelles tables, vous devez activer une collection XML car toutes les tables qu'elle contient sont créées par l'Extension XML lors de l'activation.
- Il est important de conserver la séquence d'éléments et d'attributs à occurrences multiples. DB2 Extension XML ne le fait que pour les tables qu'il crée lorsque vous activez une collection. Lors de la décomposition de documents XML en tables relationnelles existantes, le maintien de la séquence n'est pas garanti.

Pour transmettre automatiquement le fichier DAD lorsque les tables existent dans la base de données, vous n'avez pas besoin d'activer une collection XML.

Restriction relative à la taille des tables pour la décomposition

A partir du mappage du noeud RDB, l'opération de décomposition indique comment un document XML est décomposé dans des tables DB2 par l'extraction de valeurs d'éléments et d'attributs qui sont ensuite insérées dans des lignes de la table. Les valeurs de chaque document XML sont stockées dans une ou plusieurs tables DB2. Chaque table peut comporter jusqu'à 1024 lignes par document décomposé.

Par exemple, si un document XML est décomposé en cinq tables, chacune de ces cinq tables peut comporter jusqu'à 1024 lignes. Si la table concerne plusieurs documents, elle peut contenir jusqu'à 1024 lignes pour chacun d'eux. Par exemple, si la table contient 20 documents, elle peut comporter 20480 lignes (1024 lignes x 20 documents).

L'utilisation d'éléments à occurrences multiples (c'est-à-dire d'éléments dont les chemins d'emplacement peuvent apparaître plusieurs fois dans la structure XML) a une incidence sur le nombre de lignes. Par exemple, un document contenant un élément <Part> qui apparaît 20 fois, peut être décomposé en 20 lignes dans une table. Tenez compte de cette restriction relative à la taille des tables lorsque vous utilisez des éléments à occurrences multiples.

Avant de commencer

- Mappez la structure du document XML vers les tables relationnelles où résident les contenus élémentaires et les valeurs d'attributs.

- Préparez le fichier DAD en mode mappage du noeud RDB. Pour plus d'informations, reportez-vous à la section «Planification des collections XML» à la page 58.
- Vous pouvez également activer la collection XML.

Décomposition du document XML

L'Extension XML comporte deux procédures mémorisées, dxxShredXML() et dxxInsertXML(), pour décomposer des documents XML.

dxxShredXML()

Cette procédure mémorisée est utilisée pour les applications qui exécutent des mises à jour occasionnelles ou qui ne veulent pas se charger de l'administration des données XML. Elle ne nécessite pas l'activation d'une collection, mais a recours à un fichier DAD.

La procédure mémorisée dxxShredXML() utilise deux paramètres d'entrée, un fichier DAD et le document XML à décomposer. Elle renvoie deux paramètres de sortie, un code et un message retour.

La procédure mémorisée dxxShredXML() insère un document XML dans une collection XML conformément à la spécification <Xcollection> contenue dans le fichier DAD d'entrée. Les tables utilisées dans l'élément <Xcollection> du fichier DAD doivent exister et les colonnes respecter les types de données indiqués dans le mappage DAD. Si ces conditions ne sont pas remplies, un message d'erreur est renvoyé. Ensuite, la procédure mémorisée dxxShredXML() décompose le document XML et insère les données XML non balisées dans les tables indiquées dans le fichier DAD.

La procédure de composition correspondante est dxxGenXML(). Cette dernière utilise également la DAD comme paramètre d'entrée et ne nécessite pas l'activation d'une collection XML.

Pour décomposer une collection XML : dxxShredXML()

Imbriquez un appel de procédure mémorisée dans votre application par la déclaration suivante :

```
dxxShredXML(CLOB(100K)    DAD,            /* input */
            CLOB(1M)        xmlobj,           /* input */
            long            returnCode,       /* output */
            varchar(1024)   returnMsg)        /* output */
```

Pour consulter la syntaxe complète et des exemples, reportez-vous à la section «dxxShredXML()» à la page 235.

Exemple : Voici un exemple d'appel de la procédure dxxShredXML() :

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
```

```
EXEC SQL BEGIN DECLARE SECTION;
        SQL TYPE is CLOB      dad;             /* DAD*/
        SQL TYPE is CLOB_FILE dadFile;         /* DAD file*/
        SQL TYPE is CLOB      xmlDoc;          /* input XML document */
        SQL TYPE is CLOB_FILE xmlFile;          /* input XML file */
        long                  returnCode;       /* error code */
        char                  returnMsg[1024];  /* error message text */
        short                 dad_ind;
        short                 xmlDoc_ind;
        short                 returnCode_ind;
        short                 returnMsg_ind;
  EXEC SQL END DECLARE SECTION;

        /* initialize host variable and indicators */
       strcpy(dadFile.name,"c:\dxx\samples\dad\getstart_xcollection.dad");
       dadFile.name_length=strlen("c:\dxx\samples\dad\getstart_xcollection.dad");
       dadFile.file_option=SQL_FILE_READ;
       strcpy(xmlFile.name,"c:\dxx\samples\cmd\getstart_xcollection.xml");
       xmlFile.name_length=strlen("c:\dxx\samples\cmd\getstart_xcollection.xml");
       xmlFile.file_option=SQL_FILE_READ;
       SQL EXEC VALUES (:dadFile) INTO :dad;
       SQL EXEC VALUES (:xmlFile) INTO :xmlDoc;
        returnCode = 0;
        returnMsg[0] = '\0';
       dad_ind = 0;
       xmlDoc_ind = 0;
        returnCode_ind = -1;
        returnMsg_ind = -1;

        /* Call the store procedure */
       EXEC SQL CALL db2xml.dxxShredXML(:dad:dad_ind;
                    :xmlDoc:xmlDoc_ind,
                    :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

dxxInsertXML()

Cette procédure mémorisée est utilisée pour les applications qui exécutent des mises à jour régulières. En raison de la répétition des mêmes tâches, il est important d'optimiser les performances. C'est ce que permet l'activation d'une collection XML et l'utilisation de son nom dans la procédure mémorisée. La procédure mémorisée dxxInsertXML() fonctionne comme dxxShredXML(), excepté qu'elle utilise une collection XML activée comme premier paramètre d'entrée.

La procédure mémorisée dxxInsertXML() insère un document XML dans une collection XML activée, associée à un fichier DAD. Ce dernier contient des spécifications relatives aux tables de la collection et au mappage. Les tables de la collection sont vérifiées ou créées en fonction des spécifications contenues dans l'élément <Xcollection>. Ensuite, la procédure mémorisée dxxInsertXML() décompose le document XML en fonction du mappage et insère les données XML non balisées dans les tables de la collection XML indiquée.

La procédure de composition correspondante est dxxRetrieveXML(). Cette dernière utilise également une collection XML activée.

Pour décomposer une collection XML : dxxInsertXML()

Imbriquez un appel de procédure mémorisée dans votre application par la déclaration suivante :

```
dxxInsertXML(char(collectionName) collectionName,   /* input */
             CLOB(1M)       xmlobj,           /* input */
             long           returnCode,       /* output */
             varchar(1024)  returnMsg)        /* output */
```

Pour consulter la syntaxe complète et des exemples, reportez-vous à la section «dxxInsertXML()» à la page 237.

Exemple : Voici un exemple d'appel de la procédure dxxInsertXML():

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;
         char                 collection[64];   /* name of an XML collection */
         SQL TYPE is CLOB_FILE xmlFile;         /* input XML file */
         SQL TYPE is CLOB      xmlDoc;          /* input XML doc */
         long                 returnCode;       /* error code */
         char                 returnMsg[1024];  /* error message text */
         short                collection_ind;
         short                xmlDoc_ind;
         short                returnCode_ind;
         short                returnMsg_ind;
  EXEC SQL END DECLARE SECTION;

         /* initialize host variable and indicators */
         strcpy(collection,"sales_ord")
         strcpy(xmlobj.name,"c:\dxx\samples\cmd\getstart_xcollection.xml");
         xmlobj.name_length=strlen("c:\dxx\samples\cmd\getstart_xcollection.xml");
         xmlobj.file_option=SQL_FILE_READ;
         SQL EXEC VALUES (:xmlFile) INTO (:xmlDoc);
         returnCode = 0;
         returnMsg[0] = '\0';
         collection_ind = 0;
         xmlobj_ind = 0;
         returnCode_ind = -1;
         returnMsg_ind = -1;

         /* Call the store procedure */
         EXEC SQL CALL db2xml.dxxInsertXML(:collection:collection_ind;
                     :xmlDoc:xmlDoc_ind,
                     :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Accès à une collection XML

Vous pouvez effectuer des opérations de mise à jour, de suppression, de recherche et d'extraction sur des collections XML. Toutefois, nous vous rappelons que les collections XML ont pour but de stocker ou d'insérer après extraction des données non balisées dans des tables de la base de données. Les données des tables existant dans la base de données sont entièrement indépendantes des documents XML entrants. Les opérations de mise à jour, de suppression et de recherche sont effectuées par des accès SQL normaux à ces tables. Si vous décomposez les données à partir de documents XML entrants, vous perdez tout document XML d'origine.

L'Extension XML permet d'effectuer des opérations sur les données d'une vue de collection XML. A l'aide des instructions SQL UPDATE et DELETE, vous

pouvez modifier les données entrant dans la composition des documents XML et, par conséquent, mettre à jour la collection XML.

Remarques :

- Pour mettre à jour un document, ne supprimez pas dans la table la ligne de clé primaire qui constitue la ligne de clé associée des autres tables de la collection. Lorsque vous supprimez la ligne de clé primaire et de clé associée, vous supprimez le document.
- Pour remplacer ou supprimer des éléments et des valeurs d'attributs, vous pouvez supprimer et insérer des lignes dans les tables de niveau inférieur, sans supprimer le document.
- Pour supprimer un document, supprimez la ligne qui compose le noeud d'élément supérieur indiqué dans la DAD.

Mise à jour de données dans une collection XML

L'Extension XML permet de mettre à jour des données non balisées stockées dans des tables de la collection XML. Par l'actualisation des valeurs contenues dans la table de la collection XML, vous mettez à jour le texte d'un élément XML ou la valeur d'un attribut XML. En outre, la mise à jour permet de supprimer une instance de donnée provenant d'un élément ou d'un attribut à occurrences multiples.

Dans le contexte SQL, la modification d'une valeur d'élément ou d'attribut et la suppression d'une instance d'élément ou d'attribut correspondent respectivement à une mise à jour et à une suppression. Dans le contexte XML, tant que le texte élémentaire ou la valeur d'attribut du noeud d'élément racine existe, le document XML est conservé et, par conséquent, il s'agit d'une mise à jour.

Exigences : Pour mettre à jour des données dans une collection XML, respectez les règles ci-après.

- Indiquez la relation clé primaire-associée établie entre les tables de la collection si elle s'applique aux tables existantes. Si tel n'est pas le cas, assurez-vous qu'une jointure peut être réalisée sur des colonnes.
- Incluez la condition de jointure indiquée dans le fichier DAD :
  - pour le mappage SQL, dans l'élément <SQL_stmt>,
  - pour le mappage du noeud RDB, dans l'élément RDB_node du noeud d'élément racine.

Mise à jour de valeurs d'éléments et d'attributs

Dans une collection XML, le texte élémentaire et la valeur d'attribut sont mappés vers les colonnes des tables de la base de données. Vous remplacez les données des colonnes par une mise à jour SQL classique, qu'il s'agisse de données existantes ou décomposées à partir de documents XML entrants.

Pour mettre à jour une valeur d'élément ou d'attribut, ajoutez une clause WHERE dans l'instruction SQL UPDATE qui contient la condition de jointure indiquée dans le fichier DAD.

Par exemple :

```
UPDATE SHIP_TAB
    set MODE = 'BOAT'
  WHERE MODE='AIR' AND PART_KEY in
    (SELECT PART_KEY from PART_TAB WHERE ORDER_KEY=68)
```

La valeur de l'élément <ShipMode> passe de AIR à BOAT dans la table SHIP_TAB dotée de la clé 68.

Suppression d'instances d'élément et d'attribut

Pour mettre à jour des documents XML composés par la suppression d'éléments ou d'attributs à occurrences multiples, supprimez la ligne qui contient la valeur de zone correspondant à la valeur d'élément ou d'attribut, à l'aide de la clause WHERE. Tant que vous ne supprimez pas la ligne qui contient les valeurs du noeud d'élément supérieur, la suppression de valeurs élémentaires est considérée comme une mise à jour du document XML.

Par exemple, dans l'instruction DELETE ci-après, vous supprimez l'élément <shipment> par l'indication d'une valeur unique prise par l'un de ses sous-éléments.

```
DELETE from SHIP_TAB
   WHERE DATE='1999-04-12'
```

L'indication de la valeur DATE supprime la ligne qui correspond à cette valeur. Le document composé qui contenait initialement deux éléments <shipment> n'en contient qu'un désormais.

Suppression d'un document XML dans une collection XML

Vous pouvez supprimer un document XML composé à partir d'une collection. Cela signifie que si vous disposez d'une collection XML composant plusieurs documents XML, vous pouvez supprimer l'un d'eux.

Pour ce faire, vous supprimez une ligne dans la table qui compose le noeud d'élément supérieur indiqué dans le fichier DAD. Cette table contient la clé primaire de la table de niveau supérieur et la clé associée des tables de niveau inférieur.

Par exemple, l'instruction DELETE ci-après indique la valeur de la colonne de clé primaire.

```
DELETE from order_tab
   WHERE order_key=1
```

ORDER_KEY représente la clé primaire de la table ORDER_TAB et le noeud d'élément supérieur lorsque le document XML est composé. La suppression de cette ligne supprime un document XML généré lors de la composition. Par conséquent, dans le contexte XML, un document XML est supprimé de la collection XML.

Extraction de documents XML d'une collection XML

L'extraction de documents XML d'une collection XML est identique à la composition de documents à partir de la collection.

Pour extraire des documents XML, utilisez la procédure mémorisée dxxRetrieveXML(). Pour consulter la syntaxe et des exemples, reportez-vous à la section «dxxRetrieveXML()» à la page 230.

Remarque sur le fichier DAD : Lorsque vous décomposez des documents XML en collection XML, vous pouvez perdre l'ordre des éléments et des valeurs d'attributs à occurrences multiples, sauf si vous le précisez dans le fichier DAD. Pour conserver cet ordre, vous devez utiliser le schéma de mappage du noeud RDB. Ce schéma de mappage permet d'indiquer un attribut orderBy pour la table qui contient l'élément racine dans le noeud RDB associé.

Recherche dans une collection XML

Cette section décrit la recherche dans une collection XML orientée vers les objectifs suivants :

- **Génération de documents XML sur des critères de recherche :**

  Cette tâche correspond en fait à la composition à l'aide d'une condition. Vous pouvez préciser les critères de recherche suivants :

  – condition dans les éléments text_node et attribute_node du fichier DAD,

  – paramètre *overwrite* avec les procédures mémorisées dxxGenXML() et dxxRetrieveXML().

  Par exemple, si vous avez activé la collection XML sales_ord à l'aide du fichier DAD `order.dad`, mais que vous ne voulez pas remplacer le prix avec les données de formulaire provenant d'une page Web, vous pouvez remplacer la valeur de l'élément DAD <SQL_stmt> comme suit :

```
EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;
       ...

  EXEC SQL END DECLARE SECTION;

         float    price_value;

         /* create table */
```

```
                    EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

                /* initialize host variable and indicators */
                strcpy(collection,"sales_ord");
                strcpy(result_tab,"xml_order_tab");
                overrideType = SQL_OVERRIDE;
                max_row = 20;
                num_row = 0;
                returnCode = 0;
                msg_txt[0] = '\0';
                override_ind = 0;
                overrideType_ind = 0;
                rtab_ind = 0;
                maxrow_ind = 0;
                numrow_ind = -1;
                returnCode_ind = -1;
                returnMsg_ind = -1;

                /* get the price_value from some place, such as form data */
                price_value = 1000.00         /* for example*/

                /* specify the overwrite */
                sprintf(overwrite,
                       "SELECT o.order_key, customer, p.part_key, quantity, price,
                              tax, ship_id, date, mode
                FROM order_tab o, part_tab p,
    table (select substr(char(timestamp(generate_unique())),16)
                           as ship_id, date, mode from ship_tab)as s
                       WHERE p.price > %d and s.date >'1996-06-01' AND
                           p.order_key = o.order_key and s.part_key = p.part_key",
                           price_value);

                /* Call the store procedure */
               EXEC SQL CALL db2xml.dxxRetrieve(:collection:collection_ind,
                       :result_tab:rtab_ind,
                       :overrideType:overrideType_ind,:overwrite:overwrite_ind,
                       :max_row:maxrow_ind,:num_row:numrow_ind,
                             :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

La condition de prix > 2500,00 dans order.dad est remplacée par price > ?, où ? dépend de la variable d'entrée *price_value*.

- **Recherche de données XML décomposées :**

  Vous pouvez effectuer des recherches dans les tables de la collection par des requêtes SQL classiques. Vous pouvez réaliser une jointure des tables de la collection ou lancer des sous-requêtes, puis effectuer une recherche structurelle sur les colonnes de texte. Avec les résultats de la recherche structurelle, vous pouvez appliquer ces données pour extraire ou générer le document XML indiqué.

Partie 4. Guide de référence

Cette partie indique la syntaxe de la commande d'administration de l'Extension XML, des types UDT, des fonctions UDF et des procédures mémorisées. Elle répertorie également les messages d'erreur pour la résolution des incidents.

Chapitre 7. Commande d'administration de l'Extension XML : dxxadm

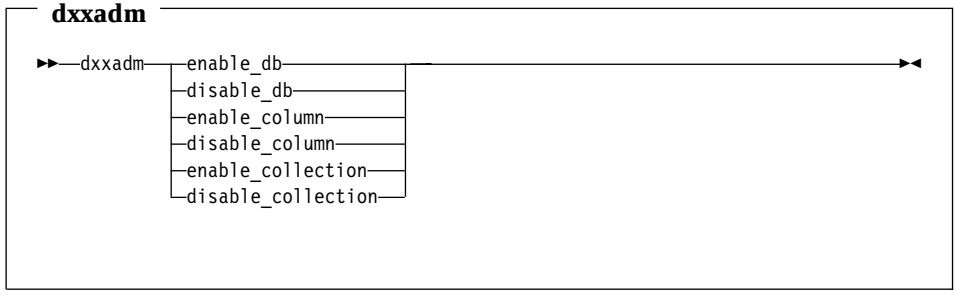
L'Extension XML comporte une commande d'administration, **dxxadm**, qui permet d'effectuer les tâches d'administration ci-après.Vous appelez la commande **dxxadm** suivie de diverses options.

- Activation ou désactivation d'une base de données pour l'Extension XML
- Activation ou désactivation d'une colonne XML
- Activation ou désactivation d'une collection XML

Vous pouvez également exécuter chaque tâche d'administration à l'aide de l'assistant d'administration ou des procédures mémorisées de l'Extension XML.

Syntaxe évoluée

Le diagramme de syntaxe ci-après indique la syntaxe évoluée de la commande **dxxadm**. Les options sont décrites dans les sections suivantes.



Options de la commande

Les sections suivantes décrivent les options **dxxadm** mises à la disposition des programmeurs système :

- «enable_db» à la page 162
- «disable_db» à la page 164
- «enable_column» à la page 166
- «disable_column» à la page 168
- «enable_collection» à la page 170
- «disable_collection» à la page 172

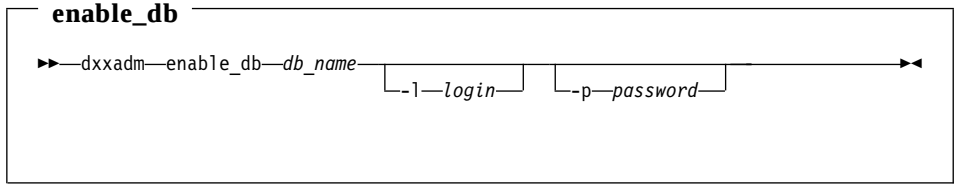
enable_db

Objet

Connecte à une base de données et l'active pour qu'elle puisse être utilisée avec l'Extension XML. Une fois la base de données activée, l'Extension XML crée les objets suivants :

- Types définis par l'utilisateur (UDT).
- Fonctions définies par l'utilisateur (UDF).
- Table de référence des DTD (DTD_REF), pour le stockage des définitions de types de documents et des informations associées. Pour obtenir une description complète de la table DTD_REF, reportez-vous à la section «Table DTD_REF» à la page 239.
- Table d'usage (XML_USAGE), pour le stockage des informations communes à chaque colonne et collection XML. Pour obtenir une description complète de la table XML_USAGE, reportez-vous à la section «Table XML_USAGE» à la page 240.

Syntaxe



Paramètres

Tableau 14. Paramètres enable_db

Paramètre	Description
db_name	Nom de la base de données où résident les données XML.
-l *login*	ID utilisateur (facultatif) sous lequel vous vous êtes connecté à la base de données, si vous en avez indiqué un. Si vous laissez cette zone à blanc, l'Extension XML utilise l'ID utilisateur en cours.
-p *password*	Mot de passe (facultatif) sous lequel vous vous êtes connecté à la base de données, si vous en avez indiqué un. Si vous laissez cette zone à blanc, l'Extension XML utilise le mot de passe en cours.

Exemples

Dans l'exemple ci-après, vous activez la base de données SALES_DB.

```
dxxadm enable_db SALES_DB
```

disable_db

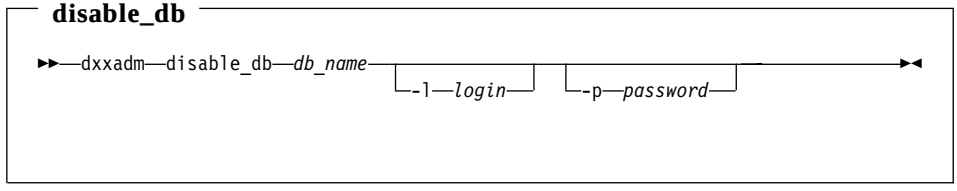
Objet

Connecte à la base de données activée pour XML et la désactive. La base de données désactivée ne peut plus être utilisée par l'Extension XML. Une fois la base de données désactivée, l'Extension XML retire les objets suivants :

- Types définis par l'utilisateur (UDT)
- Fonctions définies par l'utilisateur (UDF)
- Table de référence des DTD (DTD_REF), pour le stockage des définitions de types de documents et des informations associées. Pour obtenir une description complète de la table DTD_REF, reportez-vous à la section «Table DTD_REF» à la page 239.
- Table d'usage (XML_USAGE), pour le stockage des informations communes à chaque colonne et collection XML. Pour obtenir une description complète de la table XML_USAGE, reportez-vous à la section «Table XML_USAGE» à la page 240.

Important : Pour désactiver une base de données, vous devez d'abord désactiver toutes les colonnes et collections XML associées.

Syntaxe



Paramètres

Tableau 15. Paramètres disable_db

Paramètre	Description
db_name	Nom de la base de données où résident les données XML.
-l *login*	ID utilisateur (facultatif) sous lequel vous vous êtes connecté à la base de données, si vous en avez indiqué un. Si vous laissez cette zone à blanc, l'Extension XML utilise l'ID utilisateur en cours.
-p *password*	Mot de passe (facultatif) sous lequel vous vous êtes connecté à la base de données, si vous en avez indiqué un. Si vous laissez cette zone à blanc, l'Extension XML utilise le mot de passe en cours.

Exemples

Dans l'exemple ci-après, vous désactivez la base de données SALES_DB.

```
dxxadm disable_db SALES_DB
```

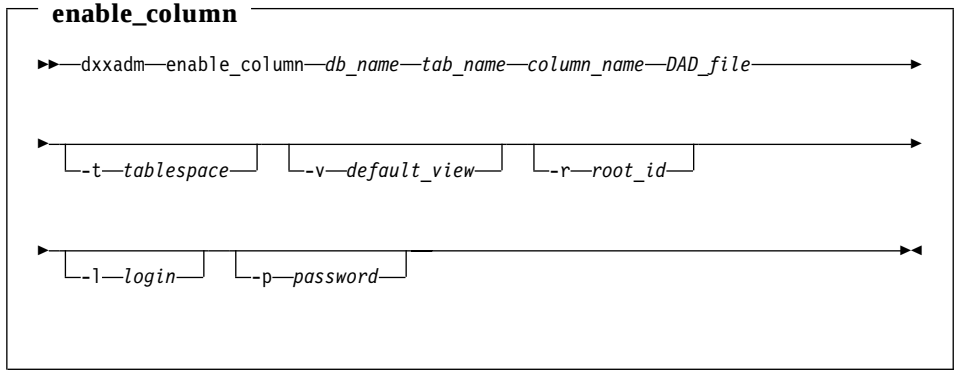
enable_column

Objet

Connecte à une base de données et active une colonne XML pour qu'elle puisse contenir les types UDT de l'Extension XML. Lors de l'activation d'une colonne, DB2 Extension XML exécute les tâches suivantes :

- Il détermine si la table XML possède une clé primaire. Dans la négative, il la modifie et ajoute une colonne DXXROOT_ID.
- Il crée les tables annexes indiquées dans le fichier DAD, avec une colonne contenant un identificateur unique pour chaque ligne de la table XML. Cette colonne correspond à l'identificateur racine (*root_id*) indiqué par l'utilisateur, ou à l'identificateur DXXROOT_ID nommé par l'Extension XML.
- Il crée une vue par défaut de la table XML et de ses tables annexes, en lui attribuant le nom éventuellement indiqué.

Syntaxe



Paramètres

Tableau 16. Paramètres enable_column

Paramètre	Description
db_name	Nom de la base de données où résident les données XML.
tab_name	Nom de la table où réside la colonne XML.
column_name	Nom de la colonne XML.
DAD_file	Nom du fichier DAD qui mappe le document XML vers la colonne et les tables annexes XML.

Tableau 16. Paramètres enable_column (suite)

Paramètre	Description
-t *tablespace*	Espace table (facultatif) contenant les tables annexes associées à la colonne XML. Si vous laissez cette zone à blanc, l'Extension XML utilise un espace table par défaut.
-v *default_view*	Nom de la vue par défaut (facultative) réalisant la jointure de la colonne XML et des tables annexes.
-r *root_id*	Nom de la clé primaire figurant dans la table de la colonne XML, à utiliser comme identificateur racine (root_id) pour les tables annexes. L'identificateur racine est facultatif.
-l *login*	ID utilisateur (facultatif) sous lequel vous vous êtes connecté à la base de données, si vous en avez indiqué un. Si vous laissez cette zone à blanc, l'Extension XML utilise l'ID utilisateur en cours.
-p *password*	Mot de passe (facultatif) sous lequel vous vous êtes connecté à la base de données, si vous en avez indiqué un. Si vous laissez cette zone à blanc, l'Extension XML utilise le mot de passe en cours.

Exemples

Dans l'exemple ci-après, vous activez une colonne XML.

```
dxxadm enable_column SALES_DB SALES_TAB ORDER -v sales_order_view -r INVOICE_NUMBER
```

disable_column

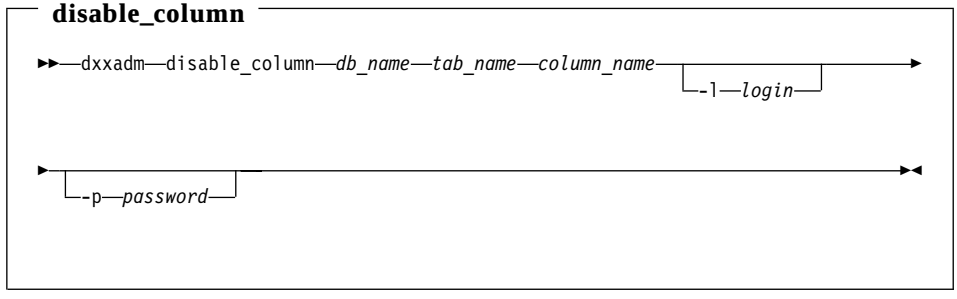
Objet

Connecte à une base de données et désactive la colonne activée pour XML. Lorsque la colonne XML est désactivée, elle ne peut plus contenir de donnée de type XML. Une fois la colonne XML désactivée, les actions suivantes sont exécutées :

- L'entrée relative à l'usage de la colonne XML est supprimée de la table XML_USAGE.
- Le compteur d'opérations USAGE_COUNT est décrémenté dans la table DTD_REF.
- Tous les déclencheurs associés à cette colonne sont retirés.
- Toutes les tables annexes associées à cette colonne sont retirées.

Important : Pour pouvoir retirer une table XML, vous devez d'abord désactiver la colonne XML qu'elle contient. Si vous ne le faites pas, l'Extension XML conserve les tables annexes créées et l'entrée de colonne XML dans la table XML_USAGE.

Syntaxe



Paramètres

Tableau 17. Paramètres disable_column

Paramètre	Description
db_name	Nom de la base de données où résident les données.
tab_name	Nom de la table où réside la colonne XML.
column_name	Nom de la colonne XML.

Tableau 17. Paramètres disable_column (suite)

Paramètre	Description
-l *login*	ID utilisateur (facultatif) sous lequel vous vous êtes connecté à la base de données, si vous en avez indiqué un. Si vous laissez cette zone à blanc, l'Extension XML utilise l'ID utilisateur en cours.
-p *password*	Mot de passe (facultatif) sous lequel vous vous êtes connecté à la base de données, si vous en avez indiqué un. Si vous laissez cette zone à blanc, l'Extension XML utilise le mot de passe en cours.

Exemples

Dans l'exemple ci-après, vous désactivez une colonne XML.

```
dxxadm disable_column SALES_DB SALES_TAB ORDER
```

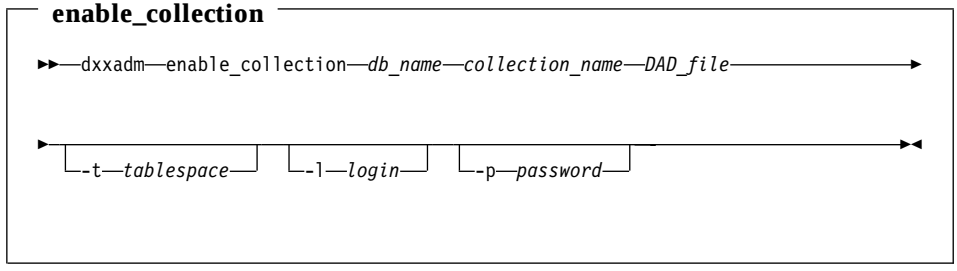
enable_collection

Objet

Connecte à une base de données et active une collection XML en fonction de la DAD indiquée. Lors de l'activation d'une collection, DB2 Extension XML exécute les tâches suivantes :

- Il crée une entrée relative à l'usage de la collection XML dans la table XML_USAGE.
- Pour le mappage du noeud RDB, il crée les tables de collection indiquées dans la DAD si elles n'existent pas dans la base de données.

Syntaxe



Paramètres

Tableau 18. Paramètres enable_collection

Paramètre	Description
db_name	Nom de la base de données où résident les données.
collection_name	Nom de la collection XML.
DAD_file	Nom du fichier DAD qui mappe le document XML vers les tables relationnelles de la collection.
-t *tablespace*	Nom de l'espace table (facultatif) associé à la collection. Si vous laissez cette zone à blanc, l'Extension XML utilise un espace table par défaut.
-l *login*	ID utilisateur (facultatif) sous lequel vous vous êtes connecté à la base de données, si vous en avez indiqué un. Si vous laissez cette zone à blanc, l'Extension XML utilise l'ID utilisateur en cours.

Tableau 18. Paramètres enable_collection (suite)

Paramètre	Description
-p *password*	Mot de passe (facultatif) sous lequel vous vous êtes connecté à la base de données, si vous en avez indiqué un. Si vous laissez cette zone à blanc, l'Extension XML utilise le mot de passe en cours.

Exemples

Dans l'exemple ci-après, vous activez une collection XML.

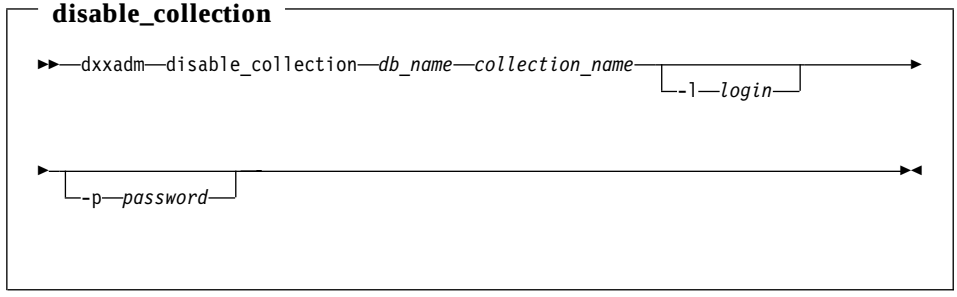
```
dxxadm enable_collection SALES_DB sales_ord getstart_xcollection.dad -t orderspace
```

disable_collection

Objet

Connecte à une base de données et désactive la collection activée pour XML. Le nom de la collection ne peut plus être utilisé dans les procédures mémorisées de composition (dxxRetrieveXML) et de décomposition (dxxInsertXML). Lorsqu'une collection XML est désactivée, l'entrée de collection associée est supprimée de la table XML_USAGE. Notez que la désactivation de la collection n'entraîne pas le retrait des tables correspondantes créées pendant la phase d'activation de la collection.

Syntaxe



Paramètres

Tableau 19. Paramètres disable_collection

Paramètre	Description
db_name	Nom de la base de données où résident les données.
collection_name	Nom de la collection XML.
-l *login*	ID utilisateur (facultatif) sous lequel vous vous êtes connecté à la base de données, si vous en avez indiqué un. Si vous laissez cette zone à blanc, l'Extension XML utilise l'ID utilisateur en cours.
-p *password*	Mot de passe (facultatif) sous lequel vous vous êtes connecté à la base de données, si vous en avez indiqué un. Si vous laissez cette zone à blanc, l'Extension XML utilise le mot de passe en cours.

Exemples

Dans l'exemple ci-après, vous désactivez une collection XML.

```
dxxadm disable_collection SALES_DB sales_ord
```

Chapitre 8. Types de données UDT de l'Extension XML

Les types de données UDT de l'Extension XML sont utilisés pour les colonnes et les collections XML. Tous les types de données UDT ont pour nom de schéma db2xml. L'Extension XML crée des types de données UDT pour l'archivage et la récupération de documents XML. Le tableau 20 offre une présentation des types de données UDT.

Tableau 20. Types de données UDT de l'Extension XML

Colonne UDT	Type de source de données	Description
XMLVARCHAR	VARCHAR(*varchar_len*)	Archive un document XML entier en tant que VARCHAR dans DB2.
XMLCLOB	CLOB(*clob_len*)	Archive un document XML entier en tant qu'objet CLOB (Character Large Object) dans DB2.
XMLFILE	VARCHAR(512)	Spécifie le nom de fichier du serveur de fichiers local. Si XMLFILE est spécifié pour la colonne XML, l'Extension XML archive le document XML dans un fichier de serveur externe. Il est impossible d'activer l'Extension Texte avec XMLFILE. Il vous incombe d'assurer l'intégrité entre le contenu du fichier et DB2 ainsi que l'intégrité de la table annexe créée pour l'indexation.

Où *varchar_len* et *clob_len* relèvent du système d'exploitation.

Pour DB2 UDB, *varchar_len* = 3 ko et *clob_len* = 2 go.

Ces types de données UDT sont utilisés uniquement pour spécifier les types de colonnes d'application ; ils ne s'appliquent pas aux tables annexes créées par l'Extension XML.

Chapitre 9. Fonctions UDF de l'Extension XML

L'Extension XML comprend des fonctions permettant l'archivage, la récupération, la recherche et la mise à jour de documents XML ainsi que l'extraction d'éléments ou d'attributs XML. Utilisez les fonctions UDF de XML pour les colonnes XML mais ne les employez pas pour les collections XML. Toutes les fonctions UDF ont pour nom de schéma db2xml, qui peut être omis devant les fonctions UDF.

Les quatre types de fonctions de l'Extension XML sont : les fonctions d'archivage, les fonctions de récupération, les fonctions d'extraction et une fonction de mise à jour.

Fonctions d'archivage
Permettent d'insérer les documents XML dans une base de données DB2. Pour connaître la syntaxe et obtenir des exemples, reportez-vous à «Fonctions d'archivage» à la page 176.

Fonctions de récupération
Permettent de prélever des documents XML dans des colonnes XML d'une base de données DB2. Pour connaître la syntaxe et obtenir des exemples, reportez-vous à «Fonctions de récupération» à la page 182.

Fonctions d'extraction
Les fonctions d'extraction permettent d'extraire et de convertir le contenu d'un élément ou la valeur d'un attribut d'un document XML au type de données spécifié par le nom de la fonction. L'Extension XML comporte un ensemble de fonctions d'extraction pour les différents types de données SQL. Pour connaître la syntaxe et obtenir des exemples, reportez-vous à «Fonctions d'extraction» à la page 189.

Fonction de mise à jour
La fonction de mise à jour Update() modifie le contenu de l'élément ou la valeur de l'attribut et renvoie une copie du document XML avec une valeur mise à jour spécifiée par le chemin d'emplacement. La fonction Update() autorise le programmeur d'applications à spécifier l'élément ou l'attribut à mettre à jour. Pour connaître la syntaxe et obtenir des exemples, reportez-vous à «Fonction de mise à jour» à la page 206.

Le tableau 21 à la page 176 présente un récapitulatif des fonctions de l'Extension XML.

Tableau 21. Fonctions UDF de l'Extension XML

Type	Fonction
Fonctions d'archivage	XMLVarcharFromFile()
	XMLCLOBFromFile()
	XMLFileFromVarchar()
	XMLFileFromCLOB()
Fonctions de récupération	Content() : récupération à partir de XMLFile vers un objet CLOB
	Content() : récupération à partir de XMLVarchar vers un fichier de serveur externe
	Content() : récupération à partir de XMLCLOB vers un fichier de serveur externe
Fonctions d'extraction	extractInteger() et extractIntegers()
	extractSmallint() et extractSmallints()
	extractDouble() et extractDoubles()
	extractReal() et extractReals()
	extractChar() et extractChars()
	extractVarchar() et extractVarchars()
	extractCLOB() et extractCLOBs()
	extractDate() et extractDates()
	extractTime() et extractTimes()
	extractTimestamp() et extractTimestamps()
Fonction de mise à jour	Update()

Lorsque vous utilisez des marqueurs de paramètres dans des fonctions UDF, en raison d'une restriction JDBC, le marqueur de paramètre de la fonction UDF doit être transtypé vers le type de données de la colonne dans laquelle les données renvoyées seront insérées. Pour savoir comment transtyper les marqueurs de paramètres, reportez-vous à la section «Limitations relatives à l'appel de fonctions à partir de JDBC» à la page 140.

Fonctions d'archivage

Les fonctions d'archivage permettent d'insérer des documents XML dans une base de données DB2. Vous pouvez utiliser les fonctions de transtypage par défaut d'une fonction UDT directement dans les instructions INSERT ou SELECT, comme indiqué dans la section «Stockage des données» à la page 124. De plus, l'Extension XML comporte des fonctions UDF permettant

d’accéder à des documents XML à partir de sources différentes du type UDT de base, puis de les convertir au format du type UDT voulu.

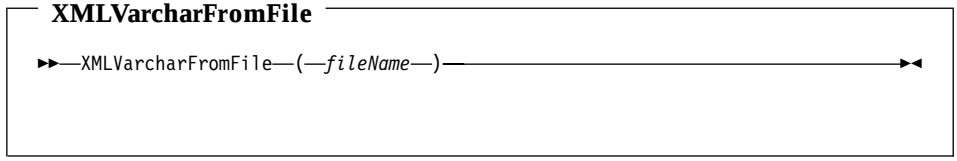
L’Extension XML fournit les fonctions d’archivage suivantes :

- «XMLVarcharFromFile()» à la page 178
- «XMLCLOBFromFile()» à la page 179
- «XMLFileFromVarchar()» à la page 180
- «XMLFileFromCLOB()» à la page 181

XMLVarcharFromFile()

Objet
Lit un document XML à partir d'un fichier de serveur et renvoie un type de document XMLVARCHAR.

Syntaxe



Paramètres

Tableau 22. Paramètre XMLVarcharFromFile

Paramètre	Type de données	Description
fileName	VARCHAR(512)	Nom qualifié complet du fichier de serveur.

Type de données renvoyé
XMLVARCHAR

Exemple
L'exemple suivant illustre la lecture d'un document XML à partir d'un fichier de serveur et son insertion dans une colonne XML en tant que valeur de type XMLVARCHAR.

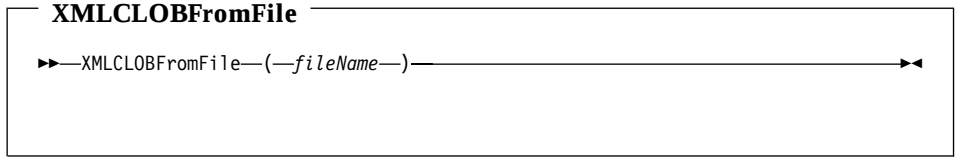
```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
VALUES('1234', 'Sriram Srinivasan',
           XMLVarcharFromFile('c:\dxx\samples\cmd\getstart.xml '))
```

Dans cet exemple, un enregistrement est inséré dans la table SALES_TAB. La fonction XMLVarcharFromFile() importe le document XML d'un fichier de DB2 et l'archive en tant que XMLVARCHAR.

XMLCLOBFromFile()

Objet
Lit un document XML à partir d'un fichier de serveur et renvoie un type de document XMLCLOB.

Syntaxe



Paramètres

Tableau 23. Paramètre XMLCLOBFromFile

Paramètre	Type de données	Description
fileName	VARCHAR(512)	Nom qualifié complet du fichier de serveur.

Type de données renvoyé
XMLCLOB as LOCATOR

Exemple
L'exemple suivant illustre la lecture d'un document XML à partir d'un fichier de serveur et son insertion dans une colonne XML en tant que valeur de type XMLCLOB.

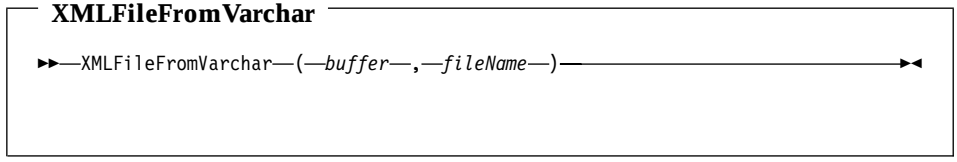
```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
VALUES('1234', 'Sriram Srinivasan',
           XMLCLOBFromFile('c:\dxx\samples\cmd\getstart.xml'))
```

La colonne ORDER dans la table SALES_TAB est définie comme une valeur de type XMLCLOB. L'exemple précédent illustre le mode d'insertion de la colonne ORDER dans la table SALES_TAB.

XMLFileFromVarchar()

Objet
Lit un document XML en mémoire en tant que VARCHAR, l'écrit dans un fichier de serveur externe et renvoie son nom de fichier et son chemin sous la forme d'une valeur de type XMLFILE.

Syntaxe



Paramètres

Tableau 24. Paramètres XMLFileFromVarchar

Paramètre	Type de données	Description
buffer	VARCHAR(3K)	Mémoire tampon.
fileName	VARCHAR(512)	Nom qualifié complet du fichier de serveur.

Type de données renvoyé
XMLFILE

Exemple
Les exemples suivants illustrent la lecture d'un document XML en mémoire en tant que VARCHAR, son écriture dans un fichier de serveur externe et l'insertion de son nom de fichier et de son chemin sous la forme d'une valeur de type XMLFILE dans une colonne XML.

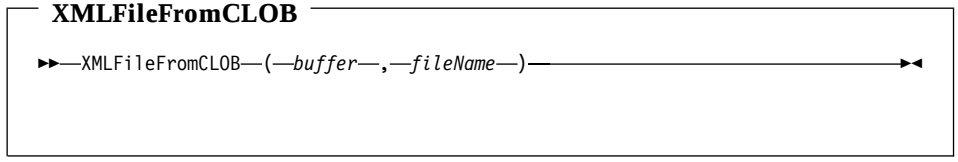
```
EXEC SQL BEGIN DECLARE SECTION;
      struct { short len; char data[3000]; } xml_buff;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
VALUES('1234', 'Sriram Srinivasan',
       XMLFileFromVarchar(:xml_buf, 'c:\dxx\samples\cmd\getstart.xml '))
```

La colonne ORDER dans la table SALES_TAB est définie comme une valeur de type XMLFILE. L'exemple précédent montre que si vous avez un document XML dans la mémoire tampon, vous pouvez l'archiver dans un fichier de serveur.

XMLFileFromCLOB()

Objet
Lit un document XML en tant que releveur de coordonnées CLOB, l'écrit dans un fichier de serveur externe et renvoie son nom de fichier et son chemin sous la forme d'une valeur de type XMLFILE.

Syntaxe



Paramètres

Tableau 25. Paramètres XMLFileFromCLOB()

Paramètres	Type de données	Description
buffer	CLOB as LOCATOR	Mémoire tampon contenant le document XML.
fileName	VARCHAR(512)	Nom qualifié complet du fichier de serveur.

Type de données renvoyé
XMLFILE

Exemple
L'exemple suivant illustre la lecture d'un document XML en tant que releveur de coordonnées CLOB, son écriture dans un fichier de serveur externe et l'insertion de son nom de fichier et de son chemin sous la forme d'une valeur de type XMLFILE dans une colonne XML.

```
EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE IS CLOB_LOCATOR xml_buff;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
VALUES('1234', 'Sriram Srinivasan',
           XMLFileFromCLOB(:xml_buf, 'c:\dxx\samples\cmd\getstart.xml '))
```

La colonne ORDER dans la table SALES_TAB est définie comme une valeur de type XMLFILE. Si un document XML figure en mémoire tampon, vous pouvez l'archiver dans un fichier de serveur.

Fonctions de récupération

Vous pouvez utiliser les fonctions de transtypage par défaut pour convertir un type UDT XML en type de données de base, comme indiqué dans la section «Extraction d'un document entier» à la page 127. L'Extension XML fournit également une *fonction multi-référencée* Content(), utilisée pour les opérations de récupération.

La fonction Content() permet les types de récupération suivants :

- **A partir d'une mémoire externe sur le serveur vers une variable hôte sur le client.**

  Vous pouvez utiliser Content() pour récupérer un document XML dans une mémoire tampon lorsqu'il est archivé sous forme de fichier de serveur externe. Si nécessaire, aidez-vous de la section «Content(): récupération à partir de XMLFILE vers un objet CLOB» à la page 183 dans ce but.

- **Récupération à partir d'une mémoire interne vers un fichier de serveur externe**

  Vous pouvez également utiliser Content() pour récupérer un document XML stocké dans DB2 et l'archiver dans un fichier de serveur figurant dans le système de fichiers du serveur DB2. Les fonctions Content() suivantes sont utilisées pour archiver des données sur des fichiers de serveur externes :

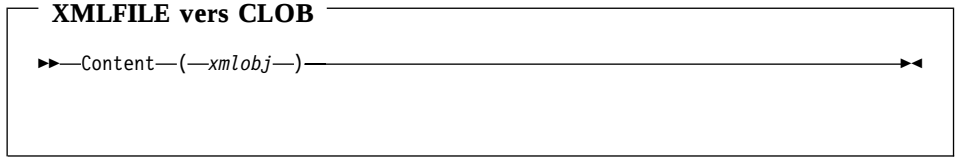
  – «Content() : récupération à partir de XMLVARCHAR vers un fichier de serveur externe» à la page 185
  – «Content() : récupération à partir de XMLCLOB vers un fichier de serveur externe» à la page 187

Les exemples de la section suivante considèrent que vous utilisez le shell de commandes DB2, dans lequel il n'est pas nécessaire de commencer les commandes par «DB2».

Content(): récupération à partir de XMLFILE vers un objet CLOB

Objet
Récupère les données d'un fichier de serveur et les archive dans un releveur de coordonnées CLOB LOCATOR.

Syntaxe



Paramètres

Tableau 26. Paramètre XMLFILE vers un objet CLOB

Paramètre	Type de données	Description
xmlobj	XMLFILE	Document XML.

Type de données renvoyé
CLOB (*clob_len*) as LOCATOR

clob_len pour DB2 UDB est égal à 2 Go.

Exemple
L'exemple suivant illustre la récupération de données à partir d'un fichier de serveur et leur archivage dans un releveur de coordonnées CLOB.

```
EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE IS CLOB_LOCATOR xml_buff;
EXEC SQL END DECLARE SECTION;

EXEC SQL CONNECT TO SALES_DB

EXEC SQL DECLARE c1 CURSOR FOR

     SELECT Content(order) from sales_tab
     WHERE sales_person = 'Sriram Srinivasan'

EXEC SQL OPEN c1;

do {
   EXEC SQL FETCH c1 INTO :xml_buff;
   if (SQLCODE != 0) {
      break;
      }
   else {
     /* do with the XML doc in buffer */
```

```
    }
  }

EXEC SQL CLOSE c1;

EXEC SQL CONNECT RESET;
```

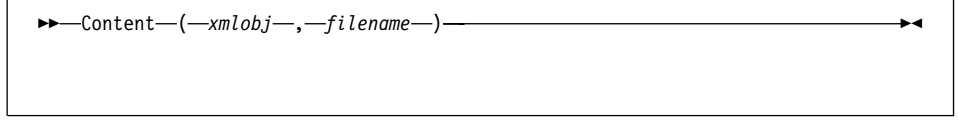
La colonne ORDER de la table SALES_TAB est de type XMLFILE, donc la fonction UDF Content() récupère des données dans un fichier de serveur et les archive dans un releveur de coordonnées CLOB.

Content() : récupération à partir de XMLVARCHAR vers un fichier de serveur externe

Objet
Récupère le contenu XML stocké sous le type XMLVARCHAR et l'archive dans un fichier de serveur externe.

Syntaxe

XMLVARCHAR vers un fichier de serveur externe



Important : Si un fichier du nom indiqué existe déjà, la fonction Content() remplace son contenu.

Remarque :

Paramètres

Tableau 27. Paramètres XMLVarchar vers un fichier de serveur externe

Paramètre	Type de données	Description
xmlobj	XMLVARCHAR	Document XML.
filename	VARCHAR(512)	Nom qualifié complet du fichier de serveur.

Type de données renvoyé
VARCHAR(512)

Exemple
L'exemple suivant illustre la récupération du contenu XML stocké sous le type XMLVARCHAR et son archivage dans un fichier de serveur externe.

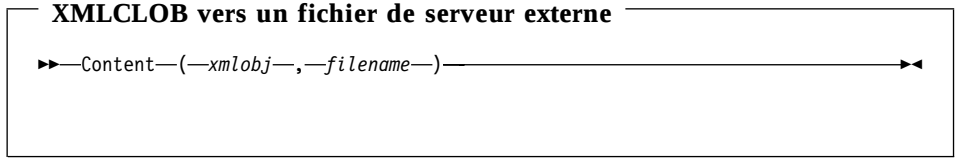
```
CREATE table app1 (id int NOT NULL, order db2xml.XMLVarchar);
INSERT into app1 values (1, '<?xml version="1.0"?>
   <!DOCTYPE SYSTEM c:\dxx\samples\dtd\getstart.dtd"->
    <Order key="1">
      <Customer>
    <Name>American Motors</Name>
          <Email>parts@am.com</Email>
      </Customer>
      <Part color="black">
          <key>68</key>
          <Quantity>36</Quantity>
          <ExtendedPrice>34850.16</ExtendedPrice>
```

```
            <Tax>6.000000e-02</Tax>
            <Shipment>
                <ShipDate>1998-08-19</ShipDate>
                <ShipMode>AIR </ShipMode>
            </Shipment>
            <Shipment>
                <ShipDate>1998-08-19</ShipDate>
                <ShipMode>BOAT </ShipMode>
            </Shipment>
        </Order>');

SELECT db2xml.Content(order, 'c:\dxx\samples\cmd\getstart_.dad')
  from app1 where ID=1;
```

Content() : récupération à partir de XMLCLOB vers un fichier de serveur externe

Objet
Récupère le contenu XML stocké sous le type XMLCLOB et l'archive dans un fichier de serveur externe.

Syntaxe



Important : Si un fichier du nom indiqué existe déjà, la fonction Content() remplace son contenu.

Paramètres

Tableau 28. Paramètres XMLCLOB vers un fichier de serveur externe

Paramètre	Type de données	Description
xmlobj	XMLCLOB as LOCATOR	Document XML.
filename	VARCHAR(512)	Nom qualifié complet du fichier de serveur.

Type de données renvoyé
VARCHAR(512)

Exemple
L'exemple suivant illustre la récupération du contenu XML stocké sous le type XMLCLOB et son archivage dans un fichier de serveur externe.

```
CREATE table app1 (id int NOT NULL, order db2xml.XMLCLOB not logged);

INSERT into app1 values (1, '<?xml version="1.0"?>
  <!DOCTYPE SYSTEM c:\dxx\samples\dtd\getstart.dtd"->
    <Order key="1">
      <Customer>
    <Name>American Motors</Name>
          <Email>parts@am.com</Email>
      </Customer>
      <Part color="black">
          <key>68</key>
          <Quantity>36</Quantity>
          <ExtendedPrice>34850.16</ExtendedPrice>
          <Tax>6.000000e-02</Tax>
```

```
            <Shipment>
                <ShipDate>1998-08-19</ShipDate>
                <ShipMode>AIR </ShipMode>
            </Shipment>
            <Shipment>
                <ShipDate>1998-08-19</ShipDate>
                <ShipMode>BOAT </ShipMode>
            </Shipment>
        </Order>');

SELECT db2xml.Content(order, 'c:\dxx\samples\cmd\getstart.xml ')
from app1 where ID=1;
```

Fonctions d'extraction

Les fonctions d'extraction extraient le contenu de l'élément ou la valeur de l'attribut d'un document XML et renvoie les types de données SQL demandés. L'Extension XML fournit un ensemble de fonctions d'extraction pour les différents types de données SQL. Les fonctions d'extraction comprennent deux paramètres d'entrée. Le premier paramètre est le type UDT Extension XML, qui peut être un des types UDT XML. Le second paramètre est le chemin d'emplacement qui indique l'élément ou l'attribut XML. Chaque fonction d'extraction renvoie la valeur ou le contenu spécifié par le chemin d'emplacement.

Certaines valeurs d'éléments ou d'attributs ayant plusieurs occurrences, les fonctions d'extraction renvoient une valeur scalaire ou une valeur de table, cette dernière étant appelée fonction de table.

L'Extension XML fournit les fonctions d'extraction suivantes :

- «extractInteger() et extractIntegers()» à la page 190
- «extractSmallint() et extractSmallints()» à la page 192
- «extractDouble() et extractDoubles()» à la page 193
- «extractReal() et extractReals()» à la page 195
- «extractChar() et extractChars()» à la page 196
- «extractVarchar() et extractVarchars()» à la page 197
- «extractCLOB() et extractCLOBs()» à la page 199
- «extractDate() et extractDates()» à la page 201
- «extractTime() et extractTimes()» à la page 202
- «extractTimestamp() et extractTimestamps()» à la page 204

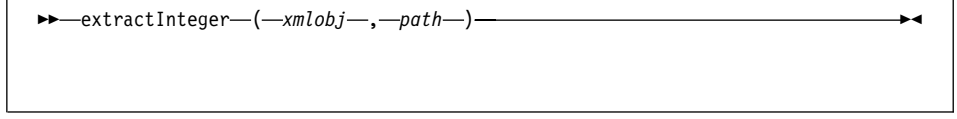
Les exemples de la section suivante considèrent que vous utilisez le shell de commandes DB2, dans lequel il n'est pas nécessaire de commencer les commandes par «DB2».

extractInteger() et extractIntegers()

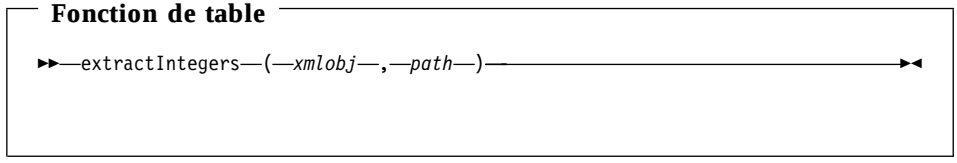
Objet
Extrait le contenu de l'élément ou la valeur de l'attribut d'un document XML et renvoie des données de type INTEGER.

Syntaxe

Fonction scalaire



Fonction de table



Paramètres

Tableau 29. Paramètres des fonctions extractInteger et extractIntegers

Paramètre	Type de données	Description
xmlobj	XMLVARCHAR, XMLFILE ou XMLCLOB	Nom de la colonne.
path	VARCHAR	Chemin d'emplacement de l'élément ou de l'attribut.

Type de données renvoyé
INTEGER

Nom de colonne renvoyé (fonction de table)
returnedInteger

Exemple
Exemple de fonction scalaire :

Dans l'exemple suivant, une seule valeur est renvoyée lorsque l'attribut key est égal à 1. La valeur extraite est de type INTEGER et est automatiquement convertie en valeur de type DECIMAL.

```
CREATE TABLE t1(key decimal(3,2));
INSERT into t1 values
SELECT * from table(db2xml.extractInteger(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/[@key="1"]'));
SELECT * from t1;
```

Exemple de fonction de table :

Dans l'exemple suivant, chaque valeur de clé de SALES_ORDER est extraite sous forme de valeur de type INTEGER :

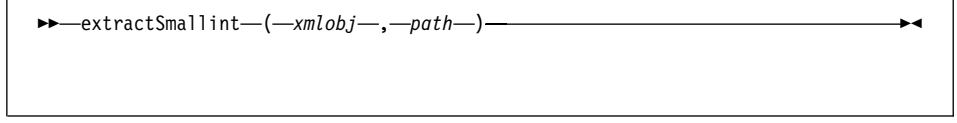
```
SELECT * from table(db2xml.extractIntegers(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/@key')) as x;
```

extractSmallint() et extractSmallints()

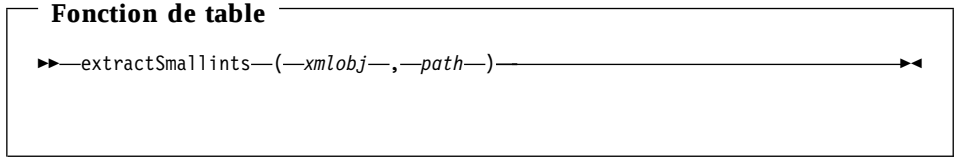
Objet
Extrait le contenu de l'élément ou la valeur de l'attribut d'un document XML et renvoie des données de type SMALLINT.

Syntaxe

Fonction scalaire



Fonction de table



Paramètres

Tableau 30. Paramètres des fonctions extractSmallint et extractSmallints

Paramètre	Type de données	Description
xmlobj	XMLVARCHAR, XMLFILE ou XMLCLOB	Nom de la colonne.
path	VARCHAR	Chemin d'emplacement de l'élément ou de l'attribut.

Type de données renvoyé
SMALLINT

Nom de colonne renvoyé (fonction de table)
returnedSmallint

Exemple
Dans l'exemple suivant, la quantité (Quantity) indiquée sur les bons de commande clients est extraite en tant que valeur SMALLINT.

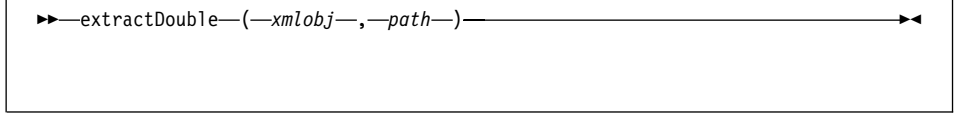
```
SELECT * from table(db2xml.extractSmallints(db2xml.File
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/Quantity')) as x;
```

extractDouble() et extractDoubles()

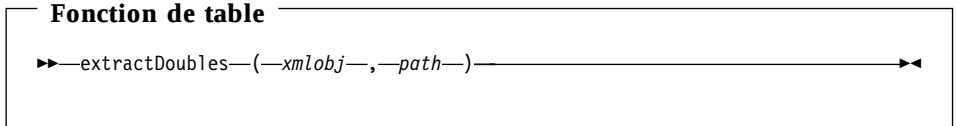
Objet
Extrait le contenu de l'élément ou la valeur de l'attribut d'un document XML et renvoie des données de type DOUBLE.

Syntaxe

Fonction scalaire



Fonction de table



Paramètres

Tableau 31. Paramètres des fonctions extractDouble et extractDoubles

Paramètre	Type de données	Description
xmlobj	XMLVARCHAR, XMLFILE ou XMLCLOB	Nom de la colonne.
path	VARCHAR	Chemin d'emplacement de l'élément ou de l'attribut.

Type de données renvoyé
DOUBLE

Nom de colonne renvoyé (fonction de table)
returnedDouble

Exemple
Exemple de fonction scalaire :

L'exemple suivant illustre la conversion automatique d'un prix de type DOUBLE à un prix de type DECIMAL.

```
CREATE TABLE t1(price, DECIMAL(5,2));
INSERT into t1 values (db2xml.extractDouble(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'),
    '/Order/Part[@color="black "]/ExtendedPrice'));
SELECT * from t1;
```

Exemple de fonction de table :

Dans l'exemple suivant, le prix calculé (ExtendedPrice) indiqué pour chaque pièce du bon de commande est extrait en tant que valeur DOUBLE.

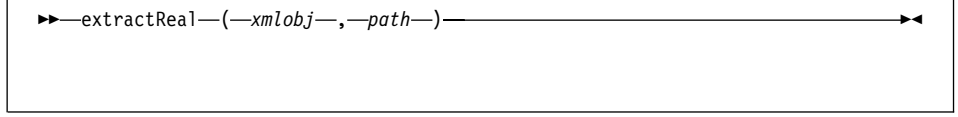
```
SELECT * from table(db2xml.extractDoubles(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/ExtendedPrice')) as x;
```

extractReal() et extractReals()

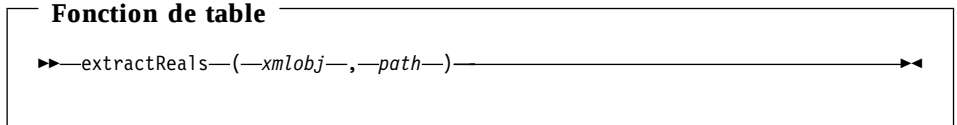
Objet
Extrait le contenu de l'élément ou la valeur de l'attribut d'un document XML et renvoie des données de type REAL.

Syntaxe

Fonction scalaire



Fonction de table



Paramètres

Tableau 32. Paramètres des fonctions extractReal et extractReals

Paramètre	Type de données	Description
xmlobj	XMLVARCHAR, XMLFILE ou XMLCLOB	Nom de la colonne.
path	VARCHAR	Chemin d'emplacement de l'élément ou de l'attribut.

Type de données renvoyé
REAL

Nom de colonne renvoyé (fonction de table)
returnedReal

Exemple
Dans l'exemple suivant, chaque taxe (Tax) est extraite en tant que valeur REAL.

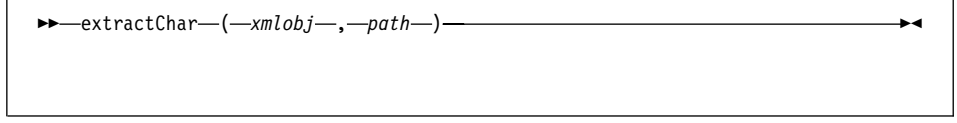
```
SELECT * from table(db2xml.extractReals(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/Tax')) as x;
```

extractChar() et extractChars()

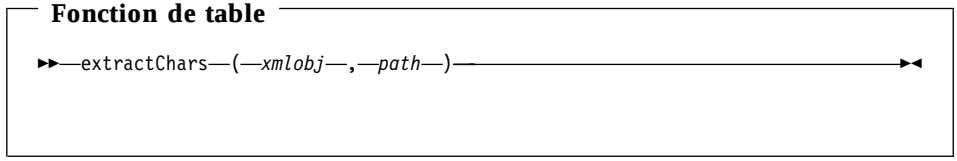
Objet
Extrait le contenu de l'élément ou la valeur de l'attribut d'un document XML et renvoie des données de type CHAR.

Syntaxe

Fonction scalaire



Fonction de table



Paramètres

Tableau 33. Paramètres des fonctions extractChar et extractChars

Paramètre	Type de données	Description
xmlobj	XMLVARCHAR, XMLFILE ou XMLCLOB	Nom de la colonne.
path	VARCHAR	Chemin d'emplacement de l'élément ou de l'attribut.

Type de données renvoyé
CHAR

Nom de colonne renvoyé (fonction de table)
returnedChar

Exemple
Dans l'exemple suivant, la couleur (Color) est extraite en tant que valeur CHAR.

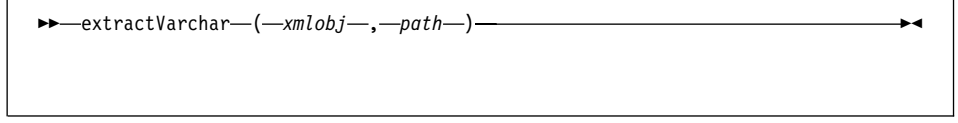
```
SELECT * from table(db2xml.extractChars(Order,
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/@Color')) as x;
```

extractVarchar() et extractVarchars()

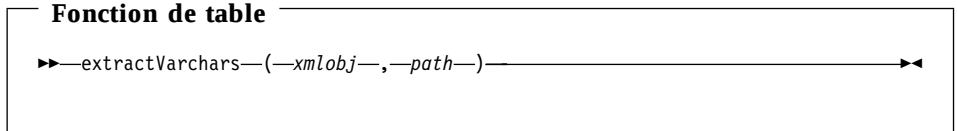
Objet
Extrait le contenu de l'élément ou la valeur de l'attribut d'un document XML et renvoie des données de type VARCHAR.

Syntaxe

Fonction scalaire



Fonction de table



Paramètres

Tableau 34. Paramètres des fonctions extractVarchar et extractVarchars

Paramètre	Type de données	Description
xmlobj	XMLVARCHAR, XMLFILE ou XMLCLOB	Nom de la colonne.
path	VARCHAR	Chemin d'emplacement de l'élément ou de l'attribut.

Type de données renvoyé
VARCHAR(4K)

Nom de colonne renvoyé (fonction de table)
returnedVarchar

Exemple
Dans une base de données comportant plus de 1000 documents XML stockés dans la colonne ORDER de la table SALES_TAB, vous souhaitez rechercher tous les clients qui ont commandé des articles dont le prix calculé (ExtendedPrice) dépasse 2500,00. L'instruction SQL suivante utilise la fonction UDF d'extraction dans la clause SELECT :

```
SELECT extractVarchar(Order, '/Order/Customer/Name') from sales_order_view
   WHERE price > 2500.00
```

La fonction UDF extractVarchar() considère la colonne ORDER comme paramètre d'entrée et le chemin d'emplacement `/Order/Customer/Name`, comme identificateur de la clause SELECT. La fonction UDF renvoie les noms des clients. Avec la clause WHERE, la fonction d'extraction évalue uniquement les commandes dont le prix calculé est supérieur à 2500,00.

extractCLOB() et extractCLOBs()

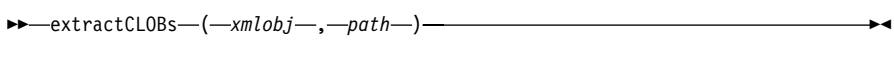
Objet

Extrait un fragment de documents XML, avec les marques d'éléments et d'attributs, le contenu des éléments et des attributs, ainsi que les sous-éléments. Cette fonction diffère des autres fonctions d'extraction car ces dernières renvoient uniquement le contenu des éléments et des attributs. Les fonctions extractClob(s) doivent être utilisées pour extraire des fragments de documents, alors que les fonctions extractVarchar(s) et extractChar(s) doivent l'être pour extraire des valeurs simples.

Syntaxe

Fonction scalaire

Fonction de table



Paramètres

Tableau 35. Paramètres des fonctions extractCLOB et extractCLOBs

Paramètre	Type de données	Description
xmlobj	XMLVARCHAR, XMLFILE ou XMLCLOB	Nom de la colonne.
path	VARCHAR	Chemin d'emplacement de l'élément ou de l'attribut.

Type de données renvoyé

CLOB(10K)

Nom de colonne renvoyé (fonction de table)

returnedCLOB

Exemple

Dans cet exemple, toutes les pièces (éléments part) sont extraites d'un bon de commande.

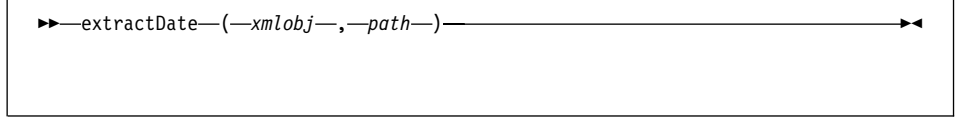
```
SELECT returnedCLOB as part
   from table(db2xml.extractCLOBs(db2xml.XMLFile('c:\dxx\samples\xml\getstart.xml'),
   '/Order/Part')) as x;
```

extractDate() et extractDates()

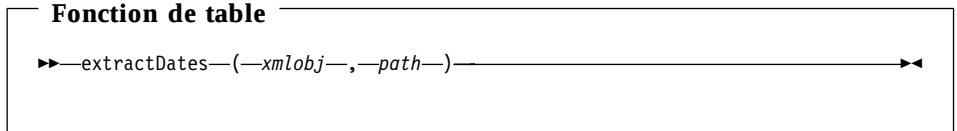
Objet
Extrait le contenu de l'élément ou la valeur de l'attribut d'un document XML et renvoie des données de type DATE.

Syntaxe

Fonction scalaire



Fonction de table



Paramètres

Tableau 36. Paramètres des fonctions extractDate et extractDates

Paramètre	Type de données	Description
xmlobj	XMLVARCHAR, XMLFILE ou XMLCLOB	Nom de la colonne.
path	VARCHAR	Chemin d'emplacement de l'élément ou de l'attribut.

Type de données renvoyé
DATE

Nom de colonne renvoyé (fonction de table)
returnedDate

Exemple
Dans l'exemple suivant, la date de livraison (ShipDate) est extraite en tant que valeur DATE.

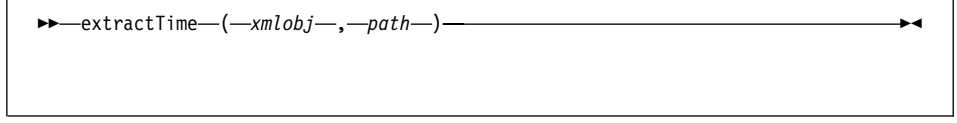
```
SELECT * from table(db2xml.extractDates(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/Shipment/ShipDate')) as x;
```

extractTime() et extractTimes()

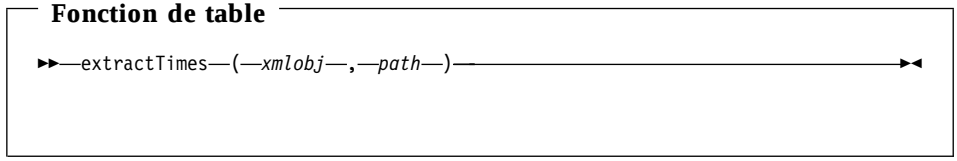
Objet
Extrait le contenu de l'élément ou la valeur de l'attribut d'un document XML et renvoie des données de type TIME.

Syntaxe

Fonction scalaire



Fonction de table



Paramètres

Tableau 37. Paramètres des fonctions extractTime et extractTimes

Paramètre	Type de données	Description
xmlobj	XMLVARCHAR, XMLFILE ou XMLCLOB	Nom de la colonne.
path	VARCHAR	Chemin d'emplacement de l'élément ou de l'attribut.

Type de données renvoyé
TIME

Nom de colonne renvoyé (fonction de table)
returnedTime

Exemple
Cet exemple utilise les fichiers exemples relatifs à la documentation. Il recherche dans le fichier XML `book.xml` la date à laquelle le prix de vente a été indiqué sur chaque manuel et renvoie les valeurs en tant que TIME.

Fichier XML :

```
<?xml version="1.0">
<DOCTYPE book SYSTEM "c:\dxx\samples\book.dtd">
<book>
<chapter id="1" date="07/01/97">
<section>This is a section in Chapter One.</section>
<chapter id="2" date="01/02/1997">
<section>This is a section in Chapter Two.</section>
</chapter>
<price date="12/22/1998" time="11.12.13" timestamp="1998-12-22-11.12.13.888888">
38.281
</price>
</book>
```

Exemple d'extraction :

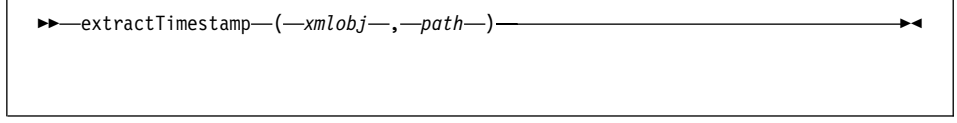
```
CREATE TABLE t1(SaleTime TIME);
INSERT INTO t1 values(db2xml.extractTime(doc, '/book/price/@time'));
SELECT * from t1;
```

extractTimestamp() et extractTimestamps()

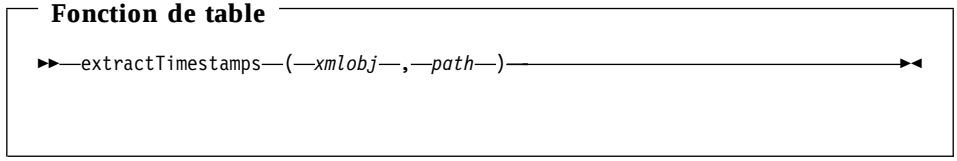
Objet
Extrait le contenu de l'élément ou la valeur de l'attribut d'un document XML et renvoie des données de type TIMESTAMP.

Syntaxe

Fonction scalaire



Fonction de table



Paramètres

Tableau 38. Paramètres des fonctions extractTimestamp et extractTimestamps

Paramètre	Type de données	Description
xmlobj	XMLVARCHAR, XMLFILE ou XMLCLOB	Nom de la colonne.
path	VARCHAR	Chemin d'emplacement de l'élément ou de l'attribut.

Type de données renvoyé
TIMESTAMP

Nom de colonne renvoyé (fonction de table)
returnedTimestamp

Exemple
Cet exemple utilise les fichiers exemples relatifs à la documentation. Il recherche dans le fichier XML `book.xml` la date à laquelle le prix de vente a été indiqué sur chaque livre et renvoie les valeurs en tant que TIMESTAMP.

Fichier XML :

```
<?xml version="1.0">
<DOCTYPE book SYSTEM "c:\dxx\samples\book.dtd">
<book>
<chapter id="1" date="07/01/97">
<section>This is a section in Chapter One.</section>
<chapter id="2" date="01/02/1997">
<section>This is a section in Chapter Two.</section>
</chapter>
<price date="12/22/1998" time="11.12.13" timestamp="1998-12-22-11.12.13.888888">
38.281
</price>
</book>
```

Exemple d'extraction :

```
CREATE TABLE t1(SaleTime TIMESTAMP);
INSERT INTO t1 values(db2xml.extractTimestamp(doc, '/book/price/@timestamp'));
SELECT * from t1;
```

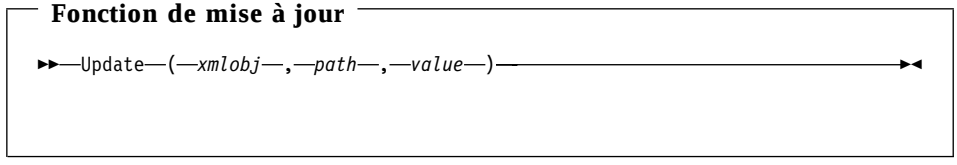
Fonction de mise à jour

La fonction Update() met à jour une valeur d'élément ou d'attribut spécifique dans un ou plusieurs documents XML stockés dans la colonne XML. Vous pouvez également utiliser les fonctions de transtypage par défaut pour convertir un type de base SQL en type UDT XML, comme indiqué dans la section «Mise à jour de données XML» à la page 132.

Objectif

Utilise comme paramètre d'entrée un nom de colonne de type UDT XML, un chemin d'emplacement (path) et une chaîne de valeur de mise à jour. Renvoie un type UDT XML identique au premier paramètre d'entrée. La fonction Update() permet de préciser l'élément ou l'attribut à mettre à jour.

Syntaxe



Paramètres

Tableau 39. Paramètres de la fonction UDF Update

Paramètre	Type de données	Description
xmlobj	XMLVARCHAR, XMLCLOB as LOCATOR	Nom de la colonne.
path	VARCHAR	Chemin d'emplacement de l'élément ou de l'attribut.
value	VARCHAR	Chaîne de mise à jour.

Important : Notez que la fonction UDF de mise à jour prend en charge les chemins d'emplacement qui comportent des prédicats avec des attributs, mais sans éléments. Par exemple, le prédicat qui suit est pris en charge :

```
'/Order/Part[@color="black "]/ExtendedPrice'
```

Par contre, le prédicat qui suit n'est pas pris en charge :

```
'/Order/Part/Shipment/[Shipdate < "11/25/00"]'
```

Type de retour

Type de données	Type de retour
XMLVARCHAR	XMLVARCHAR

Type de données	Type de retour
XMLCLOB as LOCATOR	XMLCLOB

Exemple

L'exemple suivant illustre la mise à jour du bon de commande traité par le vendeur Sriram Srinivasan.

```
UPDATE sales_tab
       set order = Update(order, '/Order/Customer/Name', 'IBM')
     WHERE sales_person = 'Sriram Srinivasan'
```

Dans cet exemple, le contenu de `/Order/Customer/Name` est remplacé par `IBM`.

Description

Lorsque vous utilisez la fonction Update pour modifier une valeur dans un ou plusieurs documents XML, elle effectue en fait un remplacement des documents XML dans la colonne XML. En fonction des données de sortie de l'analyseur XML, certaines parties du document d'origine sont conservées, alors que d'autres sont éliminées ou modifiées. Les sections qui suivent expliquent la façon dont le document est traité et fournissent des exemples de documents avant et après le traitement.

Mode de traitement du document XML par la fonction Update

Lorsque la fonction Update procède au remplacement d'un document XML, elle doit reconstruire le document en fonction des données de sortie de l'analyseur XML. Le tableau 40 à la page 208 décrit à l'aide d'exemples la façon dont les différentes parties du document sont gérées. Pour obtenir des exemples de documents XML avant et après traitement, reportez-vous à la section «Exemples» à la page 211

Tableau 40. Règles de fonctionnement de la fonction Update

Type d'élément ou de noeud	Exemple de code de document XML	Etat après mise à jour
Déclaration XML	`<?xml version='1.0' encoding='utf-8' standalone='yes' >`	La déclaration XML est conservée : • Les informations de version sont conservées. • La déclaration ENCODING est conservée et apparaît aux endroits où elle était spécifiée dans le document d'origine. • La déclaration STANDALONE est conservée et apparaît aux endroits où elle était spécifiée dans le document d'origine. • Après mise à jour, des apostrophes sont utilisées pour délimiter les valeurs.

Tableau 40. Règles de fonctionnement de la fonction Update (suite)

Type d'élément ou de noeud	Exemple de code de document XML	Etat après mise à jour

Déclaration DOCTYPE	`<!DOCTYPE books SYSTEM`
	`  "http://dtds.org/books.dtd" >`
	`<!DOCTYPE books PUBLIC`
	`  "local.books.dtd"`
	`  "http://dtds.org/books.dtd" >`
	`<!DOCTYPE books>`
	`-Any of`
	`<!DOCTYPE books`
	`  ( S ExternalID ) ?`
	`  [ internal-dtd-subset ] >`
	`-Such as`
	`<!DOCTYPE books`
	`  [ <!ENTITY mydog "Spot"> ] >?`
	`  [ internal-dtd-subset ] >`
		La déclaration de type de document est conservée :
		• Le nom de l'élément racine est pris en charge.
		• Les `ExternalID` de type PUBLIC et SYSTEM sont conservés et apparaissent aux endroits où ils étaient spécifiés dans le document d'origine.
		• Le sous-ensemble de DTD interne (Internal DTD subset) N'EST PAS conservé. Les entités sont remplacées ; les valeurs par défaut des attributs sont traitées et apparaissent dans les documents de sortie.
		• Après mise à jour, des guillemets sont utilisés pour délimiter les valeurs URI de type PUBLIC et SYSTEM.
		• L'analyseur XML4c en cours ne signale pas une déclaration XML qui ne contient pas un `ExternalID` ou un sous-ensemble de DTD interne (internal DTD subset). Dans ce cas, après mise à jour, la déclaration DOCTYPE est manquante.

Instructions de traitement	`<?xml-stylesheet`
	`  title="compact"`
	`  href="datatypes1.xsl"`
	`  type="text/xsl"?>`
		Les instructions de traitement sont conservées.

Tableau 40. Règles de fonctionnement de la fonction Update (suite)

Type d'élément ou de noeud	Exemple de code de document XML	Etat après mise à jour
Commentaires	`<!-- comment -->`	Les commentaires sont conservés lorsqu'ils se trouvent dans l'élément racine. Ceux situés en dehors de cet élément sont supprimés.

Eléments	`<books>`
	content
	`</books>`
		Les éléments sont conservés.

Attributs	`id='1' date='01/02/1997"`	Les attributs d'éléments sont conservés. • Après mise à jour, des guillemets sont utilisés pour délimiter les valeurs. • Les données situées dans les attributs sont ignorées. • Les entités sont remplacées.
Noeuds de texte	`This chapter is about my dog &mydoc;.` `This chapter is about my dog Spot.`	Les noeuds de texte (contenu d'élément) sont conservés. • Les données situées dans les noeuds de texte sont ignorées. • Les entités sont remplacées.

Occurrences multiples

Lorsqu'un chemin d'emplacement est fourni dans la fonction UDF Update(), le contenu de chaque élément ou attribut ayant un chemin similaire est mis à jour avec la valeur fournie. Cela signifie que si un document comporte plusieurs chemins d'emplacement, le fonction Update remplace les valeurs existantes par la valeur fournie dans le paramètre *value*.

Vous pouvez spécifier dans le paramètre *path* un prédicat visant à fournir des chemins d'emplacement distincts afin d'empêcher toute mise à jour involontaire. Notez que la fonction UDF de mise à jour (Update) prend en

charge les chemins d'emplacement qui comportent des prédicats avec des attributs, mais sans éléments. Pour plus d'informations, reportez-vous à la section «Paramètres» à la page 206.

Exemples

Vous trouverez ci-après des exemples de documents XML avant et après mise à jour.

Exemple 1 :

Avant :

```
<?xml version='1.0'   encoding='utf-8'    standalone="yes"?>
<!DOCTYPE book PUBLIC "public.dtd" "system.dtd">
<?pitarget option1='value1' option2='value2'?>
<!-- comment -->
<book>
  <chapter id="1" date='07/01/1997'>
    <!-- first section -->
<section>This is a section in Chapter One.</section>
</chapter>
<chapter id="2" date="01/02/1997">
<section>This is a section in Chapter Two.</section>
    <footnote>A footnote in Chapter Two is here.</footnote>
</chapter>
  <price date="12/22/1998" time="11.12.13"
       timestamp="1998-12-22-11.12.13.888888">38.281</price>
</book>
```

- Mentionne un espace dans la déclaration XML
- Spécifie une instruction de traitement
- Contient un commentaire situé en dehors du noeud racine
- Mentionne un ExternalID de type PUBLIC
- Contient un commentaire situé à l'intérieur du noeud racine

Après :

```
<?xml version='1.0' encoding='utf-8' standalone='yes'?>
<!DOCTYPE book PUBLIC "public.dtd" "system.dtd">
<?pitarget option1='value1' option2='value2'?><book>
  <chapter id="1" date="07/01/1997">
        <!-- first section -->
<section>This is a section in Chapter One.</section>
</chapter>
<chapter id="2" date="01/02/1997">
<section>This is a section in Chapter Two.</section>
        <footnote>A footnote in Chapter Two is here.</footnote>
</chapter>
  <price date="12/22/1998" time="11.12.13"
       timestamp="1998-12-22-11.12.13.888888">60.02</price>
</book>
```

- L'espace situé dans les marques est éliminé
- L'instruction de traitement est conservée
- Le commentaire situé en dehors du noeud racine n'est pas conservé
- L'instruction ExternalID de type PUBLIC est conservée
- Les commentaires situés à l'intérieur du noeud racine sont conservés
- La valeur modifiée est la valeur de l'élément <price>

Exemple 2 :

Avant :

```
<?xml version='1.0'     ?>
<!DOCTYPE book>
<!-- comment -->
<book>
  ...
</book>
```

Contient une déclaration DOCTYPE sans ExternalID ou sans sous-ensemble de DTD interne (internal DTD subset). Non pris en charge.

Après :

```
<?xml version='1.0'?>
<book>
  ...
</book>
```

La déclaration DOCTYPE n'est pas signalée par l'analyseur XML et n'est pas conservée.

Exemple 3 :

Avant :

```
<?xml version='1.0'     ?>
<!DOCTYPE book [ <!ENTITY myDog "Spot"> ]>
<!-- comment -->
<book>
  <chapter id="1" date='07/01/1997'>
       <!-- first section -->
       <section>This is a section in Chapter
         One about my dog &;myDog;.</section>
       ...
</chapter>
  ...
</book>
```

- Mentionne un espace dans les marques
- Indique un sous-ensemble de DTD interne (internal DTD subset)
- Indique une entité dans le noeud de texte

Après :

```
<?xml version='1.0'?>
<!DOCTYPE book>
<book>
  <chapter id="1" date="07/01/1997">
       <!-- first section -->
       <section>This is a section in Chapter
         One about my dog Spot.</section>
          ...
</chapter>
  ...
</book>
```

- L'espace est éliminé dans les marques
- Le sous-ensemble de DTD interne (Internal DTD subset) n'est pas conservé.
- L'entité située dans le noeud de texte est résolue et remplacée

Chapitre 10. Procédures mémorisées de l'Extension XML

L'Extension XML comporte des procédures mémorisées pour l'administration et la gestion des colonnes et des collections XML. Ces procédures mémorisées peuvent être appelées à partir du client DB2. L'interface client peut être imbriquée dans un programme SQL, ODBC ou JDBC. Pour plus d'informations sur l'appel de procédures mémorisées, reportez-vous à la section correspondante dans le manuel *DB2 UDB Administration Guide.*

Les procédures mémorisées utilisent le schéma db2xml, qui constitue le nom de schéma de l'Extension XML

L'Extension XML comporte trois types de procédures mémorisées :

- «Procédures mémorisées d'administration» à la page 217 : assistent les utilisateurs dans la réalisation des tâches administratives.
- «Procédures mémorisées de composition» à la page 225 : génèrent des documents XML à partir de données contenues dans des tables de la base de données.
- «Procédures mémorisées de décomposition» à la page 234 : décomposent ou broient les documents XML entrants et stockent les données dans des tables nouvelles ou existantes de la base de données.

Les limites des paramètres utilisés par les procédures mémorisées des collections XML sont détaillées à l'«Annexe D. Les limites de l'Extension XML» à la page 297.

Indication de fichiers d'inclusion

Vérifiez que vous avez inclus les fichiers d'en-tête externes de l'Extension XML dans le programme qui appelle les procédures mémorisées. Ces fichiers d'en-tête se trouvent dans le répertoire `DXX_INSTALL/include`. `DXX_INSTALL` est le répertoire d'installation de l'Extension XML. Il dépend du système d'exploitation. Les fichiers d'en-tête sont les suivants :

dxx.h Types de constantes et de données définis par l'Extension XML.

dxxrc.h Code retour renvoyé par l'Extension XML.

La syntaxe permettant d'inclure ces fichiers d'en-tête est la suivante :

```
#include "dxx.h"
#include "dxxrc.h"
```

Assurez-vous que le chemin des fichiers d'inclusion est indiqué dans le fichier makefile avec l'option de compilation.

Appel des procédures mémorisées de l'Extension XML

En général, la syntaxe permettant d'appeler l'Extension XML est la suivante :

```
CALL db2xml.function_entry_point
```

Où :

db2xml
Indique la bibliothèque qui regroupe les procédures mémorisées de l'Extension XML. Sous Unix, cette bibliothèque est stockée dans le répertoire `sqllib/function`. Sous Windows, elle est stockée dans le répertoire *DXX_INSTALL*`/bin`.

function_entry_point
Indique les arguments transmis à la procédure mémorisée.

Dans l'instruction CALL, les arguments transmis à la procédure mémorisée doivent être des variables hôte, et non des constantes ou des expressions. Les variables hôte peuvent être associées à des indicateurs NULL. Vous trouverez des exemples d'appel de procédures mémorisées dans les répertoires `DXX_INSTALL/samples/c` et `DXX_INSTALL/samples/cli`, ainsi que dans les sections «Composition du document XML» à la page 42 et «Chapitre 6. Gestion des données de collection XML» à la page 141 de ce manuel.

Dans le répertoire `DXX_INSTALL/samples/c`, des fichiers de code SQC sont fournis pour appeler des procédures mémorisées de collection XML à l'aide d'instructions SQL imbriquées. Dans le répertoire `DXX_INSTALL/samples/cli`, les fichiers exemples indiquent comment appeler les procédures mémorisées à l'aide de l'interface CLI (Call Level Interface).

Augmentation des limites du paramètre CLOB

La limite par défaut affectée au paramètre CLOB lorsqu'il est transmis à une procédure mémorisée est de 1 Mo. Vous pouvez augmenter cette limite en procédant comme suit :

1. Supprimez chaque procédure mémorisée. Par exemple :

   ```
   db2   "drop procedure db2xml.dxxShredXML"
   ```

2. Créez une nouvelle procédure en lui associant une limite CLOB supérieure. Par exemple :

   ```
   db2   "create procedure db2xml.dxxShredXML(in     dadBuf      clob(100K),
                                              in     XMLObj      clob(10M),
                                              out    returnCode  integer,
                                              out    returnMsg   varchar(1024)
                                              )
   ```

```
external name 'db2xml.dxxShredXML'
language C
parameter style DB2DARI
not deterministic
fenced
null call;
```

Avant de commencer

Associez votre base de données aux fichiers de lien des procédures mémorisées et de l'interface CLI DB2. Pour ce faire, vous pouvez utiliser le fichier de commandes exemple getstart_prep.cmd. Ce fichier de commandes se trouve dans le répertoire DXX_INSTALL/samples/cmd.

1. Connectez-vous à la base de données. Par exemple :

   ```
   db2 "connect to SALES_DB"
   ```

2. Allez dans le répertoire DXX_INSTALL/bnd et associez l'Extension XML à la base de données.

   ```
   db2 "bind @dxxbind.lst"
   ```

3. Allez dans le répertoire sqllib/bnd et associez l'interface CLI à la base de données.

   ```
   db2 "bind @db2cli.lst"
   ```

4. Mettez fin à la connexion.

   ```
   db2 "terminate"
   ```

Procédures mémorisées d'administration

Ces procédures mémorisées sont utilisées pour les tâches d'administration, telles que l'activation ou la désactivation d'une colonne ou d'une collection XML. Elles sont appelées par l'assistant d'administration de l'Extension XML et par la commande d'administration **dxxadm**.

dxxEnableDB()

Objet

Active la base de données. Une fois la base de données activée, Extension XML crée les objets suivants :

- Types définis par l'utilisateur (UDT)
- Fonctions définies par l'utilisateur (UDF)
- Table de référence des DTD (DTD_REF), pour le stockage des définitions de types de documents et les informations associées. Pour obtenir une description complète de la table DTD_REF, reportez-vous à la section «Table DTD_REF» à la page 239.
- Table d'usage (XML_USAGE), pour le stockage des informations communes à chaque colonne et collection XML. Pour obtenir une description complète de la table XML_USAGE, reportez-vous à la section «Table XML_USAGE» à la page 240.

```
dxxEnableDB(char(dbName) dbName,       /* input */
            long           returnCode,       /* output */
            varchar(1024)  returnMsg)        /* output */
```

Paramètres

Tableau 41. Paramètres dxxEnableDB()

Paramètre	Description	Paramètre IN/OUT
dbName	Nom de la base de données.	IN
returnCode	Code retour renvoyé par la procédure mémorisée.	OUT
returnMsg	Message renvoyé en cas d'erreur.	OUT

dxxDisableDB()

Objet

Désactive la base de données. Une fois la base de données désactivée, l'Extension XML retire les objets suivants :

- Types définis par l'utilisateur (UDT)
- Fonctions définies par l'utilisateur (UDF)
- Table de référence des DTD (DTD_REF), pour le stockage des définitions de types de documents et les informations associées. Pour obtenir une description complète de la table DTD_REF, reportez-vous à la section «Table DTD_REF» à la page 239.
- Table d'usage (XML_USAGE), pour le stockage des informations communes à chaque colonne et collection XML. Pour obtenir une description complète de la table XML_USAGE, reportez-vous à la section «Table XML_USAGE» à la page 240.

Important : Pour pouvoir désactiver une base de données, vous devez d'abord désactiver toutes les colonnes et collections XML associées.

```
dxxDisableDB(char(dbName)     dbName,       /* input */
             long             returnCode,     /* output */
             varchar(1024)    returnMsg)      /* output */
```

Paramètres

Tableau 42. Paramètres dxxDisableDB()

Paramètre	Description	Paramètre IN/OUT
dbName	Nom de la base de données.	IN
returnCode	Code retour renvoyé par la procédure mémorisée.	OUT
returnMsg	Message renvoyé en cas d'erreur.	OUT

dxxEnableColumn()

Objet

Active une colonne XML. Lors de l'activation d'une colonne, l'Extension XML exécute les tâches suivantes :

- Il détermine si la table XML possède une clé primaire. Dans la négative, il la modifie et ajoute une colonne DXXROOT_ID.
- Il crée les tables annexes indiquées dans le fichier DAD, avec une colonne contenant un identificateur unique pour chaque ligne de la table XML. Cette colonne correspond à l'identificateur racine (*root_id*) indiqué par l'utilisateur, ou à l'identificateur DXXROOT_ID nommé par l'Extension XML.
- Il crée une vue par défaut de la table XML et de ses tables annexes, en lui attribuant le nom éventuellement indiqué.

```
dxxEnableColumn(char(dbName) dbName,      /* input */
                char(tbName) tbName,      /* input */
                char(colName) colName,     /* input */
                    CLOB(100K)     DAD,         /* input */
                char(tablespace) tablespace,  /* input */
                char(defaultView) defaultView, /* input */
                char(rootID) rootID,      /* input */
             long           returnCode,       /* output */
             varchar(1024)  returnMsg)        /* output */
```

Paramètres

Tableau 43. Paramètres dxxEnableColumn()

Paramètre	Description	Paramètre IN/OUT
dbName	Nom de la base de données.	IN
tbName	Nom de la table contenant la colonne XML.	IN
colName	Nom de la colonne XML.	IN
DAD	Objet CLOB contenant le fichier DAD.	IN
tablespace	Table espace contenant les tables annexes autres que l'espace table par défaut. Si vous laissez cette zone à blanc, l'Extension XML utilise un espace table par défaut.	IN
defaultView	Nom de la vue par défaut réalisant la jointure de la table d'application et des tables annexes.	IN

Tableau 43. Paramètres dxxEnableColumn() (suite)

Paramètre	Description	Paramètre IN/OUT
rootID	Nom de la clé primaire trouvée dans la table d'application, à utiliser comme identificateur racine (*root_id*) pour les tables annexes.	IN
returnCode	Code retour renvoyé par la procédure mémorisée.	OUT
returnMsg	Message renvoyé en cas d'erreur.	OUT

dxxDisableColumn()

Objet

Désactive la colonne activée pour XML. Lorsqu'une colonne XML est désactivée, elle ne peut plus contenir de données de type XML.

```
dxxDisableColumn(char(dbName) dbName,      /* input */
                 char(tbName) tbName,      /* input */
                 char(colName) colName,     /* input */
             long           returnCode,       /* output */
             varchar(1024)  returnMsg)        /* output */
```

Paramètres

Tableau 44. Paramètres dxxDisableColumn()

Paramètre	Description	Paramètre IN/OUT
dbName	Nom de la base de données.	IN
tbName	Nom de la table contenant la colonne XML.	IN
colName	Nom de la colonne XML.	IN
returnCode	Code retour renvoyé par la procédure mémorisée.	OUT
returnMsg	Message renvoyé en cas d'erreur.	OUT

dxxEnableCollection()

Objet

Active une collection XML associée à une table d'application.

```
dxxEnableCollection(char(dbName) dbName,      /* input */
                    char(colName) colName,     /* input */
                    CLOB(100K)     DAD,         /* input */
                    char(tablespace) tablespace,  /* input */
             long           returnCode,       /* output */
             varchar(1024)  returnMsg)        /* output */
```

Paramètres

Tableau 45. Paramètres dxxEnableCollection()

Paramètre	Description	Paramètre IN/OUT
dbName	Nom de la base de données.	IN
colName	Nom de la collection XML.	IN
DAD	Objet CLOB contenant le fichier DAD.	IN
tablespace	Table espace contenant les tables annexes autres que l'espace table par défaut. Si vous laissez cette zone à blanc, l'Extension XML utilise un espace table par défaut.	IN
returnCode	Code retour renvoyé par la procédure mémorisée.	OUT
returnMsg	Message renvoyé en cas d'erreur.	OUT

dxxDisableCollection()

Objet

Désactive une collection XML, en retirant les marqueurs identifiant les tables et les colonnes au sein d'une collection.

```
dxxDisableCollection(char(dbName) dbName,      /* input */
                     char(colName) colName,    /* input */
             long           returnCode,       /* output */
             varchar(1024)  returnMsg)        /* output */
```

Paramètres

Tableau 46. Paramètres dxxDisableCollection()

Paramètre	Description	Paramètre IN/OUT
dbName	Nom de la base de données.	IN
colName	Nom de la collection XML.	IN
returnCode	Code retour renvoyé par la procédure mémorisée.	OUT
returnMsg	Message renvoyé en cas d'erreur.	OUT

Procédures mémorisées de composition

dxxGenXML() et dxxRetrieveXML() sont des procédures mémorisées de composition. Elles permettent de générer des documents XML à partir de données contenues dans des tables de la base de données. La procédure mémorisée dxxGenXML() utilise un fichier DAD comme paramètre d'entrée. Elle ne nécessite pas l'activation d'une collection XML. La procédure mémorisée dxxRetrieveXML() utilise un nom de collection XML activée comme paramètre d'entrée.

dxxGenXML()

Objet

Construit des documents XML à partir des tables de collection XML indiquées par <Xcollection> dans le fichier DAD et insère chaque document XML sous forme de ligne dans la table de résultats. Vous pouvez également ouvrir un curseur dans la table de résultats et extraire l'ensemble de résultats.

Pour une plus grande souplesse d'utilisation, dxxGenXML() permet également de préciser le nombre maximal de lignes à générer dans la table de résultats. Ceci permet de réduire le délai pendant lequel l'application doit attendre les résultats lors d'une procédure d'essai. La procédure mémorisée renvoie le nombre réel de lignes composant la table et toute information sur les erreurs, y compris les codes et les messages d'erreur.

Pour prendre en charge les requêtes dynamiques, dxxGenXML() utilise le paramètre d'entrée *override*. En fonction du paramètre d'entrée *overrideType*, l'application peut remplacer dans le fichier DAD la valeur de l'élément SQL_stmt par le mode de mappage SQL ou les conditions RDB_node par le mode de mappage du noeud RDB. Le paramètre d'entrée *overrideType* permet de préciser le type du paramètre *override*. Pour plus d'informations sur le paramètre *override*, reportez-vous à la section «Remplacement dynamique de valeurs dans le fichier DAD» à la page 145.

```
dxxGenXML(CLOB(100K)     DAD,             /* input */
          char(resultTabName) resultTabName,   /* input */
          integer        overrideType     /* input */
               varchar(1024)  override,        /* input */
               integer        maxRows,         /* input */
               integer        numRows,         /* output */
             long           returnCode,       /* output */
             varchar(1024)  returnMsg)        /* output */
```

Paramètres

Tableau 47. Paramètres dxxGenXML()

Paramètre	Description	Paramètre IN/OUT
DAD	Objet CLOB contenant le fichier DAD.	IN
resultTabName	Nom de la table de résultats, existant avant l'appel. La table ne contient qu'une seule colonne de type XMLVARCHAR ou XMLCLOB.	IN

Tableau 47. Paramètres dxxGenXML() (suite)

Paramètre	Description	Paramètre IN/OUT
overrideType	Indicateur précisant le type du paramètre *override* : • **NO_OVERRIDE** : Aucun remplacement. • **SQL_OVERRIDE** : Remplacement par une valeur SQL_stmt. • **XML_OVERRIDE** : Remplacement par une condition basée sur XPath.	IN
override	Remplace la condition dans le fichier DAD. La valeur d'entrée dépend du paramètre *overrideType*. • **NO_OVERRIDE** : chaîne de type NULL. • **SQL_OVERRIDE** : instruction SQL valide. Avec cette valeur *overrideType*, le mode de mappage SQL doit être utilisé dans le fichier DAD. L'instruction SQL d'entrée remplace la valeur de l'élément SQL_stmt dans le fichier DAD. • **XML_OVERRIDE** : chaîne d'une ou plusieurs expressions placées entre guillemets et séparées par ″AND″. Avec cette valeur *overrideType*, le mode de mappage du noeud RDB doit être utilisé dans le fichier DAD.	IN
maxRows	Nombre maximal de lignes à renvoyer dans la table de résultats.	IN
numRows	Nombre réel de lignes générées dans la table de résultats.	OUT

Tableau 47. Paramètres dxxGenXML() (suite)

Paramètre	Description	Paramètre IN/OUT
returnCode	Code retour renvoyé par la procédure mémorisée.	OUT
returnMsg	Message renvoyé en cas d'erreur.	OUT

Exemples

Soit une table de résultats XML_ORDER_TAB, contenant une colonne de type XMLVARCHAR.

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE is CLOB(100K) dad;              /* DAD */
     SQL TYPE is CLOB_FILE  dadFile;          /* dad file */
              char    result_tab[32];    /* name of the result table */
              char    override[2];       /* override, will set to NULL*/
              short   overrideType;      /* defined in dxx.h */
              short   max_row;           /* maximum number of rows */
              short   num_row;           /* actual number of rows */
              long    returnCode;        /* return error code */
              char                  returnMsg[1024]; /* error message text */
              short       dad_ind;
              short   rtab_ind;
              short   ovtype_ind;
              short   ov_inde;
              short   maxrow_ind;
              short   numrow_ind;
              short                 returnCode_ind;
              short                 returnMsg_ind;

  EXEC SQL END DECLARE SECTION;

              /* create table */
         EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

     /* read data from a file to a CLOB */
              strcpy(dadfile.name,"e:\dxx\dad\litem3.dad");
              dadfile.name_length = strlen("e:\dxx\dad\litem3.dad");
     dadfile.file_options = SQL_FILE_READ;
     EXEC SQL VALUES (:dadfile) INTO :dad;
              strcpy(result_tab,"xml_order_tab");
              override[0] = '\0';
              overrideType = NO_OVERRIDE;
              max_row = 500;
              num_row = 0;
              returnCode = 0;
              msg_txt[0] = '\0';
```

```
        collection_ind = 0;
        dad_ind = 0;
        rtab_ind = 0;
        ov_ind = -1;
        ovtype_ind = 0;
        maxrow_ind = 0;
        numrow_ind = -1;
        returnCode_ind = -1;
        returnMsg_ind = -1;

        /* Call the store procedure */
EXEC SQL CALL dxxGenXML(:dad:dad_ind;
               :result_tab:rtab_ind,
               :overrideType:ovtype_ind,:override:ov_ind,
               :max_row:maxrow_ind,:num_row:numrow_ind,
                        :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

dxxRetrieveXML()

Objet

Active le fichier DAD à utiliser pour les opérations de composition et de décomposition. La procédure mémorisée dxxRetrieveXML() permet également de récupérer des documents XML décomposés. La procédure mémorisée dxxRetrieveXML() utilise comme paramètres d'entrée une mémoire tampon contenant le fichier DAD, le nom de la table de résultats à générer et le nombre maximal de lignes à renvoyer. Elle renvoie un ensemble de la table de résultats, le nombre réel de lignes composant celle-ci, un code et un message d'erreur.

Pour prendre en charge les requêtes dynamiques, dxxRetrieveXML() utilise le paramètre d'entrée *override*. En fonction du paramètre d'entrée *overrideType*, l'application peut remplacer dans le fichier DAD la valeur de l'élément SQL_stmt par le mode de mappage SQL ou les conditions RDB_node par le mode de mappage du noeud RDB. Le paramètre d'entrée *overrideType* permet de préciser le type du paramètre *override*. Pour plus d'informations sur le paramètre *override*, reportez-vous à la section «Remplacement dynamique de valeurs dans le fichier DAD» à la page 145.

Les exigences liées au fichier DAD pour dxxRetrieveXML() sont identiques à celles de la procédure dxxGenXML(). La seule différence est que le fichier DAD n'est pas un paramètre d'entrée de dxxRetrieveXML(), mais le nom d'une collection XML activée.

```
dxxRetrieveXML(char(collectionName) collectionName,   /* input */
               char(resultTabName) resultTabName,    /* input */
               integer        overrideType,    /* input */
               varchar(1024)  override,        /* input */
               integer        maxRows,         /* input */
               integer        numRows,         /* output */
              long           returnCode,       /* output */
              varchar(1024)  returnMsg)        /* output */
```

Paramètres

Tableau 48. Paramètres dxxRetrieveXML()

Paramètre	Description	Paramètre IN/OUT
collectionName	Nom d'une collection XML activée.	IN
resultTabName	Nom de la table de résultats, existant avant l'appel. La table ne contient qu'une seule colonne de type XMLVARCHAR ou XMLCLOB.	IN

Tableau 48. Paramètres dxxRetrieveXML() (suite)

Paramètre	Description	Paramètre IN/OUT
overrideType	Indicateur précisant le type du paramètre *override* : • **NO_OVERRIDE** : Aucun remplacement. • **SQL_OVERRIDE** : Remplacement par une valeur SQL_stmt. • **XML_OVERRIDE** : Remplacement par une condition basée sur XPath.	IN
override	Remplace la condition dans le fichier DAD. La valeur d'entrée dépend du paramètre *overrideType*. • **NO_OVERRIDE** : chaîne de type NULL. • **SQL_OVERRIDE** : instruction SQL valide. Avec cette valeur *overrideType*, le mode de mappage SQL doit être utilisé dans le fichier DAD. L'instruction SQL d'entrée remplace la valeur de l'élément SQL_stmt dans le fichier DAD. • **XML_OVERRIDE** : chaîne d'une ou plusieurs expressions placées entre guillemets et séparées par ″AND″. Avec cette valeur *overrideType*, le mode de mappage du noeud RDB doit être utilisé dans le fichier DAD.	IN
maxRows	Nombre maximal de lignes à renvoyer dans la table de résultats.	IN
numRows	Nombre réel de lignes générées dans la table de résultats.	OUT

Tableau 48. Paramètres dxxRetrieveXML() (suite)

Paramètre	Description	Paramètre IN/OUT
returnCode	Code retour renvoyé par la procédure mémorisée.	OUT
returnMsg	Message renvoyé en cas d'erreur.	OUT

Exemples

Voici un exemple d'appel de la procédure dxxRetrieveXML(). Soit une table de résultats XML_ORDER_TAB, contenant une colonne de type XMLVARCHAR.

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;
              char    collection[32];    /* dad buffer */
              char    result_tab[32];    /* name of the result table */
              char    override[2];       /* override, will set to NULL*/
              short   overrideType;      /* defined in dxx.h */
              short   max_row;           /* maximum number of rows */
              short   num_row;           /* actual number of rows */
              long    returnCode;        /* return error code */
              char                   returnMsg[1024]; /* error message text */
              short   dadbuf_ind;
              short   rtab_ind;
              short   ovtype_ind;
              short   ov_inde;
              short   maxrow_ind;
              short   numrow_ind;
              short                  returnCode_ind;
              short                  returnMsg_ind;

  EXEC SQL END DECLARE SECTION;

              /* create table */
              EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

              /* initialize host variable and indicators */
              strcpy(collection,"sales_ord");
              strcpy(result_tab,"xml_order_tab");
              override[0] = '\0';
              overrideType = NO_OVERRIDE;
              max_row = 500;
              num_row = 0;
              returnCode = 0;
              msg_txt[0] = '\0';
              collection_ind = 0;
              rtab_ind = 0;
              ov_ind = -1;
              ovtype_ind = 0;
```

```
maxrow_ind = 0;
numrow_ind = -1;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Call the store procedure */
EXEC SQL CALL dxxRetrieve(:collection:collection_ind;
       :result_tab:rtab_ind,
       :overrideType:ovtype_ind,:override:ov_ind,
       :max_row:maxrow_ind,:num_row:numrow_ind,
                :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Procédures mémorisées de décomposition

dxxInsertXML() et dxxShredXML() sont des procédures mémorisées de décomposition. Elles permettent de décomposer ou de broyer des documents XML entrants et de stocker des données dans des tables nouvelles ou existantes de la base de données. La procédure mémorisée dxxInsertXML() utilise un nom de collection XML activée comme paramètre d'entrée. La procédure mémorisée dxxShredXML() utilise un fichier DAD comme paramètre d'entrée. Elle ne nécessite pas l'activation d'une collection XML.

dxxShredXML()

Objet

La procédure mémorisée dxxShredXML() est appariée à la procédure mémorisée dxxGenXML(). Pour que dxxShredXML() puisse s'exécuter, le fichier DAD doit indiquer des tables existantes, ainsi que des colonnes et des types de données cohérents avec celles-ci. Pour la procédure mémorisée dxxShredXML(), la relation clé primaire - clé associée entre les tables de jointure, créée par l'Extension XML lors de l'activation de la collection, n'est pas nécessaire. Cependant, les colonnes de condition de jointure indiquées dans le noeud RDB du noeud d'élément racine doivent exister dans les tables.

```
dxxShredXML(CLOB(100K)    DAD,            /* input */
            CLOB(1M)       xmlobj,           /* input */
            long           returnCode,       /* output */
            varchar(1024)  returnMsg)        /* output */
```

Paramètres

Tableau 49. Paramètres dxxShredXML()

Paramètre	Description	Paramètre IN/OUT
DAD	Objet CLOB contenant le fichier DAD.	IN
xmlobj	Objet document XML de type XMLCLOB.	IN
returnCode	Code retour renvoyé par la procédure mémorisée.	OUT
returnMsg	Message renvoyé en cas d'erreur.	OUT

Exemples

Voici un exemple d'appel de la procédure dxxShredXML().

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;
              SQL TYPE is CLOB dad;           /* DAD*/
              SQL TYPE is CLOB_FILE  dadFile; /* DAD file*/
              SQL TYPE is CLOB xmlDoc;        /* input XML document */
              SQL TYPE is CLOB_FILE xmlFile;  /* input XMLfile */
              long                  returnCode;      /* error code */
              char                  returnMsg[1024]; /* error message text */
              short       dad_ind;
              short                 xmlDoc_ind;
              short                 returnCode_ind;
              short                 returnMsg_ind;
```

```
EXEC SQL END DECLARE SECTION;

            /* initialize host variable and indicators */
            strcpy(dadFile.name,"c:\dxx\samples\dad\getstart_xcollection.dad");
            dadFile.name_length=strlen("c:\dxx\samples\dad\getstart_xcollection.dad");
            dadFile.file_option=SQL_FILE_READ;
            strcpy(xmlFile.name,"c:\dxx\samples\cmd\getstart.xml ");
            xmlFile.name_length=strlen("c:\dxx\samples\cmd\getstart.xml ");
            xmlFile.file_option=SQL_FILE_READ;
            SQL EXEC VALUES (:dadFile) INTO :dad;
            SQL EXEC VALUES (:xmlFile) INTO :xmlDoc;
            returnCode = 0;
            returnMsg[0] = '\0';
            dad_ind = 0;
            xmlDoc_ind = 0;
            returnCode_ind = -1;
            returnMsg_ind = -1;

            /* Call the store procedure */
            EXEC SQL CALL db2xml.dxxShredXML(:dad:dad_ind;
                           :xmlDoc:xmlDoc_ind,
                           :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

dxxInsertXML()

Objet

Utilise deux paramètres d'entrée, le nom d'une collection XML activée et celui du document XML à décomposer. Renvoie deux paramètres de sortie, un code et un message retour.

```
dxxInsertXML(char(collectionName) collectionName,   /* input */
             CLOB(1M)       xmlobj,          /* input */
             long           returnCode,      /* output */
             varchar(1024)  returnMsg)       /* output */
```

Paramètres

Tableau 50. Paramètres dxxInsertXML()

Paramètre	Description	Paramètre IN/OUT
collectionName	Nom d'une collection XML activée.	IN
xmlobj	Objet document XML de type CLOB.	IN
returnCode	Code retour renvoyé par la procédure mémorisée.	OUT
returnMsg	Message renvoyé en cas d'erreur.	OUT

Exemples

Dans l'exemple ci-après, vous appelez la procédure dxxInsertXML() qui décompose le document XML d'entrée `e:\xml\order1.xml`, puis insère les données dans les tables de la collection SALES_ORDER, en fonction du mappage indiqué dans le fichier DAD utilisé pour l'activation de la collection.

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;
             char                  collection[64];  /* name of an XML collection */
             SQL TYPE is CLOB_FILE xmlDoc;          /* input XML document */
             long                  returnCode;      /* error code */
             char                  returnMsg[1024]; /* error message text */
             short                 collection_ind;
             short                 xmlDoc_ind;
             short                 returnCode_ind;
             short                 returnMsg_ind;
  EXEC SQL END DECLARE SECTION;

             /* initialize host variable and indicators */
             strcpy(collection,"sales_ord")
             strcpy(xmlobj.name,"c:\dxx\samples\cmd\getstart.xml ");
```

```
xmlobj.name_length=strlen("c:\dxx\samples\cmd\getstart.xml ");
xmlobj.file_option=SQL_FILE_READ;
returnCode = 0;
returnMsg[0] = '\0';
collection_ind = 0;
xmlobj_ind = 0;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Call the store procedure */
EXEC SQL CALL db2xml.dxxInsertXML(:collection:collection_ind;
                :xmlobj:xmlobj_ind,
                :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Chapitre 11. Tables de gestion

Lorsqu'une base de données est activée, les tables DTD_REF et XML_USAGE sont créées. La table DTD_REF contient des informations sur toutes les définitions DTD. La table XML_USAGE stocke les informations communes pour chaque colonne compatible XML.

Les limites de paramètres répertoriées dans les tables de gestion sont détaillées dans l'«Annexe D. Les limites de l'Extension XML» à la page 297.

Table DTD_REF

L'Extension XML sert également de référentiel XML DTD. Lorsqu'une base de données est compatible XML, une table DTD_REF est créée. Chaque ligne de cette table représente une DTD accompagnée d'autres informations sur les métadonnées. Les utilisateurs peuvent accéder à cette table et insérer leurs propres DTD. Les DTD de la table DTD_REF permettent de valider des documents XML et d'aider les applications à définir un fichier DAD. Le nom de schéma associé à cette table est db2xml. Une table DTD_REF peut contenir les colonnes indiquées dans le tableau 51.

Tableau 51. Table DTD_REF

Nom de colonne	Type de données	Description
DTDID	VARCHAR(128)	Clé primaire (unique et différente de NULL). Elle sert à identifier la DTD. Lorsqu'elle est spécifiée dans le fichier DAD, celui-ci doit suivre le schéma défini par la DTD.
CONTENT	XMLCLOB	Contenu de la DTD.
USAGE_COUNT	INTEGER	Nombre de colonnes et de collections XML de la base de données utilisant la DTD pour définir leurs fichiers DAD.
AUTHOR	VARCHAR(128)	Auteur de la DTD, information facultative destinée à l'utilisateur (en entrée).
CREATOR	VARCHAR(128)	ID utilisateur auteur de la première insertion. La colonne CREATOR est facultative.
UPDATOR	VARCHAR(128)	ID utilisateur auteur de la dernière mise à jour. La colonne UPDATOR est facultative.

Restriction : La DTD peut être modifiée par l'application uniquement lorsque la valeur de USAGE_COUNT est égale à zéro.

Table XML_USAGE

Cette table stocke les informations communes à chaque colonne compatible XML. Le nom de schéma de la table XML_USAGE est db2xml et sa clé primaire est (*nom_table*, *nom_col*). Une table XML_USAGE est créée lorsque la base de données est activée avec les colonnes indiquées dans le tableau 52.

Tableau 52. Table XML_USAGE

Nom de colonne	Description
table_schema	Pour une colonne XML, nom de schéma de la table utilisateur contenant une colonne XML. Pour une collection XML, valeur ″DXX_COLL″ utilisée comme nom de schéma par défaut.
table_name	Pour une colonne XML, nom de la table utilisateur contenant une colonne XML. Pour une collection XML, valeur ″DXX_COLLECTION,″ identifiant l'entité en tant que collection.
col_name	Nom de la colonne ou de la collection XML. Il fait partie de la clé composite tout comme nom_table.
DTDID	ID de la DTD figurant dans la table DTD_REF et servant à définir le fichier DAD. Correspond à la clé associée.
DAD	Contenu du fichier DAD associé à la colonne.
default_view	Stocke le nom de la vue par défaut, le cas échéant.
trigger_suffix	Non NULL. Pour les noms de déclencheur unique.
Validation	1 pour oui, 0 pour non.
access_mode	1 pour une collection XML, 0 pour une colonne XML

Restriction : La DTD peut être modifiée par l'application uniquement lorsque la valeur de USAGE_COUNT est égale à zéro.

Chapitre 12. Informations de diagnostic

Les instructions SQL imbriquées et les appels DB2 CLI (Call Level Interface) de votre programme, y compris ceux qui appellent les fonctions UDF DB2 Extension XML, génèrent des codes indiquant si l'instruction SQL imbriquée ou l'appel DB2 CLI s'est exécuté correctement.

Votre programme peut récupérer des données qui complètent ces codes. Il s'agit notamment de données SQLSTATE et de messages d'erreur. Ces informations de diagnostic vous permettent d'isoler et de résoudre les incidents intervenant dans votre programme.

L'origine d'un incident est parfois difficile à diagnostiquer. Dans ce cas, vous devrez peut-être fournir des informations au centre de support logiciel pour identifier et résoudre cet incident. L'Extension XML comprend une fonction de trace qui enregistre l'activité de l'Extension XML. Les informations de trace peuvent constituer des entrées précieuses pour le centre de support logiciel IBM. Vous devez utiliser la fonction de trace uniquement sur recommandation du centre de support logiciel IBM.

Le présent chapitre envisage les modalités d'accès aux informations de diagnostic. Il inclut les sections suivantes :

- Traitement des codes retour UDF de l'Extension XML.
- Gestion de la fonction de trace.

Il répertorie également les codes SQLSTATE et les messages d'erreur que l'Extension XML peut renvoyer.

Traitement des codes retour UDF

Les instructions SQL imbriquées renvoient des codes dans les zones SQLCODE, SQLWARN et SQLSTATE de la structure SQLCA. Cette structure est définie dans un fichier SQLCA INCLUDE. (Pour plus d'informations sur la structure SQLCA et le fichier SQLCA INCLUDE, reportez-vous au manuel *DB2 Application Development Guide.*

Les appels DB2 CLI renvoient des valeurs SQLCODE et SQLSTATE récupérables à l'aide de la fonction SQLError. (Pour plus d'informations sur la recherche de données d'erreur à l'aide de la fonction SQLError, reportez-vous au manuel *CLI Guide and Reference.*

Une valeur SQLCODE définie à 0 indique que l'instruction a abouti (avec d'éventuelles conditions d'avertissement). Une valeur positive de SQLCODE indique que l'instruction a abouti mais qu'elle a déclenché un avertissement. (Les instructions SQL imbriquées renvoient des informations sur l'avertissement associé aux valeurs SQLCODE égales à 0 ou positives dans la zone SQLWARN.) Une valeur SQLCODE négative indique qu'une erreur s'est produite.

DB2 associe un message avec chaque valeur de SQLCODE. Si une fonction UDF Extension XML rencontre une condition d'erreur ou d'avertissement, elle renvoie les informations associées à DB2 pour insertion dans le message SQLCODE.

Les valeurs SQLSTATE contiennent des codes qui complètent les messages SQLCODE. Reportez-vous à la section «Codes SQLSTATE» à la page 243 pour obtenir la description de chaque code SQLSTATE renvoyé par l'Extension XML.

Les instructions SQL imbriquées et les appels DB2 CLI qui appellent les fonctions UDF DB2 Extension XML peuvent éventuellement renvoyer des messages SQLCODE et des valeurs SQLSTATE uniques à ces fonctions UDF, mais DB2 renvoie ces valeurs de la même manière que pour d'autres instructions SQL imbriquées ou d'autres appels DB2 CLI. Ainsi, le mode d'accès à ces valeurs est le même que pour les instructions SQL imbriquées ou les appels DB2 CLI qui n'ont pas démarré les fonctions UDF DB2 Extension XML.

Reportez-vous à la section «Codes SQLSTATE» à la page 243 pour connaître les valeurs SQLSTATE et les numéros des messages associés susceptibles d'être renvoyés par l'Extension XML. Reportez-vous à la section «Messages» à la page 247 pour plus de détails sur chaque message.

Traitement des codes retour des procédures mémorisées

L'Extension XML fournit des codes retour pour vous aider à résoudre les incidents liés aux procédures mémorisées. Lorsque vous recevez un code retour d'une procédure mémorisée, consultez le fichier suivant, qui associe le code retour à un numéro de message d'erreur de l'Extension XML et à la constante symbolique.

```
DXX_INSTALL/include/dxxrc.h
```

Vous pouvez utiliser le message d'erreur indiqué dans la section «Messages» à la page 247, et les informations de diagnostic correspondantes.

Codes SQLSTATE

Le tableau 53 répertorie les valeurs SQLSTATE renvoyées par l'Extension XML. La description de chaque valeur SQLSTATE comprend sa représentation symbolique. Le tableau présente également le numéro de message associé à chaque valeur SQLSTATE. Reportez-vous à la section «Messages» à la page 247 pour plus de détails sur chaque message.

Tableau 53. Codes SQLSTATE et numéros de message correspondants

SQLSTATE	N° du message	Description
00000	DXX*nnnn*I	Aucune erreur ne s'est produite.
01HX0	DXXD003W	L'élément ou l'attribut spécifié dans l'expression du chemin est absent du document XML.
38X00	DXXC000E	L'Extension XML n'a pas ouvert le fichier indiqué.
38X01	DXXA072E	L'Extension XML a tenté de définir automatiquement les accès à la base de données avant de l'activer, mais n'a pas trouvé les fichiers de liens.
	DXXC001E	L'Extension XML n'a pas trouvé le fichier indiqué.
38X02	DXXC002E	L'Extension XML n'a pas lu les données du fichier indiqué.
38X03	DXXC003E	L'Extension XML n'a pas écrit de données dans le fichier.
	DXXC011E	L'Extension XML n'a pas écrit de données dans le fichier de contrôle de la trace.
38X04	DXXC004E	L'Extension XML n'a pas traité le releveur de coordonnées indiqué.
38X05	DXXC005E	La taille du fichier est supérieure à celle de XMLVarchar et l'Extension XML ne peut importer toutes les données du fichier.
38X06	DXXC006E	La taille du fichier est supérieure à celle de XMLCLOB et l'Extension XML ne peut importer toutes les données du fichier.
38X07	DXXC007E	Le nombre d'octets du releveur de coordonnées LOB n'est pas identique à la taille du fichier.

Tableau 53. Codes SQLSTATE et numéros de message correspondants (suite)

SQLSTATE	N° du message	Description
38X08	DXXD001E	Une fonction scalaire d'extraction a utilisé un chemin d'emplacement à occurrences multiples. Les fonctions scalaires n'admettent pas les chemins d'emplacement à occurrences multiples.
38X09	DXXD002E	L'expression du chemin de recherche est incorrecte au niveau de la syntaxe.
38X10	DXXG002E	L'Extension XML n'a pas alloué de mémoire à partir du système d'exploitation.
38X11	DXXA009E	Cette procédure mémorisée s'applique uniquement aux colonnes XML.
38X12	DXXA010E	Lors de la tentative d'activation de la colonne, l'Extension XML n'a pas trouvé l'ID de DTD, qui désigne l'identificateur spécifié pour le DTD dans le fichier DAD (définition d'accès à un document).
38X14	DXXD000E	Il y a eu une tentative de stockage d'un document incorrect dans une table. La validation n'a pas abouti.
38X15	DXXA056E	L'élément de validation dans le fichier de définition DAD est incorrecte ou absente.
	DXXA057E	L'attribut de nom de table annexe indiqué dans le fichier DAD est incorrect ou absent.
	DXXA058E	L'attribut de nom de colonne indiqué dans le fichier DAD est incorrect ou absent.
	DXXA059E	L'attribut de type d'une colonne dans le fichier de définition d'accès à un document (DAD) est incorrect ou absent.

Tableau 53. Codes SQLSTATE et numéros de message correspondants (suite)

SQLSTATE	N° du message	Description
	DXXA060E	L'attribut path d'une colonne dans le fichier de définition d'accès à un document (DAD) est incorrect ou absent.
	DXXA061E	L'attribut multi_occurrence d'une colonne dans le fichier de définition d'accès à un document (DAD) est incorrect ou absent.
	DXXQ000E	Un élément obligatoire est absent du fichier de définition d'accès à un document (DAD).
38X16	DXXG004E	La valeur NULL d'un paramètre obligatoire a été transmise dans une procédure mémorisée XML.
38X17	DXXQ001E	L'instruction SQL de la définition d'accès à un document (DAD) ou celle qui l'a remplacée n'est pas correcte. Une instruction SELECT est indispensable pour la génération de documents XML.
38X18	DXXG001E	L'Extension XML a rencontré une erreur interne.
	DXXG006E	L'Extension XML a rencontré une erreur interne lors de l'utilisation de CLI (Call Level Interface).
38X19	DXXQ002E	Le système ne dispose pas de mémoire suffisante ou d'espace disque disponible. Aucun espace ne peut contenir les documents XML créés.
38X20	DXXQ003W	La requête SQL définie par l'utilisateur génère plus de documents XML que la limite maximale définie. Seul le nombre de documents indiqué est renvoyé.
38X21	DXXQ004E	La colonne indiquée ne fait pas partie des colonnes figurant dans les résultats de la requête SQL.
38X22	DXXQ005E	Le mappage de la requête SQL sur XML est incorrect.

Tableau 53. Codes SQLSTATE et numéros de message correspondants (suite)

SQLSTATE	N° du message	Description
38X23	DXXQ006E	Un élément attribute_node figurant dans le fichier DAD n'a pas d'attribut de nom.
38X24	DXXQ007E	L'élément attribute_node figurant dans le fichier DAD ne possède pas d'élément colonne ou de RDB_node.
38X25	DXXQ008E	Un élément text_node du fichier DAD ne possède pas d'élément colonne.
38X26	DXXQ009E	La table de résultats indiquée est introuvable dans le catalogue système.
38X27	DXXQ010E	Le RDB_node de l'élément attribute_node ou text_node doit être associé à une table.
	DXXQ011E	Le RDB_node de l'élément attribute_node ou text_node doit être associé à une colonne.
	DXXQ017E	Un document XML généré par l'Extension XML est trop volumineux pour tenir dans la colonne de la table de résultats.
38X28	DXXQ012E	Lors du traitement de la DAD, l'Extension XML n'a pas trouvé l'élément attendu.
	DXXQ016E	Toutes les tables doivent être définies dans l'élément RDB_node de l'élément supérieur du fichier DAD. Les tables des sous-éléments doivent correspondre à celles définies dans l'élément supérieur. Le nom de la table indiqué dans cet élément RDB_node ne figure pas dans l'élément supérieur.
38X29	DXXQ013E	L'élément table ou colonne doit avoir un nom dans le fichier DAD.
	DXXQ015E	La condition indiquée dans l'élément condition du fichier DAD n'est pas au bon format.

Tableau 53. Codes SQLSTATE et numéros de message correspondants (suite)

SQLSTATE	N° du message	Description
38X30	DXXQ014E	Un élément element_node du fichier DAD ne possède pas d'attribut de nom.
	DXXQ018E	La clause ORDER BY n'existe pas dans l'instruction SQL figurant dans un fichier DAD effectuant le mappage de SQL vers XML.
38X31	DXXQ019E	L'élément objids ne possède pas d'élément colonne dans le fichier DAD (définition d'accès au document) qui mappe SQL vers XML.
38X36	DXXA073E	Les accès de la base de données n'étaient pas définis lorsque l'utilisateur a tenté d'activer cette dernière.
38X37	DXXG007E	La variable d'environnement local du système d'exploitation du serveur est incohérente avec la page de codes DB2.
38X38	DXXG008E	La variable d'environnement local du système d'exploitation du serveur est introuvable dans la table des pages de codes.
38x33	DXXG005E	Ce paramètre n'est pas pris en charge dans cette version, il le sera dans la version suivante.
38x34	DXXG000E	Un nom de fichier incorrect a été spécifié.

Messages

L'Extension XML fournit des messages d'erreur pour faciliter la détermination des incidents.

Messages d'erreur

L'Extension XML génère les messages suivants lorsqu'il termine une opération ou détecte une erreur.

DXXA000I **Activation de la colonne** <*nom_colonne*> **en cours. Veuillez patienter.**

Explication : Ceci est un message d'information.

Action de l'utilisateur : Aucune action n'est requise.

DXXA001S **Erreur inattendue dans le code exécutable** <*ID_code*>**, fichier** <*nom_fichier*> **et ligne** <*numéro_ligne*>**.**

Explication : Une erreur inattendue s'est produite.

Action de l'utilisateur : Si cette erreur persiste, prenez contact avec votre centre de support logiciel. Lorsque vous signalez l'erreur, indiquez le texte intégral du message, le fichier de trace et une explication pour pouvoir reproduire l'incident.

DXXA002I **Connexion à la base de données** <*base_de_données*> **en cours.**

Explication : Ceci est un message d'information.

Action de l'utilisateur : Aucune action n'est requise.

DXXA003E **Connexion à la base de données** <*base_de_données*> **impossible.**

Explication : La base de données indiquée est peut-être inexistante ou endommagée.

Action de l'utilisateur :

1. Assurez-vous que la base de données est indiquée correctement.
2. Vérifiez que la base de données existe bien et qu'elle est accessible.
3. Déterminez si la base de données est endommagée. Dans ce cas, contactez votre administrateur de base de données pour la récupérer à partir d'une sauvegarde.

DXXA004E **Activation de la base de données** <*base_de_données*> **impossible.**

Explication : La base de données est peut-être déjà activée ou endommagée.

Action de l'utilisateur :

1. Déterminez si la base de données est activée.
2. Déterminez si la base de données est endommagée. Si c'est le cas, prenez contact avec votre administrateur de base de données pour la récupérer à partir d'une sauvegarde.

DXXA005I **Activation de la base de données** <*base_de_ données*> **en cours. Veuillez patienter.**

Explication : Ceci est un message d'information.

Action de l'utilisateur : Aucune action n'est requise.

DXXA006I **L'activation de la base de données** <*base_de_données*> **a abouti.**

Explication : Ceci est un message d'information.

Action de l'utilisateur : Aucune action n'est requise.

DXXA007E **Désactivation de la base de données** <*base_de_données*> **impossible.**

Explication : La base de données ne peut pas être désactivée par l'Extension XML si elle contient des colonnes ou des collections XML.

Action de l'utilisateur : Faites une sauvegarde de toutes les données importantes, désactivez les colonnes ou les collections XML et mettez à jour ou supprimez des tables jusqu'à ce que tous les types de données XML aient disparu de la base de données.

DXXA008I Désactivation de la colonne <*nom_colonne*> **en cours. Veuillez patienter.**

Explication : Ceci est un message d'information.

Action de l'utilisateur : Aucune action n'est requise.

DXXA009E La balise Xcolumn n'est pas mentionnée dans le fichier DAD.

Explication : Cette procédure mémorisée s'applique uniquement aux colonnes XML.

Action de l'utilisateur : Vérifiez que la balise Xcolumn est indiquée correctement dans le fichier DAD.

DXXA010E Echec de la tentative de recherche de l'ID de DTD <*dtdid*>**.**

Explication : Lors de la tentative d'activation de la colonne, l'Extension XML n'a pas trouvé l'identificateur de la DTD (définition du type de document) dans le fichier DAD (définition d'accès à un document).

Action de l'utilisateur : Assurez-vous que la valeur correcte de l'ID DTD est indiquée dans le fichier DAD.

DXXA011E Echec de la tentative d'insertion dans la table DB2XML.XML_USAGE.

Explication : Lors de la tentative d'activation de la colonne, l'Extension XML n'a pas pu insérer d'enregistrement dans la table DB2XML.XML_USAGE.

Action de l'utilisateur : Vérifiez que table DB2XML.XML_USAGE existe et que la table ne contient pas d'enregistrement du même nom.

DXXA012E Echec de la tentative de mise à jour de la table DB2XML.DTD_REF.

Explication : Lors de la tentative d'activation de la colonne, l'Extension XML n'a pas mis à jour la table DB2XML.DTD_REF.

Action de l'utilisateur : Vérifiez l'existence de la table DB2XML.DTD_REF. Déterminez si la table est endommagée ou si l'ID utilisateur de niveau administrateur est associé aux droits requis pour mettre à jour la table.

DXXA013E Echec de la tentative de modification de la table <*nom_table*>**.**

Explication : Lors de la tentative d'activation de la colonne, l'Extension XML n'a pas modifié la table indiquée.

Action de l'utilisateur : Vérifiez les privilèges requis pour pouvoir modifier la table.

DXXA014E La colonne d'ID racine indiquée <*id_racine*> **n'est pas une clé primaire unique de la table** <*nom_table*>**.**

Explication : L'ID racine indiqué n'est pas une clé ou n'est pas une clé unique de la table *nom_table*.

Action de l'utilisateur : Vérifiez que l'ID racine indiqué est bien une clé primaire unique de la table.

DXXA015E La colonne DXXROOT_ID existe déjà dans la table <*nom_table*>**.**

Explication : La colonne DXXROOT_ID existe, mais n'a pas été créée par l'Extension XML.

Action de l'utilisateur : Indiquez une colonne primaire pour l'option ID racine lorsque vous activez une colonne, à l'aide d'un autre nom de colonne.

DXXA016E La table d'entrée <*nom_table*> **n'existe pas.**

Explication : L'Extension XML n'a pas trouvé dans le catalogue système la table indiquée.

Action de l'utilisateur : Vérifiez que la table existe dans la base de données et qu'elle est correctement indiquée.

DXXA017E La colonne d'entrée <*nom_colonne*> **n'existe pas dans la table** <*nom_table*>**.**

Explication : L'Extension XML n'a pas trouvé cette colonne dans le catalogue système.

Action de l'utilisateur : Vérifiez que la colonne existe dans une table utilisateur.

DXXA018E La colonne indiquée n'est pas activée pour les données XML.

Explication : L'Extension XML n'a pas trouvé dans la table DB2XML.XML_USAGE la colonne à désactiver, ce qui signifie qu'elle n'est pas activée. Si la colonne n'a pas été activée pour XML, vous n'avez pas besoin de la désactiver.

Action de l'utilisateur : Aucune action n'est requise.

DXXA019E L'un des paramètres d'entrée obligatoires pour l'activation de la colonne est de valeur NULL.

Explication : L'un des paramètres d'entrée requis pour la procédure mémorisée enable_column() est de valeur NULL.

Action de l'utilisateur : Vérifiez tous les paramètres d'entrée de la procédure mémorisée enable_column().

DXXA020E Les colonnes sont introuvables dans la table <*nom_table*>**.**

Explication : Lors de la tentative de création d'une vue par défaut, l'Extension XML n'a pas trouvé de colonne dans la table indiquée.

Action de l'utilisateur : Assurez-vous que les noms de tables et de colonnes sont correctement indiqués.

DXXA021E Création de la vue par défaut <*vue_par_défaut*> **impossible.**

Explication : Lors de la tentative d'activation de la colonne, l'Extension XML n'a pas créé la vue indiquée.

Action de l'utilisateur : Vérifiez que le nom de la vue par défaut est unique. Si une vue du même nom existe déjà, indiquez un nom unique pour la vue par défaut.

DXXA022I Colonne <*nom_colonne*> **activée.**

Explication : Ceci est un message d'information.

Action de l'utilisateur : Aucune action n'est requise.

DXXA023E Fichier DAD introuvable.

Explication : Lors de la tentative de désactivation d'une colonne, l'Extension XML n'a pas trouvé le fichier DAD (définition d'accès à un document).

Action de l'utilisateur : Vérifiez que vous avez correctement indiqué le nom de la base de données, le nom de la table ou de la colonne.

DXXA024E L'Extension XML a rencontré une erreur interne lors de l'accès aux tables du catalogue système.

Explication : L'Extension XML n'a pas pu accéder à la table du catalogue système.

Action de l'utilisateur : Vérifiez que la base de données est à l'état stable.

DXXA025E Suppression de la vue par défaut <*vue_par_défaut*> **impossible.**

Explication : Lors de la tentative de désactivation d'une colonne, l'Extension XML n'a pas supprimé la vue par défaut.

Action de l'utilisateur : Vérifiez si l'ID utilisateur de l'administrateur pour l'Extension XML dispose des droits requis pour supprimer la vue par défaut.

DXXA026E Suppression de la table annexe <*table_annexe*> **impossible.**

Explication : Lors de la tentative de désactivation d'une colonne, l'Extension XML n'a pas supprimé la table indiquée.

Action de l'utilisateur : Vérifiez que l'ID utilisateur de l'administrateur pour l'Extension XML dispose des droits requis pour supprimer la table.

DXXA027E Désactivation de la colonne impossible.

Explication : L'Extension XML n'a pas désactivé la colonne en raison de l'échec d'un déclencheur interne.

Action de l'utilisateur : Créez un fichier de trace à l'aide de la fonction de trace, puis essayez de résoudre l'incident. S'il persiste, prenez contact avec votre centre de support logiciel et transmettez-lui le fichier de trace.

DXXA028E Désactivation de la colonne impossible.

Explication : L'Extension XML n'a pas désactivé la colonne en raison de l'échec d'un déclencheur interne.

Action de l'utilisateur : Créez un fichier de trace à l'aide de la fonction de trace, puis essayez de résoudre l'incident. S'il persiste, prenez contact avec votre centre de support logiciel et transmettez-lui le fichier de trace.

DXXA029E Désactivation de la colonne impossible.

Explication : L'Extension XML n'a pas désactivé la colonne en raison de l'échec d'un déclencheur interne.

Action de l'utilisateur : Créez un fichier de trace à l'aide de la fonction de trace, puis essayez de résoudre l'incident. S'il persiste, prenez contact avec votre centre de support logiciel et transmettez-lui le fichier de trace.

DXXA030E Désactivation de la colonne impossible.

Explication : L'Extension XML n'a pas désactivé la colonne en raison de l'échec d'un déclencheur interne.

Action de l'utilisateur : Créez un fichier de trace à l'aide de la fonction de trace, puis essayez de résoudre l'incident. S'il persiste, prenez contact avec votre centre de support logiciel et transmettez-lui le fichier de trace.

DXXA031E La réinitialisation de la valeur de colonne DXXROOT_ID (retour à la valeur NULL) dans la table d'application est impossible.

Explication : Lors de la tentative de désactivation d'une colonne, l'Extension XML n'a pas défini la valeur de DXXROOT_ID sur NULL dans la table d'application.

Action de l'utilisateur : Vérifiez que l'ID utilisateur de l'administrateur pour l'Extension XML dispose des droits nécessaires pour modifier la table d'application.

DXXA032E Echec de la diminution de la valeur de USAGE_COUNT dans la table DB2XML.XML_USAGE.

Explication : Lors de la tentative de désactivation de la colonne, l'Extension XML n'a pas réduit la valeur de la colonne USAGE_COUNT d'un incrément.

Action de l'utilisateur : Vérifiez que la table DB2XML.XML_USAGE existe et que l'ID utilisateur de l'administrateur pour l'Extension XML dispose des droits requis pour mettre à jour cette table.

DXXA033E Echec de la tentative de suppression d'une ligne de la table DB2XML.XML_USAGE.

Explication : Lors de la tentative de désactivation d'une colonne, l'Extension XML n'a pas supprimé la ligne qui lui est associée dans la table DB2XML.XML_USAGE.

Action de l'utilisateur : Vérifiez que la table DB2XML.XML_USAGE existe et que l'ID utilisateur de l'administrateur pour l'Extension XML dispose des droits requis pour mettre à jour cette table.

DXXA034I L'Extension XML a désactivé la colonne <*nom_colonne*>**.**

Explication : Ceci est un message d'information.

Action de l'utilisateur : Aucune action n'est requise.

DXXA035I Désactivation de la base de données <*base_de_données*> **en cours par l'Extension XML. Veuillez patienter.**

Explication : Ceci est un message d'information.

Action de l'utilisateur : Aucune action n'est requise.

DXXA036I L'Extension XML a désactivé la base de données <*base_de_données*>**.**

Explication : Ceci est un message d'information.

Action de l'utilisateur : Aucune action n'est requise.

DXXA037E Le nom d'espace table indiqué comporte plus de 18 caractères.

Explication : Le nom d'espace table ne doit pas dépasser 18 caractères alphanumériques.

Action de l'utilisateur : Indiquez un nom comportant moins de 18 caractères.

DXXA038E Le nom de vue par défaut indiqué comporte plus de 18 caractères.

Explication : Le nom de vue par défaut ne doit pas dépasser 18 caractères alphanumériques.

Action de l'utilisateur : Indiquez un nom comportant moins de 18 caractères.

DXXA039E Le nom ROOT_ID indiqué comporte plus de 18 caractères.

Explication : Le nom ROOT_ID ne doit pas dépasser 18 caractères alphanumériques.

Action de l'utilisateur : Indiquez un nom comportant moins de 18 caractères.

DXXA046E Création de la table annexe <*table_annexe*> **impossible.**

Explication : Lors de la tentative d'activation d'une colonne, l'Extension XML n'a pas créé la table annexe indiquée.

Action de l'utilisateur : Vérifiez que l'ID utilisateur de l'administrateur pour l'Extension XML dispose des droits requis pour créer la table annexe.

DXXA047E Activation de la colonne impossible.

Explication : L'Extension XML n'a pas activé la colonne en raison de l'échec d'un déclencheur interne.

Action de l'utilisateur : Créez un fichier de trace à l'aide de la fonction de trace, puis essayez de résoudre l'incident. S'il persiste, prenez contact avec votre centre de support logiciel et transmettez-lui le fichier de trace.

DXXA048E Activation de la colonne impossible.

Explication : L'Extension XML n'a pas activé la colonne en raison de l'échec d'un déclencheur interne.

Action de l'utilisateur : Créez un fichier de trace à l'aide de la fonction de trace, puis essayez de résoudre l'incident. S'il persiste, prenez contact avec votre centre de support logiciel et transmettez-lui le fichier de trace.

DXXA049E Activation de la colonne impossible.

Explication : L'Extension XML n'a pas activé la colonne en raison de l'échec d'un déclencheur interne.

Action de l'utilisateur : Créez un fichier de trace à l'aide de la fonction de trace, puis essayez de résoudre l'incident. S'il persiste, prenez contact avec votre centre de support logiciel et transmettez-lui le fichier de trace.

DXXA050E Activation de la colonne impossible.

Explication : L'Extension XML n'a pas activé la colonne en raison de l'échec d'un déclencheur interne.

Action de l'utilisateur : Créez un fichier de trace à l'aide de la fonction de trace, puis essayez de résoudre l'incident. S'il persiste, prenez contact avec votre centre de support logiciel et transmettez-lui le fichier de trace.

DXXA051E Désactivation de la colonne impossible.

Explication : L'Extension XML n'a pas désactivé la colonne en raison de l'échec d'un déclencheur interne.

Action de l'utilisateur : Créez un fichier de trace à l'aide de la fonction de trace, puis essayez de résoudre l'incident. S'il persiste, prenez contact avec votre centre de support logiciel et transmettez-lui le fichier de trace.

DXXA052E Désactivation de la colonne impossible.

Explication : L'Extension XML n'a pas désactivé la colonne en raison de l'échec d'un déclencheur interne.

Action de l'utilisateur : Créez un fichier de trace à l'aide de la fonction de trace, puis essayez de résoudre l'incident. S'il persiste, prenez contact avec votre centre de support logiciel et transmettez-lui le fichier de trace.

DXXA053E Activation de la colonne impossible.

Explication : L'Extension XML n'a pas activé la colonne en raison de l'échec d'un déclencheur interne.

Action de l'utilisateur : Créez un fichier de trace à l'aide de la fonction de trace, puis essayez de résoudre l'incident. S'il persiste, prenez contact avec votre centre de support logiciel et transmettez-lui le fichier de trace.

DXXA054E Activation de la colonne impossible.

Explication : L'Extension XML n'a pas activé la colonne en raison de l'échec d'un déclencheur interne.

Action de l'utilisateur : Créez un fichier de trace à l'aide de la fonction de trace, puis essayez de résoudre l'incident. S'il persiste, prenez contact avec votre centre de support logiciel et transmettez-lui le fichier de trace.

DXXA056E La valeur de validation *<valeur_validation>* **indiquée dans le fichier DAD est incorrecte.**

Explication : L'élément validation dans le fichier de définition DAD est incorrect ou absent.

Action de l'utilisateur : Vérifiez que l'élément validation est indiqué correctement dans le fichier DAD.

DXXA057E Le nom de table annexe *<nom_table_annexe>* **figurant dans la DAD est incorrect.**

Explication : L'attribut de nom de table annexe indiqué dans le fichier DAD est incorrect ou absent.

Action de l'utilisateur : Vérifiez que l'attribut de nom de table annexe est correctement indiqué dans le fichier DAD.

DXXA058E Le nom de colonne *<nom_colonne>* **figurant dans le fichier DAD est incorrect.**

Explication : L'attribut de nom de colonne indiqué dans le fichier DAD est incorrect ou absent.

Action de l'utilisateur : Vérifiez que l'attribut de nom de colonne est correctement indiqué dans le fichier DAD.

DXXA059E Le type <*type_colonne*> **de la colonne** <*nom_colonne*> **du fichier DAD est incorrect.**

Explication : L'attribut de type d'une colonne dans le fichier de définition d'accès à un document (DAD) est incorrect ou absent.

Action de l'utilisateur : Vérifiez que l'attribut de type de colonne est correctement indiqué dans le fichier DAD.

DXXA060E L'attribut path <*chemin_emplacement*> **de** <*nom_colonne*> **dans le fichier DAD est incorrect.**

Explication : L'attribut path d'une colonne dans le fichier de définition d'accès à un document (DAD) est incorrect ou absent.

Action de l'utilisateur : Vérifiez que l'attribut path d'une colonne est indiqué correctement dans le fichier DAD.

DXXA061E L'attribut multi_occurrence <*multi_occurrence*> **de** <*nom_colonne*> **dans le fichier DAD est incorrect.**

Explication : L'attribut multi_occurrence d'une colonne dans le fichier de définition d'accès à un document (DAD) est incorrect ou absent.

Action de l'utilisateur : Vérifiez que l'attribut multi_occurrence d'une colonne est correctement indiqué dans le fichier DAD.

DXXA062E Impossible d'extraire le numéro de colonne de <*nom_colonne*> **dans la table** <*nom_table*>**.**

Explication : L'Extension XML n'a pas récupéré le numéro de colonne de *nom_colonne* dans la table *nom_table* du catalogue système.

Action de l'utilisateur : Vérifiez que la table d'application est correctement définie.

DXXA063I Activation de la collection <*nom_collection*> **en cours. Veuillez patienter.**

Explication : Ceci est un message d'information.

Action de l'utilisateur : Aucune action n'est requise.

DXXA064I Désactivation de la collection <*nom_collection*> **en cours. Veuillez patienter.**

Explication : Ceci est un message d'information.

Action de l'utilisateur : Aucune action n'est requise.

DXXA065E Echec de l'appel de la procédure mémorisée <*nom_procédure*>**.**

Explication : Vérifiez dans la bibliothèque partagée db2xml si les droits d'accès sont corrects.

Action de l'utilisateur : Assurez-vous que le client dispose des droits requis pour exécuter la procédure mémorisée.

DXXA066I L'Extension XML a désactivé la collection <*nom_collection*>**.**

Explication : Ceci est un message d'information.

Action de l'utilisateur : Aucune action n'est requise.

DXXA067I L'Extension XML a activé la collection <*nom_collection*>**.**

Explication : Ceci est un message d'information.

Action de l'utilisateur : Aucune action n'est requise.

DXXA068I L'Extension XML a activé la fonction de trace.

Explication : Ceci est un message d'information.

Action de l'utilisateur : Aucune action n'est requise.

DXXA069I L'Extension XML a désactivé la fonction de trace.

Explication : Ceci est un message d'information.

Action de l'utilisateur : Aucune action n'est requise.

DXXA070W La base de données a déjà été activée.

Explication : La commande d'activation de la base de données a été exécutée sur une base de données activée.

Action de l'utilisateur : Aucune action n'est requise.

DXXA071W La base de données a déjà été désactivée.

Explication : La commande de désactivation de la base de données a été exécutée sur une base de données désactivée.

Action de l'utilisateur : Aucune action n'est requise.

DXXA072E L'Extension XML n'a pas pu trouver les fichiers de liens. Définissez les accès de la base de données avant de l'activer.

Explication : L'Extension XML a tenté de définir automatiquement les accès à la base de données avant de l'activer, mais n'a pas pu trouver les fichiers de liens.

Action de l'utilisateur : Définissez les accès de la base de données avant de l'activer.

DXXA073E Les accès de la base de données ne sont pas définis. Définissez-les avant d'activer la base.

Explication : Les accès de la base de données n'étaient pas définis lorsque l'utilisateur a tenté d'activer cette dernière.

Action de l'utilisateur : Définissez les accès de la base de données avant de l'activer.

DXXA074E Type de paramètre incorrect. La procédure mémorisée attend un paramètre de type STRING.

Explication : La procédure mémorisée attend un paramètre de type STRING.

Action de l'utilisateur : Déclarez un paramètre d'entrée de type STRING.

DXXA075E Type de paramètre incorrect. Le paramètre d'entrée doit être de type LONG.

Explication : La procédure mémorisée attend un paramètre d'entrée de type LONG.

Action de l'utilisateur : Déclarez un paramètre d'entrée de type LONG.

DXXA076E L'ID d'instance de trace de l'Extension XML est incorrect.

Explication : Impossible de démarrer la fonction de trace avec l'ID d'instance fourni.

Action de l'utilisateur : Assurez-vous que l'ID d'instance est un ID utilisateur AS/400 valide.

DXXC000E Ouverture du fichier spécifié impossible.

Explication : L'Extension XML n'a pas ouvert le fichier indiqué.

Action de l'utilisateur : Vérifiez que l'ID utilisateur de l'application dispose des droits requis pour accéder au fichier en lecture et en écriture.

DXXC001E Fichier spécifié introuvable.

Explication : L'Extension XML n'a pas trouvé le fichier indiqué.

Action de l'utilisateur : Vérifiez que le fichier existe et qu'il est correctement désigné.

DXXC002E Fichier illisible.

Explication : L'Extension XML n'a pas lu les données du fichier indiqué.

Action de l'utilisateur : Vérifiez que l'ID utilisateur de l'application dispose des droits requis pour accéder en lecture au fichier.

DXXC003E Ecriture impossible sur le fichier spécifié.

Explication : L'Extension XML n'a pas écrit de données dans le fichier.

Action de l'utilisateur : Vérifiez que l'ID utilisateur de l'application dispose des droits requis pour accéder en écriture au fichier ou que le système de fichiers comporte un espace suffisant.

DXXC004E Traitement du releveur de coordonnées LOB impossible : cr=<*cr_releveur*>**.**

Explication : L'Extension XML n'a pas traité le releveur de coordonnées indiqué.

Action de l'utilisateur : Vérifiez que le releveur de coordonnées LOB est défini correctement.

DXXC005E La taille du fichier d'entrée est supérieure à celle de XMLVarchar.

Explication : La taille du fichier est supérieure à celle de XMLVarchar et l'Extension XML ne peut pas importer toutes les données du fichier.

Action de l'utilisateur : Utilisez le type de colonne XMLCLOB.

DXXC006E Le fichier d'entrée dépasse les limites LOB DB2.

Explication : La taille du fichier est supérieure à celle de XMLCLOB et l'Extension XML ne peut importer toutes les données du fichier.

Action de l'utilisateur : Décomposez le fichier en objets de plus petite taille ou utilisez une collection XML.

DXXC007E Impossible de copier les données du fichier dans le releveur de coordonnées LOB.

Explication : Le nombre d'octets du releveur de coordonnées LOB n'est pas identique à la taille du fichier.

Action de l'utilisateur : Vérifiez que le releveur de coordonnées LOB est défini correctement.

DXXC008E Suppression du fichier <*nom_fichier*> **impossible.**

Explication : L'accès partagé au fichier n'est pas respecté ou le fichier est encore ouvert.

Action de l'utilisateur : Fermez le fichier ou arrêtez tout processus qui l'utilise. Vous devez peut-être arrêter puis redémarrer DB2.

DXXC009E Création du fichier impossible dans le répertoire <*répertoire*>**.**

Explication : L'Extension XML n'a pas créé de fichier dans le répertoire indiqué.

Action de l'utilisateur : Vérifiez que le répertoire existe, que l'ID utilisateur de l'application dispose des droits requis pour accéder en écriture au fichier, ou que le système de fichiers comporte un espace suffisant.

DXXC010E Erreur lors de l'écriture dans le fichier <*nom_fichier*>**.**

Explication : Une erreur s'est produite lors de l'écriture dans le fichier *nom_fichier*.

Action de l'utilisateur : Vérifiez que le système de fichiers comporte un espace suffisant.

DXXC011E Ecriture impossible sur le fichier de contrôle de la trace.

Explication : L'Extension XML n'a pas écrit de données dans le fichier de contrôle de la trace.

Action de l'utilisateur : Vérifiez que l'ID utilisateur de l'application dispose des droits requis pour accéder en écriture au fichier ou que

le système de fichiers comporte un espace suffisant.

DXXC012E Création du fichier temporaire impossible.

Explication : Le fichier ne peut pas être créé dans le répertoire système temporaire.

Action de l'utilisateur : Vérifiez que l'ID utilisateur de l'application dispose des droits requis pour accéder en écriture au répertoire temporaire du système de fichiers, ou que le système de fichiers comporte un espace suffisant.

DXXD000E Rejet d'un document XML incorrect.

Explication : Il y a eu une tentative de stockage d'un document incorrect dans une table. La validation n'a pas abouti.

Action de l'utilisateur : Vérifiez le document et la DTD associée à l'aide d'un éditeur permettant d'afficher les caractères incorrects invisibles. Pour corriger cette erreur, désactivez la validation dans le fichier DAD.

DXXD001E Le chemin d'emplacement *<chemin_emplacement>* **a des occurrences multiples.**

Explication : Une fonction scalaire d'extraction a utilisé un chemin d'emplacement à occurrences multiples. Les fonctions scalaires n'admettent pas les chemins d'emplacement à occurrences multiples.

Action de l'utilisateur : Utilisez une fonction de table (ajoutez un 's' à la fin du nom de la fonction scalaire).

DXXD002E Une erreur de syntaxe s'est produite près de la position *<position>* **dans le chemin de recherche.**

Explication : La syntaxe du chemin de recherche est incorrecte.

Action de l'utilisateur : Corrigez l'argument du chemin de recherche de la requête. Reportez-vous à la documentation sur la syntaxe des chemins.

DXXD003W Chemin d'accès introuvable. La valeur NULL est renvoyée.

Explication : L'élément ou l'attribut indiqué dans le chemin est absent du document XML.

Action de l'utilisateur : Vérifiez que le chemin indiqué est correct.

DXXG000E Le nom de fichier *<nom_fichier>* **est incorrect.**

Explication : Un nom de fichier incorrect a été indiqué.

Action de l'utilisateur : Indiquez un nom de fichier correct et réessayez.

DXXG001E Erreur interne dans le code exécutable *<ID_code>***, fichier** *<nom_fichier>* **et ligne** *<numéro_ligne>***.**

Explication : L'Extension XML a rencontré une erreur interne.

Action de l'utilisateur : Prenez contact avec votre centre de support logiciel. Lorsque vous signalez l'erreur, veillez à inclure tous les messages, le fichier de trace et les indications permettant de reproduire l'incident.

DXXG002E Mémoire insuffisante.

Explication : L'Extension XML n'a pas alloué de mémoire à partir du système d'exploitation.

Action de l'utilisateur : Fermez quelques applications et réessayez. Si l'incident persiste, reportez-vous à la documentation du système d'exploitation pour obtenir de l'aide. Certains systèmes d'exploitation peuvent nécessiter le réamorçage du système pour résoudre l'incident.

DXXG004E Paramètre de valeur NULL incorrect.

Explication : La valeur NULL d'un paramètre obligatoire a été transmise dans une procédure mémorisée XML.

Action de l'utilisateur : Vérifiez tous les paramètres obligatoires de la liste d'arguments destinés à l'appel de la procédure mémorisée.

DXXG005E Paramètre non pris en charge.

Explication : Ce paramètre n'est pas pris en charge dans cette version, il le sera dans la version suivante.

Action de l'utilisateur : Indiquez la valeur NULL pour ce paramètre.

DXXG006E Erreur interne CLISTATE=<*état_CLI*>**, CR=**<*cr_CLI*>**, compilation** <*id_compilation*>**, fichier** <*nom_fichier*>**, ligne** <*numéro_ligne*> **CLIMSG=**<*msg_CLI*>**.**

Explication : L'Extension XML a rencontré une erreur interne lors de l'utilisation de CLI (Call Level Interface).

Action de l'utilisateur : Prenez contact avec votre centre de support logiciel. Cette erreur peut être provoquée par des données d'entrée utilisateur incorrectes. Lorsque vous signalez l'erreur, veillez à inclure tous les messages de sortie, le journal de trace et les indications permettant de reproduire l'incident. Envoyez si possible les fichiers DAD, les documents XML et les définitions de table applicables.

DXXG007E La variable d'environnement local <*variable d'environnement local*> **est incohérente avec la page de codes DB2** <*page_codes*>**.**

Explication : La variable d'environnement local du système d'exploitation du serveur est incohérente avec la page de codes DB2.

Action de l'utilisateur : Corrigez la variable d'environnement du système d'exploitation du serveur et redémarrez DB2.

DXXG008E La variable d'environnement local <*variable d'environnement local*> **n'est pas prise en charge.**

Explication : La variable d'environnement local du système d'exploitation du serveur est introuvable dans la table des pages de codes.

Action de l'utilisateur : Corrigez la variable d'environnement du système d'exploitation du serveur et redémarrez DB2.

DXXQ000E <*Elément*> **manquant dans le fichier DAD.**

Explication : Un élément obligatoire est absent du fichier de définition d'accès à un document (DAD).

Action de l'utilisateur : Ajoutez l'élément manquant dans le fichier DAD.

DXXQ001E Instruction SQL incorrecte pour la génération XML.

Explication : L'instruction SQL de la définition d'accès à un document (DAD) ou celle qui l'a remplacée n'est pas correcte. Une instruction SELECT est indispensable pour la génération de documents XML.

Action de l'utilisateur : Corrigez l'instruction SQL.

DXXQ002E Génération d'espace de stockage destiné aux documents XML impossible.

Explication : Le système ne dispose pas de mémoire suffisante ou d'espace disque disponible. Aucun espace ne peut contenir les documents XML créés.

Action de l'utilisateur : Limitez le nombre de documents à générer. Réduisez la taille de chaque document en supprimant des noeuds d'attributs ou d'éléments inutiles dans le fichier DAD.

DXXQ003W Les résultats dépassent la limite maximale définie.

Explication : La requête SQL définie par l'utilisateur génère plus de documents XML que la limite maximale définie. Seul le nombre de documents indiqué est renvoyé.

Action de l'utilisateur : Aucune action n'est requise. Si tous les documents sont nécessaires, indiquez zéro comme nombre maximal de documents.

DXXQ004E La colonne <*nom_colonne*> **ne figure pas dans les résultats de la requête.**

Explication : La colonne indiquée ne fait pas partie des colonnes figurant dans les résultats de la requête SQL.

Action de l'utilisateur : Dans le fichier DAD, remplacez le nom de colonne indiqué par l'un des noms de colonnes figurant dans les résultats de la requête SQL. Vous pouvez également modifier la requête SQL pour que ses résultats intègrent la colonne indiquée.

DXXQ004W ID de la DTD introuvable dans la DAD.

Explication : Dans la définition DAD, VALIDATION a pour valeur YES mais l'ID DTD n'est pas indiqué. Aucun contrôle de validation n'est effectué.

Action de l'utilisateur : Aucune action n'est requise. Si la validation est nécessaire, indiquez l'ID DTD dans le fichier DAD.

DXXQ005E Mappage relationnel incorrect. L'élément <*nom_élément*> **se situe à un niveau inférieur à celui de sa colonne enfant** <*nom_colonne*>**.**

Explication : Le mappage de la requête SQL sur XML est incorrect.

Action de l'utilisateur : Vérifiez que les colonnes de résultats de la requête SQL sont dans un ordre descendant par rapport à la hiérarchie relationnelle. Vérifiez également qu'il existe une clé candidate composée d'une seule colonne au début de chaque niveau. Si une telle clé n'est pas disponible dans une table, la requête doit en générer une pour la table à l'aide d'une expression de table et de la fonction intégrée DB2 generate_unique().

DXXQ006E Il existe un élément attribute_node sans nom.

Explication : Un élément attribute_node figurant dans le fichier DAD n'a pas d'attribut de nom.

Action de l'utilisateur : Vérifiez que tous les éléments attribute_node ont un nom dans le fichier DAD.

DXXQ007E L'élément attribute_node <*nom_attribut*> **ne possède pas d'élément colonne ou de noeud RDB.**

Explication : L'élément attribute_node figurant dans le fichier DAD ne possède pas d'élément colonne ou de noeud RDB.

Action de l'utilisateur : Vérifiez que tous les éléments attribute_node ont un élément colonne ou un noeud RDB dans le fichier DAD.

DXXQ008E Il existe un élément text_node qui ne possède pas d'élément colonne.

Explication : Un élément text_node du fichier DAD ne possède pas d'élément colonne.

Action de l'utilisateur : Vérifiez que tous les éléments text_node ont un élément colonne dans le fichier DAD.

DXXQ009E La table de résultats <*nom_table*> **n'existe pas.**

Explication : La table de résultats indiquée est introuvable dans le catalogue système.

Action de l'utilisateur : Créez la table de résultats avant d'appeler la procédure mémorisée.

DXXQ010E Le fichier DAD ne contient pas de table associée à l'élément RDB_node de <*nom_noeud*>**.**

Explication : Le noeud RDB de l'élément attribute_node ou text_node doit être associé à une table.

Action de l'utilisateur : Indiquez la table correspondant au noeud RDB pour l'élément attribute_node ou text_node figurant dans le fichier DAD.

DXXQ011E Le fichier DAD ne contient pas de colonne associée à l'élément RDB_node de <*nom_noeud*>**.**

Explication : Le noeud RDB de l'élément attribute_node ou text_node doit être associé à une colonne.

Action de l'utilisateur : Indiquez la colonne correspondant au noeud RDB pour l'élément attribute_node ou text_node figurant dans le fichier DAD.

DXXQ012E Erreurs dans la DAD.

Explication : Lors du traitement de la DAD, l'Extension XML n'a pas trouvé l'élément attendu.

Action de l'utilisateur : Vérifiez que la DAD est un document XML valide et qu'elle contient tous les éléments requis par la DTD associée. Pour plus d'informations sur la DTD de la DAD, consultez la documentation relative à l'Extension XML.

DXXQ013E Le fichier DAD ne contient pas de nom associé à l'élément table ou colonne.

Explication : L'élément table ou colonne doit avoir un nom dans le fichier DAD.

Action de l'utilisateur : Indiquez le nom de l'élément table ou colonne dans le fichier DAD.

DXXQ014E Il existe un élément element_node sans nom.

Explication : Un élément element_node du fichier DAD ne possède pas d'attribut de nom.

Action de l'utilisateur : Vérifiez que tous les éléments element_node ont un nom dans le fichier DAD.

DXXQ015E Le format de condition est incorrect.

Explication : La condition indiquée dans l'élément condition du fichier DAD n'est pas au bon format.

Action de l'utilisateur : Vérifiez que le format de la condition est correct.

DXXQ016E Dans cet élément RDB_node, le nom de la table n'est pas défini dans l'élément supérieur du fichier DAD.

Explication : Toutes les tables doivent être définies dans l'élément RDB_node de l'élément supérieur du fichier DAD. Les tables des sous-éléments doivent correspondre à celles définies dans l'élément supérieur. Le nom de la table indiqué dans cet élément RDB_node ne figure pas dans l'élément supérieur.

Action de l'utilisateur : Vérifiez que la table de l'élément RDB_node est définie dans l'élément supérieur du fichier DAD.

DXXQ017E La colonne de la table de résultats <*nom_table*> **est trop petite.**

Explication : Un document XML généré par l'Extension XML est trop volumineux pour tenir dans la colonne de la table de résultats.

Action de l'utilisateur : Supprimez la table de résultats. Créez une autre table de résultats avec une colonne plus large. Exécutez à nouveau la procédure mémorisée.

DXXQ018E Clause ORDER BY manquante dans l'instruction SQL.

Explication : La clause ORDER BY n'existe pas dans l'instruction SQL figurant dans un fichier DAD effectuant le mappage de SQL vers XML.

Action de l'utilisateur : Editez le fichier DAD. Ajoutez une clause ORDER BY contenant les colonnes identifiant l'entité.

DXXQ019E Le fichier DAD ne contient pas d'élément colonne associé à l'élément objids.

Explication : L'élément objids ne possède pas d'élément colonne dans le fichier DAD qui mappe SQL vers XML.

Action de l'utilisateur : Editez le fichier DAD. Ajoutez les colonnes clés en tant que sous-éléments de l'élément objids.

DXXQ020I La génération XML a abouti.

Explication : Les documents XML requis ont été générés à partir de la base de données.

Action de l'utilisateur : Aucune action n'est requise.

DXXQ021E La table <*nom_table*> **ne comporte pas de colonne** <*nom_colonne*>**.**

Explication : La table ne contient pas la colonne indiquée dans la base de données.

Action de l'utilisateur : Indiquez un autre nom de colonne dans le fichier DAD ou ajoutez la colonne indiquée dans la base de données de la table.

DXXQ022E La colonne <*nom_colonne*> **de la table** <*nom_table*> **doit être de type** <*nom_type*>**.**

Explication : Le type de colonne est incorrect.

Action de l'utilisateur : Corrigez le type de colonne dans le fichier DAD.

DXXQ023E La colonne <*nom_colonne*> **de la table** <*nom_table*> **ne doit pas être de longueur supérieure à** <*longueur*>**.**

Explication : La longueur de colonne définie dans la DAD est excessive.

Action de l'utilisateur : Corrigez la longueur de colonne dans la DAD.

DXXQ024E Création de la table <*nom_table*> **impossible.**

Explication : La table indiquée ne peut pas être créée.

Action de l'utilisateur : Vérifiez que l'ID utilisateur qui crée la table dispose des droits requis pour créer une table dans la base de données.

DXXQ025I La décomposition XML a abouti.

Explication : Un document XML a été décomposé et stocké dans une collection.

Action de l'utilisateur : Aucune action n'est requise.

DXXQ026E Les données XML <*nom_xml*> **sont trop volumineuses pour pouvoir s'inscrire dans la colonne** <*nom_colonne*>**.**

Explication : Les données XML désignées sont trop volumineuses pour s'inscrire dans la colonne indiquée.

Action de l'utilisateur : Augmentez la longueur de la colonne à l'aide de l'instruction ALTER TABLE, ou réduisez la taille des données en modifiant le document XML.

DXXQ028E Collection <*nom_collection*> **introuvable dans la table XML_USAGE.**

Explication : Aucun enregistrement de la collection n'est trouvé dans la table XML_USAGE.

Action de l'utilisateur : Vérifiez que vous avez activé la collection.

DXXQ029E DAD introuvable dans la table XML_USAGE. Collection : *<nom_collection>*.

Explication : Aucun enregistrement DAD pour la collection n'est trouvé dans la table XML_USAGE.

Action de l'utilisateur : Vérifiez que vous avez correctement activé la collection.

DXXQ030E Remplacement XML incorrect.

Explication : La valeur XML_override est indiquée de façon incorrecte dans la procédure mémorisée.

Action de l'utilisateur : Vérifiez que la syntaxe de la valeur XML_override est correcte.

DXXQ031E Le nom de la table ne doit pas dépasser la longueur maximale admise dans DB2.

Explication : Le nom de table indiqué par l'élément condition dans la DAD est trop long.

Action de l'utilisateur : Corrigez la longueur du nom de table dans la DAD.

DXXQ032E Le nom de la colonne ne doit pas dépasser la longueur maximale admise dans DB2.

Explication : Le nom de colonne indiqué par l'élément condition dans la DAD est trop long.

Action de l'utilisateur : Corrigez la longueur du nom de colonne dans la DAD.

DXXQ033E Identificateur non valide : *<identificateur>*

Explication : La chaîne n'est pas un identificateur DB2 SQL valide.

Action de l'utilisateur : Corrigez la chaîne dans la DAD pour la mettre en conformité avec les règles applicables aux identificateurs DB2 SQL.

DXXQ034E Le noeud RDB supérieur de la DAD est associé à un élément condition non valide *<condition>*

Explication : L'élément condition doit être une clause WHERE valide, composée de conditions de jointure reliées par la conjonction AND.

Action de l'utilisateur : Pour consulter la syntaxe correcte de la condition de jointure dans une DAD, reportez-vous à la documentation relative à l'Extension XML.

DXXQ035E Le noeud RDB supérieur de la DAD est associé à une condition de jointure non valide : *<condition>*

Explication : Les noms de colonnes figurant dans l'élément condition du noeud RDB supérieur doivent être qualifiés par le nom de table approprié lorsque la DAD indique plusieurs tables.

Action de l'utilisateur : Pour consulter la syntaxe correcte de la condition de jointure dans une DAD, reportez-vous à la documentation relative à l'Extension XML.

DXXQ036E Un nom de schéma indiqué dans une balise de condition de la DAD dépasse la longueur admise.

Explication : Erreur détectée lors de l'analyse syntaxique du texte contenu dans une balise de condition de la DAD. Le texte de la condition contient un ID qualifié par un nom de schéma trop long.

Action de l'utilisateur : Corrigez le texte des balises de condition figurant dans la DAD.

DXXQ037E Génération de *<élément>* **impossible avec l'option occurrences multiples.**

Explication : Le noeud d'élément et ses éléments enfants n'ont pas de mappage vers la base de données, mais l'option occurrences multiples est active.

Action de l'utilisateur : Pour corriger la DAD,

désactivez l'option occurrences multiples ou	créez un noeud RDB dans l'un des noeuds enfants.

Traçage de diagnostic

L'Extension XML comprend une fonction de trace qui enregistre l'activité serveur d'Extension XML. Vous devez utiliser la fonction de trace uniquement sur recommandation du centre de support logiciel IBM.

La fonction de trace enregistre les informations dans un fichier de serveur concernant toute une série d'événements, tels que l'entrée ou la sortie d'un composant Extension XML ou le renvoi d'un code d'erreur par un composant Extension XML. Comme elle enregistre les informations liées à de nombreux événements, la fonction de trace ne doit être utilisée qu'en cas de besoin, par exemple si vous étudiez les conditions d'erreur. De plus, vous devez limiter le nombre d'applications actives lorsque vous utilisez la fonction de trace. Cette opération peut faciliter l'isolement de l'origine d'un incident.

La commande **dxxtrc** permet de démarrer ou d'arrêter la fonction de trace. Vous pouvez émettre cette commande à partir d'une ligne de commande sur un serveur AIX, Windows NT ou Solaris. Vous devez disposer des droits SYSADM, SYSCTRL ou SYSMINT pour pouvoir émettre cette commande.

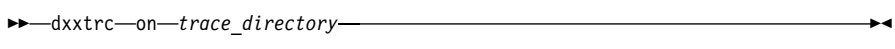
Démarrage de la fonction de trace

Objet

Enregistre l'activité serveur de l'Extension XML. Pour démarrer la fonction de trace, lancez la commande **dxxtrc** avec l'option ON, accompagnée du nom du répertoire contenant le fichier de trace. Lorsque la fonction de trace est activée, le fichier dxx*INSTANCE*.trc est inséré dans le répertoire indiqué. *INSTANCE* est la valeur de DB2INSTANCE. Chaque instance DB2 possède son propre fichier journal.

Syntaxe

Démarrage de la fonction de trace



Paramètres

Tableau 54. Paramètres de la trace

Paramètre	Description
trace_directory	Nom du répertoire où figure le fichier dxx*INSTANCE*.trc. Obligatoire, pas de valeur par défaut.

Exemples

L'exemple suivant illustre le démarrage de la fonction de trace pour une instance db2inst1 sous AIX. Le fichier de trace, dxxdb2inst1.trc, est placé dans le répertoire /home/db2inst1/sqllib/log.

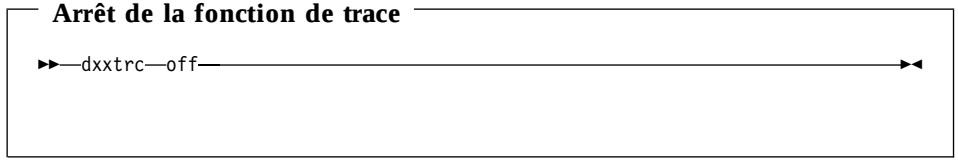
```
dxxtrc on /home/db2inst1/sqllib/log
```

Arrêt de la fonction de trace

Objet

Désactive la fonction de trace. Aucune information de trace n'est répertoriée dans le journal. Comme l'exécution de la trace peut avoir une incidence sur les performances, il est recommandé de désactiver la trace dans un environnement de production.

Syntaxe



Exemples

Cet exemple présente la désactivation de la fonction de trace.

```
dxxtrc off
```

Partie 5. Annexes

Annexe A. DTD de fichier DAD

La présente section décrit la DTD (définition ou déclaration de type de document) applicable au fichier DAD (définition d'accès au document). Le fichier DAD est un document constitué d'une arborescence XML et requiert une DTD. Le nom de fichier de la DTD est `dxxdad.dtd`. La figure 13 à la page 270 présente la DTD correspondant au fichier DAD. Les éléments de ce fichier sont décrits après la figure.

```
<?xml encoding="US-ASCII"?>

  <!ELEMENT DAD (dtdid?, validation, (Xcolumn | Xcollection))>
  <!ELEMENT dtdid (#PCDATA)>
  <!ELEMENT validation (#PCDATA)>
  <!ELEMENT Xcolumn (table*)>
  <!ELEMENT table (column*)>
  <!ATTLIST table name CDATA #REQUIRED
                              key CDATA #IMPLIED
                              orderBy CDATA #IMPLIED>
  <!ELEMENT column EMPTY>
  <!ATTLIST column
                      name CDATA #REQUIRED
                      type CDATA #IMPLIED
                      path CDATA #IMPLIED
                       multi_occurrence CDATA #IMPLIED>
  <!ELEMENT Xcollection (SQL_stmt?, objids?, prolog, doctype, root_node)>
  <!ELEMENT SQL_stmt (#PCDATA)>
  <!ELEMENT objids (column+)>
  <!ELEMENT prolog (#PCDATA)>
  <!ELEMENT doctype (#PCDATA | RDB_node)*>
  <!ELEMENT root_node (element_node)>
  <!ELEMENT element_node (RDB_node*,
                                       attribute_node*,
                                       text_node?,
                                       element_node*,
                                       namespace_node*,
                                       process_instruction_node*,
                                       comment_node*)>
  <!ATTLIST element_node
                     name CDATA #REQUIRED
                      ID CDATA #IMPLIED
                     multi_occurrence CDATA "NO"
                     BASE_URI CDATA #IMPLIED>
  <!ELEMENT attribute_node (column | RDB_node)>
  <!ATTLIST attribute_node
                     name CDATA #REQUIRED>
  <!ELEMENT text_node (column | RDB_node)>
  <!ELEMENT RDB_node (table+, column?, condition?)>
  <!ELEMENT condition (#PCDATA)>
  <!ELEMENT comment_node (#PCDATA)>
  <!ELEMENT namespace_node (EMPTY)>
  <!ATTLIST namespace_node
                      name CDATA #IMPLIED
                     value CDATA #IMPLIED>
  <!ELEMENT process_instruction_node (#PCDATA)>
```

Figure 13. DTD de fichier DAD

Le fichier DAD est constitué de quatre éléments principaux :

- DTDID
- validation
- Xcolumn

- Xcollection

Xcolumn et Xcollection ont un élément enfant et des attributs qui facilitent le mappage des données XML dans les tables relationnelles de DB2. La liste suivante décrit les éléments principaux et leurs éléments enfants ainsi que leurs attributs. Les exemples de syntaxe sont extraits de la figure 13 à la page 270.

élément DTDID

Indique l'ID de la DTD stockée dans la table DTD_REF. `DTDID` pointe vers la DTD qui valide les documents XML ou guide le mappage entre les tables de collection XML et les documents XML. `DTDID` doit être spécifié pour les collections XML. Pour les colonnes XML, la spécification de DTDID est facultative et n'est nécessaire que si vous voulez créer des tables annexes pour l'indexation d'éléments ou d'attributs ou pour valider des documents XML. `DTDID` doit être identique à SYSTEM ID spécifié dans le `doctype` des documents XML.

Syntaxe : `<!ELEMENT dtdid (#PCDATA)>`

élément validation

Indique si un document XML doit être validé avec la DTD du fichier DAD. Si YES est spécifié, `DTDID` doit l'être également.

Syntaxe : `<!ELEMENT validation(#PCDATA)>`

élément Xcolumn

Définit le mode d'indexation d'une colonne XML. Il est constitué de zéro ou plusieurs tables.

Syntaxe : `<!ELEMENT Xcolumn (table*)>Xcolumn` dispose d'un élément enfant : `table`.

élément table

Définit une ou plusieurs tables relationnelles créées pour l'indexation d'éléments ou d'attributs de documents stockés dans une colonne XML.

Syntaxe :

```
<!ELEMENT table (column+)>
  <!ATTLIST table name CDATA #REQUIRED
key CDATA #IMPLIED
orderBy CDATA #IMPLIED>
```

L'élément `table` dispose d'un attribut :

attribut name

Spécifie le nom de la table annexe

L'élément table possède un élément enfant :

attribut key
Clé primaire unique de la table.

attribut orderBy
Noms des colonnes qui déterminent l'ordre séquentiel d'un texte à plusieurs éléments récurrents ou valeurs d'attributs lors de la génération de documents XML.

élément column
Spécifie la colonne de la table qui contient la valeur d'un chemin d'emplacement du type spécifié.

Syntaxe :

```
<!ATTLIST column
                    name CDATA #REQUIRED
                     type CDATA #IMPLIED
                     path CDATA #IMPLIED
                      multi_occurrence CDATA #IMPLIED>
```

L'élément column comprend les attributs suivants :

attribut name
Spécifie le nom de la colonne. Il s'agit de l'alias du chemin d'emplacement qui identifie un élément ou un attribut.

attribut type
Définit le type de données de la colonne. Il peut s'agir de n'importe quel type de données SQL.

attribut path
Affiche le chemin d'emplacement d'un élément ou d'un attribut XML et doit être le chemin d'emplacement simple (voir tableau 3.1.a).

attribut multi_occurrence
Spécifie si cet élément ou attribut peut se reproduire plusieurs fois dans un document XML. Les valeurs admises sont YES ou NO.

Xcollection
Définit le mappage entre les documents XML et une collection XML de tables relationnelles.

Syntaxe : `<!ELEMENT Xcollection(SQL_stmt*, prolog, doctype, root_node)>Xcollection` détient les éléments enfants suivants :

SQL_stmt
Spécifie l'instruction SQL utilisée par l'Extension XML pour définir la collection. Explicitement, l'instruction sélectionne les données XML dans les tables de collection XML et utilise ces

données pour générer les documents XML dans la collection. La valeur de cet élément doit être une instruction SQL valide. Cet élément n'est utilisé que pour la composition, et un seul élément SQL_stmt est autorisé. Pour la décomposition, plusieurs valeurs de SQL_stmt peuvent être spécifiées pour effectuer la création et l'insertion de la table.

Syntaxe : `<!ELEMENT SQL_stmt #PCDATA >`

prolog Texte du prologue XML. Le même prologue est fourni pour tous les documents dans toute la collection. La valeur de prolog est fixe.

Syntaxe : `<!ELEMENT prolog #PCDATA>`

doctype
Définit le texte correspondant à la définition du type de document XML.

Syntaxe : `<!ELEMENT doctype #PCDATA | RDB_node>doctype` peut être spécifié de l'une des manières suivantes :

- Définition d'une valeur explicite. Cette valeur est fournie pour tous les documents dans toute la collection.
- Lors d'une décomposition, spécifiez l'élément enfant, RDB_node, qui peut être mappé et stocké en tant que données de colonne d'une table.

doctype possède un élément enfant :

RDB_node
Mise en oeuvre en cours.

root_node
Définit le noeud racine virtuel. root_node doit avoir un élément enfant, element_node, qui ne peut être utilisé qu'une seule fois. element_node dans la rubrique root_node est en fait l'élément root_node du document XML.

Syntaxe : `<!ELEMENT root_node(element_node)>`

element_node
Représente un élément XML. Il doit être défini dans la DTD indiquée pour la collection. Dans le cas du mappage du noeud RDB, le noeud d'élément (element_node) racine doit être doté d'une valeur RDB_node pour indiquer toutes les tables contenant des données XML pour lui-même et pour tous ses noeuds enfants. Il peut avoir zéro ou plusieurs noeuds d'attributs (attribute_node) et noeuds d'éléments (element_node) enfants, ainsi que zéro ou un noeud de texte

(text_node). Pour les éléments différents de l'élément racine, aucun RDB_node n'est nécessaire.

Syntaxe :

Un element_node est défini par les éléments enfants suivants :

RDB_node
(Facultatif) Spécifie les conditions, la colonne et les tables pour les données XML. Le paramètre RDB_node d'un élément ne doit être défini que pour le mappage du RDB_node. Dans ce cas, une ou plusieurs tables doivent être spécifiées. La colonne n'est pas nécessaire car le contenu de l'élément est spécifié par text_node. La condition est facultative en fonction de la DTD et de la condition de requête.

noeuds enfants
(Facultatif) Un element_node peut également avoir les noeuds enfants suivants :

element_node
Représente les éléments enfants de l'élément XML en cours.

attribute_node
Représente les attributs de l'élément XML en cours.

text_node
Représente le texte CDATA de l'élément XML en cours.

attribute_node
Représente un attribut XML. Il s'agit du noeud définissant le mappage entre un attribut XML et les données de colonne dans une table relationnelle.

Syntaxe :

attribute_node doit comprendre les définitions d'un attribut name et un élément enfant, soit column ou RDB_node. attribute_node comprend l'attribut suivant :

name Nom de l'attribut.

attribute_node comprend les éléments enfants suivants :

Column
Utilisé pour le mappage SQL. column doit être spécifié dans la clause SELECT de SQL_stmt.

RDB_node

Utilisé pour le mappage RDB_node. Le noeud définit le mappage entre cet attribut et les données de colonne dans la table relationnelle. La table et la colonne doivent être spécifiées. La condition est facultative.

text_node

Représente le texte d'un élément XML. Il s'agit du noeud définissant le mappage entre un contenu d'élément XML et les données de colonne dans une table relationnelle.

Syntaxe : Il doit être défini par un élément enfant `column` ou `RDB_node` :

Column

Nécessaire pour le mappage SQL. Dans ce cas, column doit être spécifié dans la clause SELECT de `SQL_stmt`.

RDB_node

Nécessaire pour le mappage RDB_node. Le noeud définit le mappage entre ce texte et les données de colonne de la table relationnelle. `table` et `column` doivent être spécifiés. La condition est facultative.

Annexe B. Exemples

Cette annexe présente les exemples d'objets utilisés dans ce manuel.

DTD XML

La DTD suivante est utilisée pour le document `getstart.xml` cité en référence dans ce manuel (figure 15 à la page 278).

```
<!xml encoding="US-ASCII"?>

<!ELEMENT Order (Customer, Part+)>
<!ATTLIST Order key CDATA #REQUIRED>
<!ELEMENT Customer (Name, Email)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
<!ELEMENT Part (key,Quantity,ExtendedPrice,Tax, Shipment+)>
<!ELEMENT key (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ELEMENT ExtendedPrice (#PCDATA)>
<!ELEMENT Tax (#PCDATA)>
<!ATTLIST Part color CDATA #REQUIRED>
<!ELEMENT Shipment (ShipDate, ShipMode)>
<!ELEMENT ShipDate (#PCDATA)>
<!ELEMENT ShipMode (#PCDATA)>
```

Figure 14. Exemple de DTD XML : getstart.dtd

Document XML : getstart.xml

`getstart.xml`, est l'exemple de document XML utilisé dans ce manuel. Il contient des balises XML permettant de générer un bon de commande.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd">
    <Order key="1">
        <Customer>
    <Name>American Motors</Name>
          <Email>parts@am.com</Email>
        </Customer>
        <Part color="black">
          <key>68</key>
          <Quantity>36</Quantity>
          <ExtendedPrice>34850.16</ExtendedPrice>
          <Tax>6.000000e-02</Tax>
          <Shipment>
              <ShipDate>1998-08-19</ShipDate>
              <ShipMode>BOAT </ShipMode>
          </Shipment>
          <Shipment>
              <ShipDate>1998-08-19</ShipDate>
              <ShipMode>AIR </ShipMode>
          </Shipment>
  </Part>
  <Part color="red   ">
    <key>128</key>
    <Quantity>28</Quantity>
    <ExtendedPrice>38000.00</ExtendedPrice>
    <Tax>7.000000e-02</Tax>
          <Shipment>
      <ShipDate>1998-12-30</ShipDate>
      <ShipMode>TRUCK </ShipMode>
          </Shipment>
  </Part>
</Order>
```

Figure 15. Exemple de document XML : `getstart.xml`

Fichiers DAD

Les sections ci-après contiennent des fichiers DAD (définition d'accès au document) qui mappent des données XML vers des tables relationnelles DB2, via les colonnes ou les collections XML.

- «Fichier DAD : colonne XML» à la page 279
- «Fichier DAD : collection XML (mappage SQL)» à la page 279 : Cette section présente un fichier DAD pour collection XML utilisant le mappage SQL.
- «Fichier DAD : collection XML (mappage du noeud RDB)» à la page 281 : Cette section présente un fichier DAD pour collection XML utilisant le mappage du noeud RDB.

Fichier DAD : colonne XML

Le fichier DAD contient le mappage nécessaire pour une colonne XML et définit la table, les tables annexes et les colonnes qui doivent contenir les données XML.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "c:\dxx\dad.dtd">
<DAD>
   <dtdid>c:\dxx\samples\dtd\getstart.dtd</dtdid>
   <validation>YES</validation>

   <Xcolumn>
     <table name="order_side_tab">
         <column name="order_key"
            type="integer"
            path="/Order/@key"
            multi_occurrence="NO"/>
         <column name="customer"
            type="varchar(50)"
            path="/Order/Customer/Name"
            multi_occurrence="NO"/>
     </table>
     <table name="part_side_tab">
         <column name="price"
            type="decimal(10,2)"
            path="/Order/Part/ExtendedPrice"
            multi_occurrence="YES"/>
     </table>
     <table name="ship_side_tab">
         <column name="date"
            type="DATE"
            path="/Order/Part/Shipment/ShipDate"
            multi_occurrence="YES"/>
     </table>

   </Xcolumn>

</DAD>
```

Figure 16. Fichier exemple DAD pour une colonne XML

Fichier DAD : collection XML (mappage SQL)

Le fichier DAD comporte une instruction SQL indiquant les tables, les colonnes et les conditions DB2 qui doivent contenir des données XML.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
<validation>NO</validation>
<Xcollection>
<SQL_stmt>SELECT o.order_key, customer_name, customer_email, p.part_key, color, quantity,
   price, tax, ship_id, date, mode from order_tab o, part_tab p,
table (select substr(char(timestamp(generate_unique())),16)
  as ship_id, date, mode, part_key from ship_tab) s
      WHERE o.order_key = 1 and
                p.price > 20000 and
            p.order_key = o.order_key and
            s.part_key = p.part_key
         ORDER BY order_key, part_key, ship_id</SQL_stmt>
      <prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
```

Figure 17. Fichier exemple DAD pour collection XML utilisant le mappage SQL (Numéro 1 de 2)

```
      <root_node>
         <element_node name="Order">
  <attribute_node name="key">
                 <column name="order_key"/>
             </attribute_node>
          <element_node name="Customer">
    <element_node name="Name">
      <text_node><column name="customer_name"/></text_node>
  </element_node>
    <element_node name="Email">
      <text_node><column name="customer_email"/></text_node>
  </element_node>
             </element_node>
          <element_node name="Part">
    <attribute_node name="color">
      <column name="color"/>
             </attribute_node>
    <element_node name="key">
      <text_node><column name="part_key"/></text_node>
  </element_node>
<element_node name="Quantity">
      <text_node><column name="quantity"/></text_node>
  </element_node>
          <element_node name="ExtendedPrice">
      <text_node><column name="price"/></text_node>
  </element_node>
          <element_node name="Tax">
      <text_node><column name="tax"/></text_node>
  </element_node>
    <element_node name="Shipment" multi_occurence="YES">
             <element_node name="ShipDate">
        <text_node><column name="date"/></text_node>
    </element_node>
             <element_node name="ShipMode">
        <text_node><column name="mode"/></text_node>
  </element_node>
  </element_node>
             </element_node>
             </element_node>
      </root_node>
  </Xcollection>
</DAD>
```

Figure 17. Fichier exemple DAD pour collection XML utilisant le mappage SQL (Numéro 2 de 2)

Fichier DAD : collection XML (mappage du noeud RDB)

Ce fichier DAD définit à l'aide des éléments <RDB_node> les tables, les colonnes et les conditions DB2 qui doivent contenir des données XML.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
   <dtdid>c:\dxx\samples\dtd\getstart.dtd</dtdid>
   <validation>YES</validation>
<Xcollection>
      <prolog>?xml version="1.0"?</prolog>
      <doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
      <root_node>
         <element_node name="Order">
                     <RDB_node>
                     <table name="order_tab"/>
                     <table name="part_tab"/>
                        <table name="ship_tab"/>
                        <condition>
                 order_tab.order_key = part_tab.order_key AND
                     part_tab.part_key = ship_tab.part_key
                        </condition>
                     </RDB_node>
  <attribute_node name="key">
                     <RDB_node>
                     <table name="order_tab"/>
                  <column name="order_key"/>
                     </RDB_node>
               </attribute_node>
           <element_node name="Customer">
                 <text_node>
                    <RDB_node>
                    <table name="order_tab"/>
                    <column name="customer"/>
                    </RDB_node>
                 </text_node>
              </element_node>
           <element_node name="Part">
                     <RDB_node>
                     <table name="part_tab"/>
                        <table name="ship_tab"/>
                        <condition>
                     part_tab.part_key = ship_tab.part_key
                        </condition>
                     </RDB_node>
  <attribute_node name="key">
                     <RDB_node>
                     <table name="part_tab"/>
                     <column name="part_key"/>
                     </RDB_node>
               </attribute_node>
```

Figure 18. Fichier exemple DAD pour collection XML utilisant le mappage du noeud RDB (Numéro 1 de 3)

```
<element_node name="Quantity">
      <text_node>
         <RDB_node>
         <table name="part_tab"/>
         <column name="quantity"/>
         </RDB_node>
      </text_node>
   </element_node>
<element_node name="ExtendedPrice">
      <text_node>
         <RDB_node>
         <table name="part_tab"/>
            <column name="price"/>
            <condition>
               price > 2500.00
            </condition>
         </RDB_node>
      </text_node>
   </element_node>
<element_node name="Tax">
      <text_node>
         <RDB_node>
         <table name="part_tab"/>
         <column name="tax"/>
         </RDB_node>
      </text_node>
   </element_node>
```

Figure 18. Fichier exemple DAD pour collection XML utilisant le mappage du noeud RDB (Numéro 2 de 3)

```
            <element_node name="shipment">
                    <RDB_node>
                       <table name="ship_tab"/>
                       <condition>
                    part_key = part_tab.part_key
                       </condition>
                    </RDB_node>
               <element_node name="ShipDate">
                  <text_node>
                     <RDB_node>
                        <table name="ship_tab"/>
                        <column name="date"/>
                        <condition>
                           date > "1966-01-01"
                        </condition>
                     </RDB_node>
                  </text_node>
               </element_node>
               <element_node name="ShipMode">
                  <text_node>
                     <RDB_node>
                        <table name="ship_tab"/>
                        <column name="mode"/>
                     </RDB_node>
                  </text_node>
               </element_node>
               <element_node name="Comment">
                  <text_node>
                     <RDB_node>
                        <table name="ship_tab"/>
                        <column name="comment"/>
                     </RDB_node>
                  </text_node>
               </element_node>
             </element_node> <! -- end of element Shipment>
           </element_node> <! -- end of element Part --->
         </element_node> <! -- end of element Order --->
       </root_node>
  </Xcollection>
</DAD>
```

Figure 18. Fichier exemple DAD pour collection XML utilisant le mappage du noeud RDB (Numéro 3 de 3)

Annexe C. Remarques relatives aux pages de codes

Il est important de vérifier que les documents XML, ainsi que les autres fichiers associés, soient correctement codés afin que chaque client ou serveur puisse accéder aux fichiers. Il est donc important de comprendre les présuppositions de l'Extension XML lors du traitement d'un fichier, ainsi que la façon dont le produit gère ensuite les conversions de pages de codes. Les principales opérations à faire sont les suivantes :

- S'assurer que la page de codes réelle du client qui extrait un document XML à partir de DB2, correspond à la déclaration de codage (ENCODING) contenue dans le document XML.
- S'assurer que lorsque le document est traité par un analyseur XML, la déclaration ENCODING du document XML est également cohérente avec le codage.

Les sections qui suivent décrivent les inconvénients liés aux considérations préalablement évoquées et la façon de limiter ces inconvénients, ainsi que la façon dont l'Extension XML et DB2 prennent en charge les pages de codes lors du transfert de documents du client vers le serveur ou vers la base de données.

Terminologie

La présente section utilise les termes qui suivent :

codage du document
Page de codes en cours d'un document XML.

déclaration de codage du document (ou déclaration ENCODING)
Nom de la page de codes spécifiée dans la déclaration XML. Par exemple :

```
<?xml version="1.0" encoding="ibm037"?>
```

document cohérent
Document dont le codage correspond à la page de codes indiquée dans la déclaration ENCODING.

document incohérent
Document dont le codage ne correspond pas à la page de codes indiquée dans la déclaration ENCODING.

variable de registre (d'environnement) DB2CODEPAGE
Définit la page de codes des données présentées à DB2 à partir d'une application client de base de données. DB2 extrait la page de codes du client à partir de l'environnement local du système d'exploitation du

client, sauf si cette variable est définie. Pour DB2, cette valeur, lorsqu'elle est définie, remplace la valeur définie dans l'environnement local du système d'exploitation du client.

page de codes du client
Page de codes de l'application. Si la variable DB2CODEPAGE est définie, la page de codes du client a pour valeur celle de DB2CODEPAGE. Dans le cas contraire, la page de codes du client est celle définie dans l'environnement local du système d'exploitation du client.

page de codes du serveur ou page de codes définie dans l'environnement local du système d'exploitation du serveur
Environnement local du système d'exploitation sur lequel la base de données DB2 est installée.

page de codes de la base de données
Codage des données stockées, déterminé au moment de la création de la base de données. Si cette valeur n'est pas définie explicitement dans la clause USING CODESET, elle prend par défaut celle définie dans l'environnement local du système d'exploitation du serveur.

Présuppositions de DB2 et de l'Extension XML quant à la page de codes

Lorsque DB2 reçoit ou envoie un document XML, il ne vérifie pas la déclaration ENCODING. Il examine plutôt la page de codes du client pour s'assurer qu'elle correspond à celle de la base de données. Si les deux pages de codes sont différentes, DB2 convertit les données du document XML pour les adapter à la page de codes :

- de la base de données, lors de l'importation du document ou d'un fragment de document dans une table de base de données,
- de la base de données, lors de la décomposition d'un document en une ou plusieurs tables de base de données,
- du client, lors de l'exportation du document à partir de la base de données et de la présentation du document au client,
- du serveur, lors du traitement d'un fichier par une fonction UDF qui renvoie des données dans un fichier sur le système de fichiers du serveur.

Lorsqu'un document XML est importé dans la base de données, il l'est généralement sous forme de document XML à stocker dans une colonne XML ou à décomposer dans une collection XML dans laquelle le contenu des éléments et des attributs sera sauvegardé sous forme de données DB2. Lorsqu'un document est importé, DB2 remplace le codage du document par celui de la base de données. DB2 suppose que le document est dans la page de codes spécifiée dans la colonne «Page de codes source» du tableau ci-après. Le tableau 55 à la page 287 résume les conversions de pages de codes

effectuées par DB2 lors de l'importation d'un document XML.

Tableau 55. Utilisation de fonctions UDF et de procédures mémorisées lors de l'importation d'un fichier XML dans la base de données

Méthode de traitement	Page de codes source pour la conversion	Page de codes cible pour la conversion	Commentaires
Insertion d'un fichier DTD dans la table DTD_REF	Page de codes du client	Page de codes de la base de données	
Colonne active, procédure mémorisée de collection active ou commandes d'administration, qui importe(nt) le fichier DAD	Page de codes du client	Page de codes de la base de données	
Fonctions UDF : • XMLVarcharFromFile() • XMLCLOBFromFile() • Content(): récupération à partir de XMLFILE vers un objet CLOB	Page de codes du serveur	Page de codes de la base de données	
Procédures mémorisées de décomposition	Page de codes du client	Page de codes de la base de données	Le document à décomposer est supposé être dans la page de codes du client. Les données issues de la décomposition sont stockées dans des tables définies dans la page de codes de la base de données.

Lorsqu'un document XML est exporté à partir d'un base de données, cette exportation fait généralement suite à la demande de présentation de ce document par un client, à la demande de visualisation de son contenu ou à une demande effectuée durant la composition d'un document à partir de données DB2. Lorsqu'un document est importé, DB2 remplace le codage du document par celui du client ou du serveur, selon que l'endroit à partir duquel la demande a été effectuée et l'endroit au niveau duquel les données doivent être présentées. Le tableau 56 à la page 288 résume les conversions de pages de codes effectuées par DB2 lors de l'exportation d'un document XML.

Tableau 56. Utilisation de fonctions UDF et de procédures mémorisées lors de l'exportation d'un fichier XML à partir de la base de données

Méthode de traitement	Conversion	Commentaires
Fonction UDF • XMLFileFromVarchar() • XMLFileFromCLOB()	De la page de codes de la base de données vers celle du client lors de la présentation des données au client	
Fonction UDF • Content() : récupération à partir de XMLVARCHAR vers un fichier de serveur externe	De la page de codes de la base de données vers celle du serveur	
Procédure mémorisée de composition : tables résultantes stockées dans une table résultat, qui peut être exportée ou faire l'objet d'une requête.	De la page de codes de la base de données vers celle du client lors de la présentation du jeu de résultats au client	Lors de la composition des documents, l'Extension XML copie la déclaration de codage, spécifiée par la marque dans le fichier DAD, dans le document nouvellement créé. Elle doit correspondre à la page de codes du client lors de la présentation.

Remarques relatives à la déclaration de codage

La déclaration ENCODING, qui permet de déclarer le codage du document XML, apparaît dans l'instruction de déclaration XML. Lorsque l'on utilise l'Extension XML, il est important de vérifier si le codage du document correspond à la page de codes du client ou du serveur, selon l'endroit où le fichier est situé.

Déclarations ENCODING admises

Vous pouvez utiliser toute déclaration ENCODING dans les documents XML, à condition de respecter certaines conditions. Cette section présente les conditions à respecter, ainsi que les déclarations ENCODING admises.

Les codages recommandés pour les données XML sont les codages UTF-8 et UTF-16, en fonction de la spécification XML. Si vous utilisez ces codages, votre application sera utilisable conjointement par plusieurs entreprises. Si vous utilisez les codages figurant dans le tableau 57 à la page 289, votre application pourra être transférable d'un système d'exploitation IBM à un autre. Si vous utilisez d'autres codes, vos données auront moins de chances d'être transférables.

Pour tous les systèmes d'exploitation, les déclarations ENCODING prises en charge sont indiquées ci-après. La liste qui suit explique la signification de chaque colonne :

- **ENCODING** désigne la chaîne ENCODING à utiliser dans la déclaration XML.
- **SE** désigne le système d'exploitation sur lequel DB2 prend en charge la page de codes indiquée.
- **Page de codes** désigne la page de codes IBM associée à la chaîne ENCODING indiquée.

Tableau 57. Déclarations ENCODING prises en charge par l'Extension XML

Catégorie	ENCODING	SE	Page de codes
Unicode	UTF-8	AIX, SUN, Linux	1208
	UTF-16	AIX, SUN, Linux	1200
ASCII	iso-8859-1	AIX, Linux, Sun	819
	ibm-1252	Windows NT	1252
	iso-8859-2	AIX, Linux, Sun	912
	iso-8859-5	AIX, Linux	915
	iso-8859-6	AIX	1089
	iso-8859-7	AIX, Linux	813
	iso-8859-8	AIX, Linux	916
	iso-8859-9	AIX, Linux	920
MBCS	gb2312	Windows NT	1386
	ibm-932, shift_jis78	AIX	932
	Shift_JIS	AIX 4.3 uniquement, Windows NT	943
	IBM-eucCN	AIX, Sun	1383
	IBM-eucJP, EUC-JP	AIX, Linux, Sun	954, 33722
	euc-tw, IBM-eucTW	AIX, Sun	964
	euc-kr, IBM-eucKR	AIX	970
	big5	AIX, Sun, Windows NT	950

La chaîne ENCODING doit être compatible avec la page de codes du site cible du document. Si un document est renvoyé d'un serveur à un client, sa chaîne ENCODING doit être compatible avec la page de codes du client. Pour connaître les conséquences de codages incompatibles, reportez-vous à la section «Codages et déclarations ENCODING compatibles» à la page 290. Pour

obtenir la liste des pages de codes prises en charge par l'analyseur XML utilisé par l'Extension XML, consultez le Web à l'adresse suivante :

`http://www.ibm.com/software/data/db2/extenders/xmlext/moreinfo/encoding.html`

Codages et déclarations ENCODING compatibles

Lorsqu'un document XML est traité ou échangé avec un autre système, il est important que la déclaration ENCODING corresponde au codage du document. Il est important de vérifier que le codage d'un document est cohérent avec celui du client car les outils XML, tels que les analyseurs syntaxiques, génèrent une erreur lorsque l'entité contient une déclaration de codage autre que celle indiquée dans la déclaration.

La figure 19 présente des clients ayant des pages de codes cohérentes avec le codage et les déclarations ENCODING du document.

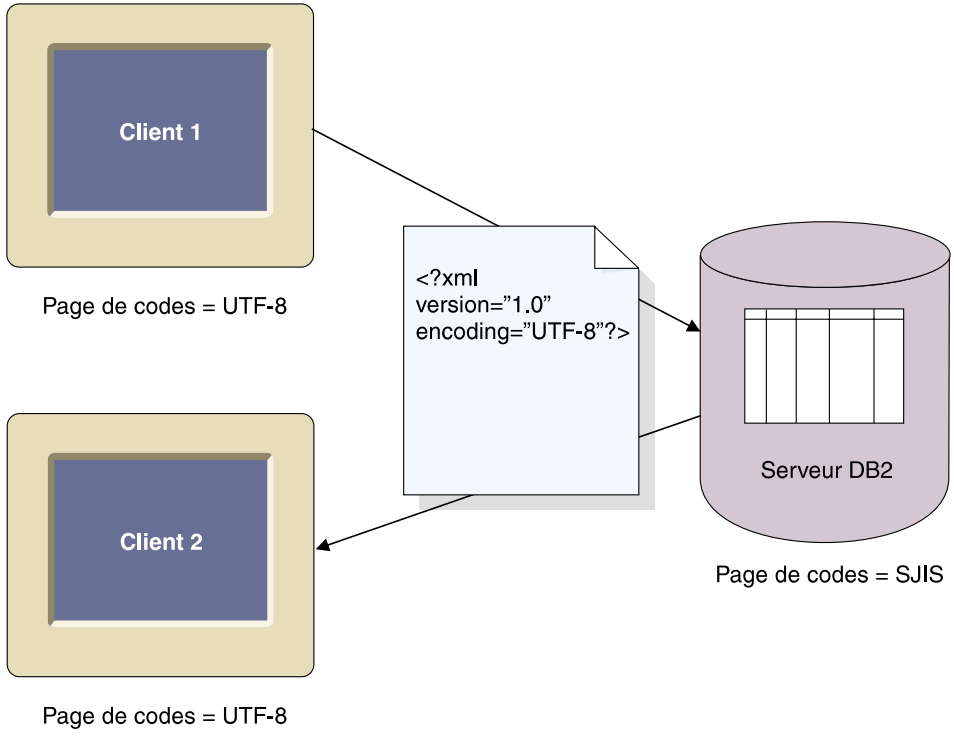


Figure 19. Les clients disposent de pages de codes concordantes

Les conséquences induites par les différences de pages de codes peuvent être les suivantes :

- Une conversion avec perte de données peut intervenir, en particulier si la page de codes source est de type Unicode, alors que la page de codes cible ne l'est pas. Unicode contient la totalité du jeu de conversions de caractères.

Si un fichier est converti à partir de UTF-8 dans une page de codes qui ne prend pas en charge la totalité des caractères utilisés dans le document, des données peuvent être perdues durant la conversion.

- Le codage déclaré et le codage réel du document XML peuvent ne plus être cohérents lorsque le document a été extrait par un client doté d'une page de codes non conforme au codage déclaré.

La figure 20 représente un environnement dans lequel les pages de codes des clients sont incohérentes.

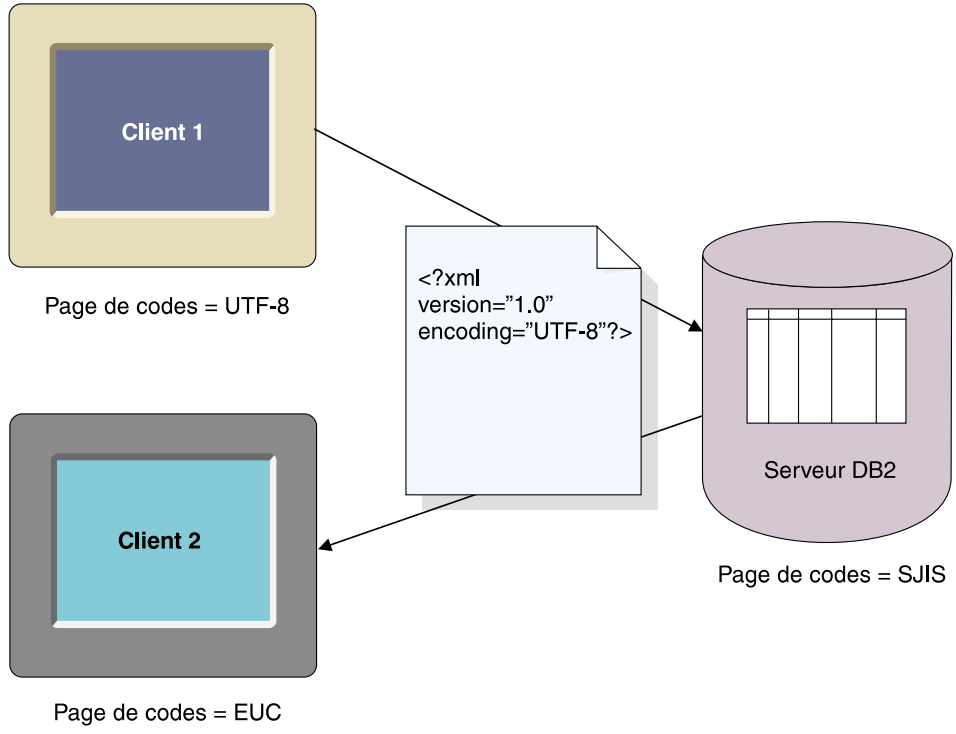


Figure 20. Clients dont les pages de codes ne correspondent pas

Le Client2 reçoit le document dans la page de codes EUC mais ce document comporte une déclaration ENCODING UTF-8.

Déclaration ENCODING

La valeur par défaut de la déclaration ENCODING est UTF-8 et l'absence de déclaration de codage signifie que le document est en UTF-8.

Pour déclarer une valeur de codage, procédez comme suit :

Dans la déclaration de document XML, définissez la déclaration ENCODING par le nom de la page de codes du client. Par exemple :

```
<?xml version="1.0" encoding="UTF-8" ?>
```

Scénarios de conversion

L'Extension XML traite les documents XML lors des opérations suivantes :

- Stockage et retrait des données de la colonne XML, à l'aide de la méthode d'accès et de stockage adaptée.
- Composition et décomposition des documents XML

Les documents subissent une conversion de page de codes lorsqu'ils sont transférés d'un client ou d'un serveur vers une base de données. Les conversions de pages de codes entre clients, serveurs et bases de données peuvent donner lieu à des incohérences ou des dommages au niveau des documents XML. Lorsque vous choisissez la déclaration ENCODING d'un document et planifiez les clients et serveurs qui pourront importer des documents de la base de données ou en exporter vers celle-ci, prenez en considération les conversions décrites dans les tables précédemment mentionnées et les scénarios décrits ci-après.

Vous trouverez ci-après les scénarios de conversion les plus courants :

Scénario 1 : Ce scénario concerne une configuration avec des codages cohérents, sans conversion DB2 et un document importé à partir du serveur. La déclaration ENCODING du document a pour valeur UTF-8, celle du serveur, UTF-8, et celle de la base de données, UTF-8.

1. Le document est importé dans DB2 via la fonction XMLClobfromFile.
2. Il est ensuite extrait vers le serveur.
3. Il n'est pas nécessaire que DB2 convertisse le document car la page de codes du serveur et celle de la base de données sont identiques. Le codage et la déclaration sont cohérents.

Scénario 2 : Ce scénario concerne une configuration avec des codages cohérents, une conversion DB2 et un document importé à partir du serveur et exporté vers le client. Le codage et la déclaration ENCODING ont pour valeur SJIS, le client a pour page de codes SJIS, et le serveur et la base de données ont pour page de codes UTF-8.

1. Le document est importé dans DB2 via la fonction UDF XMLClobfromfile à partir du serveur.
2. DB2 le convertit à partir de SJIS et le stocke en UTF-8. La déclaration ENCODING et le codage sont incohérents dans la base de données.

3. Un client utilisant SJIS demande la présentation du document sur le navigateur Web.
4. DB2 convertit le document en SJIS (page de codes du client). Le codage et la déclaration ENCODING sont à présent cohérents sur le client.

Scénario 3 : Ce scénario concerne une configuration avec des codages incohérents, une conversion DB2 et un document importé à partir du serveur et exporté vers le client. La déclaration ENCODING du document entrant a pour valeur SJIS. Le serveur a pour page de codes SJIS, et le client et la base de données, UTF-8.

1. Le document est importé dans la base de données via une fonction UDF de stockage.
2. DB2 le convertit de SJIS en UTF-8. Le codage et la déclaration sont incohérents.
3. Un client ayant la page de codes UTF-8 demande la présentation du document sur un navigateur Web.
4. DB2 n'effectue pas de conversion car les pages de codes du client et de la base de données sont identiques.
5. Le codage et la déclaration ENCODING du document sont incohérentes car la déclaration a pour valeur SJIS et le codage, UTF-8. Le document ne peut pas être traité par un analyseur XML ou par un autre outil de traitement XML.

Scénario 4 : Ce scénario concerne une configuration avec une perte de données, une conversion DB2 et un document importé à partir d'un serveur ayant une page de codes UTF-8 : la déclaration ENCODING du document a pour valeur UTF-8, le serveur a pour page de codes UTF-8 et la base de données, SJIS.

1. Le document est importé dans DB2 via la fonction XMLClobfromFile.
2. DB2 le convertit en SJIS. Le document, qui est stocké dans la base de données, peut être altéré lors de son importation car les caractères représentés en page de codes UTF-8 peuvent ne pas l'être en SJIS.

Scénario 5 :

Ce scénario concerne une configuration faisant intervenir une limitation Windows NT. Sous Windows NT, l'environnement local du système d'exploitation ne peut pas être défini en page de codes UTF-8. Cependant, DB2 permet de définir le client en page de codes UTF-8 via la commande `db2set DB2CODEPAGE=1208`. Dans ce scénario, le client et le serveur sont sur la même machine. Le client est en UTF-8, mais le serveur ne peut pas être défini en UTF-8 car sa page de codes est 1252. Le document est en page de codes 1252 et la déclaration ENCODING a pour valeur ibm-1252. La page de codes de la base de données a pour valeur UTF-8.

1. Le document est importé à partir du serveur par une fonction UDF de stockage et converti de la page de codes 1252 à la page 1208.
2. Le document est exporté à partir de DB2 via la fonction UDF Content() utilisée par le client.
3. DB2 convertit le document de la page de codes UTF-8 dans la page de codes 1252, même si le client attend la page de de codes 1208, étant donné qu'il est sur le même système que le serveur, lui-même en page de codes 1208.

Mesures visant à éviter les documents incohérents XML

Les sections précédentes ont expliqué les problèmes d'incohérence de codage pouvant intervenir dans les documents XML, c'est-à-dire une déclaration ENCODING différente du codage du document. Les incohérences de codage peuvent provoquer la perte de données ou rendre des documents XML inutilisables.

Les conseils ci-après permettent de vérifier la cohérence entre le codage du document XML et la page de codes des clients, avant le transfert du document vers un processeur XML tel qu'un analyseur syntaxique.

- Lorsque vous exportez un document à partir de la base de données via les fonctions UDF de l'Extension XML, essayez l'une des techniques suivantes : (on suppose que l'Extension XML a exporté le fichier dans la page de codes du serveur vers le système de fichiers sur le serveur).
  - Convertissez le document dans la page de codes du codage déclaré.
  - Ignorez le codage déclaré si l'outil le permet.
  - Remplacez manuellement la déclaration ENCODING du document exporté par la codage réel du document (c'est-à-dire la page de codes du serveur).
- Lorsque vous exportez un document à partir de la base de données via les procédures mémorisées de l'Extension XML, essayez l'une des techniques suivantes : (on suppose que le client effectue une requête concernant la table résultat dans laquelle le document composé est stocké).
  - Convertissez le document dans la page de codes du codage déclaré.
  - Ignorez le codage déclaré si l'outil le permet.
  - Avant d'effectuer la requête concernant la table résultat, définissez la variable d'environnement du client DB2CODEPAGE de façon à ce que la page de codes du client soit une page de codes compatible avec la déclaration ENCODING du document XML.
  - Remplacez manuellement la déclaration ENCODING du document exporté par le codage réel du document (c'est-à-dire la page de codes du client).

- **Limitation lors de l'utilisation d'Unicode et de Windows NT :** Sous Windows NT, l'environnement local du système d'exploitation ne peut pas être défini en page de codes UTF-8. Suivez les instructions suivantes lors de l'importation ou de l'exportation de documents :
  - Lors de l'importation de fichiers et de fichiers DTD codés en UTF-8, définissez la page de codes du client par UTF-8, en utilisant l'instruction suivante :

    ```
    db2set DB2CODEPAGE=1208
    ```

    Utilisez cette technique dans les cas suivants :
    - Insertion d'un fichier DTD dans la table db2xml.DTD_REF
    - Activation d'une colonne ou d'une collection
    - Décomposition de procédures mémorisées
  - Lorsque vous utilisez les fonctions UDF Content() ou XMLxxxfromFile pour importer des documents XML, les documents doivent être codés dans la page de codes de l'environnement local du système d'exploitation du serveur, qui ne peut être la page de codes UTF-8.
  - Lorsque vous exportez un fichier XML à partir de la base de données, définissez la page de codes du client via la commande suivante afin que DB2 code les données résultantes en UTF-8 :

    ```
    db2set DB2CODEPAGE=1208
    ```

    Utilisez cette technique dans les cas suivants :
    - Requête relative à une table résultat à la suite d'une composition
    - Extraction de données à partir d'une colonne XML via les fonctions d'extraction UDF
  - Lorsque vous utilisez les fonctions UDF Content() ou XMLxxxfromFile pour exporter des documents XML vers des fichiers situés sur le système de fichiers du serveur, les documents résultants sont codés dans la page de codes de l'environnement local du système d'exploitation du serveur, qui ne peut être la page de codes UTF-8.

Annexe D. Les limites de l'Extension XML

Le fichier DAD, les procédures mémorisées et les tables de l'Extension XML sont soumis aux limites suivantes :

Tableau 58. Limites de l'Extension XML

Valeur	**Limite**
Table dans une collection XML de décomposition	1024 lignes pour chaque document XML décomposé
Paramètres des procédures mémorisées :	
CLOB de document XML	1 Mo[2]
CLOB de fichier DAD (Définition d'accès à un document)	100 ko[2]
collectionName	30
colName	30
dbName	18
defaultView	128
rootID	128
resultTabName	128
tablespace	128
tbName	128[1]
Colonnes de la table db2xml.DTD_REF	
AUTHOR	128
CREATOR	128
UPDATOR	128
DTDID	128
CLOB	100 ko

Remarques :

1. Si le paramètre *tbName* est qualifié par un nom de schéma, la totalité du nom (y compris le caractère de séparation) ne doit pas dépasser 128 caractères.
2. Cette taille peut être modifiée. Pour savoir comment, reportez-vous à la section «Augmentation des limites du paramètre CLOB» à la page 216.

Les noms peuvent être rallongés lorsqu'ils sont convertis via DB2 d'une page de codes client vers une page de codes base de données. Un nom peut alors

être dans les limites de taille autorisées pour le client, mais dépasser ces limites une fois converti, lorsque la procédure mémorisée le reçoit. Pour plus d'informations, reportez-vous à la section concernant «le développement des applications de support de la langue nationale» du chapitre relatif à la «programmation dans des environnement complexes» dans le manuel*DB2 Application Development Guide*.

Remarques

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM non annoncés dans ce pays. Pour plus de détails, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial IBM. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevets couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous confère aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM Europe Director of Licensing
IBM Europe Middle-East Africa
Tour Descartes - La Défense 5
2 avenue Gambetta
92066 Paris-La Défense CEDEX
France

Les informations sur les licences concernant les produits utilisant un jeu de caractères double octet peuvent être obtenues par écrit à l'adresse suivante :

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japon

Le paragraphe suivant ne s'applique ni au Royaume-Uni, ni dans aucun pays dans lequel il serait contraire aux lois locales. LE PRESENT DOCUMENT EST LIVRE EN L'ETAT. IBM DECLINE TOUTE RESPONSABILITE, EXPRESSE OU IMPLICITE, RELATIVE AUX INFORMATIONS QUI Y SONT CONTENUES, Y COMPRIS EN CE QUI CONCERNE LES GARANTIES DE QUALITE MARCHANDE OU D'ADAPTATION A VOS BESOINS. Certaines juridictions n'autorisent pas l'exclusion des garanties implicites, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

Le présent document peut contenir des inexactitudes ou des coquilles. Il est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut modifier sans préavis les programmes et les logiciels qu'il décrit.

Les détenteurs de licences souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

IBM Corporation
J74/G4
555 Bailey Avenue
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans ce document et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM conformément aux termes du Contrat sur les produits et services IBM, des Conditions internationales d'utilisation des logiciels IBM ou de tout autre accord équivalent.

Les informations concernant les produits non IBM ont été obtenues auprès des fournisseurs de ces produits, par l'intermédiaire d'annonces publiques ou via d'autres sources disponibles. IBM n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances ni leur compatibilité. Elle ne peut recevoir aucune réclamation concernant des produits non IBM. Toute question concernant les performances de produits non IBM doit être adressée aux fournisseurs de ces produits.

Toute instruction relative aux intentions d'IBM pour ses opérations à venir est susceptible d'être modifiée ou annulée sans préavis, et doit être considérée uniquement comme un objectif.

LICENCE DE COPYRIGHT :

Le présent logiciel contient des exemples de programmes d'application en langage source destinés à illustrer les techniques de programmation sur différentes plateformes d'exploitation. Vous avez le droit de copier, de modifier et de distribuer ces exemples de programmes sous quelque forme que ce soit et sans paiement d'aucune redevance à IBM, à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes aux interfaces de programmation des plateformes pour lesquels ils ont été écrits ou aux interfaces de programmation IBM. Ces

exemples de programmes n'ont pas été rigoureusement testés dans toutes les conditions. Par conséquent, IBM ne peut garantir expressément ou implicitement la fiabilité, la maintenabilité ou le fonctionnement de ces programmes.

Marques

Les termes suivants sont des marques d'International Business Machines Corporation dans certains pays :

DB2	Net.Data
DB2 Extenders (DB2 Extensions)	OS/2
DB2 Universal Database (UDB)	OS/390
IBM	OS/400
IMS	VTAM

Java et toutes les marques et logos incluant Java sont des marques de Sun Microsystems, Inc. dans certains pays.

Microsoft, Windows, Windows NT et le logo Windows sont des marques de Microsoft Corporation dans certains pays.

UNIX est une marque déposée dans certains pays, dont seule la société X/Open Limited Company peut concéder la licence.

D'autres sociétés sont propriétaires des autres marques, noms de produits ou logos qui pourraient apparaître dans ce document.

Glossaire

API : Voir *Interface de programmation d'applications.*

attribut : Voir *attribut XML.*

attribut XML : Tout attribut spécifié par ATTLIST sous l'élément XML dans la DTD. L'Extension XML utilise le chemin d'emplacement pour identifier un attribut.

balise XML : Toute balise du langage de marquage XML valide, principalement, élément XML. Les termes ″balise″ et ″élément″ sont utilisés indifféremment.

chemin d'emplacement : Séquence de balises XML identifiant un attribut ou un élément XML. Le chemin d'emplacement identifie la structure du document XML, indiquant le contexte de l'élément ou de l'attribut. Un chemin avec une barre oblique simple (/) indique que le contexte est constitué du document entier. Le chemin d'emplacement est utilisé dans les fonctions UDF d'extraction pour identifier les éléments et les attributs à extraire. Il est également employé dans le fichier DAD pour spécifier le mappage entre un élément ou un attribut XML et une colonne DB2 lors de la définition du mode d'indexation d'une colonne XML. Par ailleurs, l'Extension Texte l'utilise pour la recherche structurelle.

chemin d'emplacement absolu : Chemin d'accès complet d'un objet. Le nom du chemin absolu commence au niveau le plus élevé, ou élément racine (ROOT), identifié par la barre oblique (/) ou la barre oblique inversée (\).

chemin d'emplacement simple : Séquence de noms de types d'élément connectés par une barre oblique simple (/).

clé associée : Clé faisant partie de la définition d'une contrainte référentielle, constituée d'une ou de plusieurs colonnes d'une table dépendante.

clé primaire : Clé unique faisant partie de la définition d'une table. Une clé primaire est la clé parente par défaut d'une définition de contrainte référentielle.

CLOB : Objet CLOB (Character Large OBject).

collection XML : Collection de tables relationnelles présentant les données nécessaires pour composer des documents XML ou les données à décomposer à partir de documents XML.

colonne XML : Colonne d'une table d'application, activée pour les types UDT de l'Extension XML.

composer : Générer des documents XML à partir de données relationnelles figurant dans une collection XML.

condition : Spécification soit des critères de sélection des données XML soit de la méthode de jointure des tables d'une collection XML.

DAD : Voir *Définition d'accès à un document.*

DATALINK : Type de données DB2 permettant des références logiques entre la base de données et un fichier stocké hors de celle-ci.

DBCLOB : Objet DBCLOB (Double-Byte Character Large OBject).

décomposer : Sépare les documents XML en une série de tables relationnelles dans une collection XML.

définition d'accès à un document (DAD) : Utilisée pour définir le schéma d'indexation d'une colonne XML ou le schéma de mappage d'une collection XML. Elle peut être utilisée pour activer une colonne de l'Extension XML d'une collection XML, au format XML.

définition ou déclaration de type de document : Ensemble de déclarations pour les attributs et éléments XML. La DTD définit les éléments utilisés dans le document XML, l'ordre dans lequel ils peuvent être utilisés et les éléments pouvant contenir d'autres éléments. Vous pouvez associer une DTD à un fichier DAD (définition d'accès au document) pour valider des documents XML.

document bien formé : Document XML ne contenant pas de DTD. Toutefois, dans la spécification XML, un document avec une DTD valide doit également être bien formé.

document valide : Document XML disposant d'une DTD associée. Pour être valide, le document XML doit respecter les règles de syntaxe spécifiées dans sa DTD.

données de colonne : Données stockées dans une colonne DB2. Le type de ces données doit être pris en charge par DB2.

DTD : (1) . (2) Voir *définition ou déclaration de type de document*.

échange de données : Partage de données entre applications. XML prend en charge l'échange de données sans avoir besoin de passer par le processus de transformation des données d'un format propriétaire.

Echange de données informatisées (EDI) : Norme appliquée dans le cadre de l'échange de données informatisées pour les applications de type B2B (business-to-business).

EDI : Echange de données informatisées.

élément : Voir *élément XML*.

élément racine : Elément supérieur d'un document XML.

élément XML : Toute balise XML ou ELEMENT comme indiqué dans la DTD XML. L'Extension XML utilise le chemin d'emplacement pour identifier un élément.

ensemble de résultats : Ensemble de lignes renvoyées par une procédure mémorisée.

espace table : Abstraction représentant une collection de conteneurs dans lesquels sont stockés les objets de base de données. Un espace table fournit un niveau d'indirection entre une base de données et les tables stockées dans la base de données. Un espace table :

- dispose d'espace sur les unités de stockage de supports qui lui sont assignés.
- contient des tables créées à l'intérieur. Ces tables remplissent l'espace des conteneurs appartenant à l'espace table. Les données, index, champ étendu et sections LOB d'une table peuvent être stockés dans le même espace table ou peuvent être séparés individuellement entre plusieurs espaces table.

expression de chemin : Voir *chemin d'emplacement*.

Extensible Stylesheet Language (XSL) : Langage utilisé pour les feuilles de style. XSL est formé de deux parties : un langage permettant de transformer des documents XML et un dictionnaire XML pour spécifier la sémantique de mise en forme.

Extensive Stylesheet Language Transformation (XSLT) : Langage utilisé pour transformer des documents XML en d'autres documents XML. XSLT est conçu pour être utilisé comme une partie de XSL, un langage de feuille de style employé dans XML.

fichier externe : Fichier existant dans un système de fichiers hors DB2.

fonction définie par l'utilisateur (UDF) : Fonction définie sur le système de gestion de base de données pouvant être utilisée par la suite dans des requêtes SQL. Il peut s'agir d'une des fonctions suivantes :

- Une fonction externe, dans laquelle le corps de la fonction est écrit dans un langage de programmation dont les arguments sont des valeurs scalaires, avec renvoi de résultat scalaire par appel.
- Une fonction dérivée, mise en oeuvre par une fonction intégrée ou une fonction UDF déjà connue dans le SGBD. Cette fonction peut être une fonction scalaire ou une fonction

(d'agrégation) de colonne, et renvoie une valeur unique à partir d'un ensemble de valeurs (par exemple, MAX ou AVG).

fonction de transtypage : Fonction utilisée pour convertir les instances d'un type de données (source) en instance d'un autre type de données (cible). En général, une fonction de transtypage a le nom du type de données cible. Elle ne possède qu'un seul argument dont le type correspond au type de données source. Le type renvoyé est le type de données source.

fonction de transtypage par défaut : Fonction qui convertit le type de base SQL en type UDT.

fonction multi-référencée : Nom de fonction pour laquelle il existe plusieurs instances de fonction.

fonction scalaire : Opération SQL qui renvoie une seule valeur à partir d'une autre valeur et qui est exprimée sous la forme d'un nom de fonction, suivi d'une liste d'arguments entre parenthèses.

fonction UDF XML : Fonction UDF DB2 fournie par l'Extension XML.

ID racine (ROOT ID) : Identificateur unique sous lequel toutes les tables annexes sont associées à la table de l'application.

indexation en arbre B : Mode d'indexation natif fourni par le moteur DB2. Il construit les entrées d'index dans une structure en arbre B. Prend en charge les types de données de base de DB2.

indexer : Ensemble de pointeurs classés de façon logique par les valeurs d'une clé. Les index fournissent un accès rapide aux données et peuvent imposer les règles d'unicité aux lignes de la table.

index structurel : Index permettant d'indexer des chaînes de texte en fonction de l'arborescence du document XML, à l'aide de l'Extension Texte DB2.

Interface de programmation d'applications (API) :

(1) Interface fonctionnelle fournie par le système d'exploitation ou par un logiciel sous licence à commander. Permet à un programme d'application écrit en langage évolué d'utiliser des données ou des fonctions spécifiques au système d'exploitation ou aux logiciels sous licence.

(2) Dans DB2, fonction figurant dans l'interface. Par exemple, l'API ″Get error message″.

Java Database Connectivity (JDBC) : Interface API ayant les mêmes caractéristiques qu'Open Database Connectivity (ODBC) mais spécialement conçue pour les applications de base de données Java. Pour les bases de données ne disposant pas d'un pilote JDBC, JDBC a prévu une passerelle de JDBC à ODBC, un mécanisme convertissant JDBC en ODBC : JDBC présente l'API JDBC aux applications de base de données Java et la convertit en ODBC. JDBC a été développé par Sun Microsystems, Inc. et plusieurs partenaires et fournisseurs.

JDBC : Java Database Connectivity.

jointure : Opération relationnelle permettant l'extraction de données dans deux ou plusieurs tables en fonction des correspondances entre les valeurs de colonnes.

LOB : Objet LOB (Large OBject).

mappage de noeud RDB : Emplacement du contenu d'un élément XML ou valeur d'un attribut XML, définis par le noeud RDB. L'Extension XML utilise ce mappage pour déterminer l'emplacement où stocker ou extraire les données XML.

mappage SQL : Définition de la relation entre le contenu d'un élément XML ou la valeur d'un attribut XML et des données relationnelles, utilisant une ou plusieurs instructions SQL et le modèle de données XSLT. L'Extension XML utilise cette définition pour déterminer l'emplacement où stocker ou extraire les données XML. Le mappage SQL est défini avec l'élément SQL_stmt dans la DAD.

méthode d'accès et de stockage : Permet d'associer des documents XML à une base de données DB2 par le biais de deux méthodes d'accès et de stockage essentielles : les colonnes XML et les collections XML. Voir aussi *colonne XML* et *collection XML*.

Modèle de données XPath : Structure arborescente permettant de modéliser et de parcourir un document XML via des noeuds.

navigateur : Voir *navigateur Web*.

navigateur Web : Programme client qui émet des demandes à un serveur Web et affiche les informations renvoyées par le serveur.

noeud : En partitionnement de base de données, synonyme de partition de base de données.

noeud d'attribut (attribute_node) : Représentation d'un attribut d'élément.

noeud de base de données relationnelle (RDB_node) : Noeud contenant une ou plusieurs définitions d'élément destinées aux tables et aux colonnes et conditions facultatives. Les tables et les colonnes sont utilisées pour définir le mode de stockage des données XML dans la base de données. La condition indique soit les critères de sélection des données XML, soit le mode de jointure des tables d'une collection XML.

noeud d'élément (element_node) : Représentation d'un élément. Un noeud d'élément peut être l'élément racine ou un élément enfant.

noeud d'élément supérieur : Représentation de l'élément racine dans le document XML de la DAD.

noeud de texte (text_node) : Représentation du texte CDATA d'un élément.

noeud RDB (RDB_node) : Noeud de base de données relationnelle.

objet : En programmation orientée objet, abstraction se composant de données et d'opérations associées à ces données.

objet CLOB : Chaîne de caractères mono-octet dont la taille peut aller jusqu'à 2 Go. Les objets CLOB sont associés à une page de codes. Les objets texte qui contiennent des caractères mono-octets sont stockés dans une base de données DB2 sous forme d'objets CLOB.

objet DBCLOB : Chaîne de caractères double octets, ou association de caractères mono et double octets, dont la taille peut aller jusqu'à 2 Go. Les objets DBCLOB sont associés à une page de codes. Les objets texte qui contiennent des caractères à double octets sont stockés dans une base de données DB2 sous forme d'objets DBCLOB.

objet LOB : Séquence d'octets dont la longueur peut aller jusqu'à 2 Go. Un objet LOB peut être de trois types : *objet BLOB*, *objet CLOB* ou *objet DBCLOB*.

objet XML : Equivalent de document XML.

occurrences multiples (multi-occurrence) : Option indiquant si un élément ou un attribut de colonne peut être utilisé plusieurs fois dans un document. Est précisée dans le fichier DAD.

ODBC : Open Database Connectivity.

Open Database Connectivity (ODBC) : Interface API standard pour l'accès aux données de systèmes de gestion de base de données (SGBD) relationnels et non relationnels. Avec cette interface, les applications de base de données peuvent accéder aux données stockées dans des SGBD sur une large gamme d'ordinateurs, même si chaque SGBD utilise un format de stockage de données et une interface de programmation différents. ODBC est basé sur l'interface CLI (Call Level Interface), spécification de X/Open SQL Access Group, développée par Digital Equipment Corporation (DEC), Lotus, Microsoft et Sybase. Par opposition à *Java Database Connectivity (JDBC)*.

partition : Zone de stockage de taille définie.

prédicat : Elément d'une condition de recherche exprimant ou impliquant une comparaison.

procédure : Voir *procédure mémorisée*.

procédure mémorisée : Bloc de structures de traitement et d'instructions SQL imbriquées stocké dans une base de données et pouvant être appelé par un nom. Les procédures mémorisées permettent à un programme d'application de s'exécuter en deux parties. Une partie s'exécutant sur le client et l'autre sur le serveur. Ceci permet via un seul appel d'obtenir plusieurs accès à la base de données.

recherche de section : Recherche de texte intervenant dans une section qui peut être définie par l'application. Pour prendre en charge une recherche structurelle, une section peut être définie par le chemin d'emplacement abrégé de Xpath.

recherche intégrale : Dans l'Extension Texte DB2, recherche de chaînes de texte n'importe où, sans considération de la structure du document.

référentiel des DTD : Table DB2, appelée DTD_REF, dans laquelle chaque ligne représente une DTD accompagnée d'autres informations sur les métadonnées.

releveur de coordonnées : Pointeur pouvant être utilisé pour localiser un objet. Dans DB2, le releveur de coordonnées LOB est le type de données permettant de localiser les objets LOB.

requête : Demande d'informations émanant de la base de données et basée sur certaines conditions. Par exemple, une requête peut être une demande de liste contenant tous les clients d'une table dont le solde est supérieur à 1000.

schéma : Collection d'objets de base de données : par exemple, des tables, des vues, des index ou des déclencheurs. Il fournit une classification logique des objets de base de données.

schéma de mappage : Définition du mode de représentation des données XML dans une base de données relationnelle. Le schéma de mappage choisi est précisé dans la DAD. L'Extension XML comporte deux types de schémas de mappage : le *mappage SQL* et le *mappage de noeud RDB (noeud de base de données relationnelle).*

source de données : Gestionnaire de données local ou éloigné capable de prendre en charge l'accès aux données via un pilote ODBC acceptant les API ODBC.

sous-requête : Instruction SELECT complète, utilisée dans une condition de recherche d'une instruction SQL.

SQL imbriqué : Ensemble d'instructions SQL codées à l'intérieur d'un programme d'application. Voir *SQL statique.*

SQL statique : Instructions SQL imbriquées dans un programme et préparées lors du processus de préparation du programme avant son exécution. Après avoir été préparée, une instruction SQL statique n'est pas modifiable, bien que les valeurs des variables hôte spécifiées dans l'instruction puissent changer.

système de fichiers local : Système de fichiers propre à DB2

table annexe : Tables complémentaires créées par l'Extension XML pour améliorer les performances lors de la recherche d'éléments ou d'attributs dans une colonne XML.

table de métadonnées : Voir *table de gestion*.

table de référence DTD (table DTD_REF) : Table contenant des DTD, utilisés pour valider des documents XML et aider les applications à définir une DAD. Les utilisateurs peuvent insérer leurs propres DTD dans la table DTD_REF. Cette table est créée lorsqu'une base de données est compatible XML.

table de résultats : Table contenant des lignes résultant d'une requête SQL ou de l'exécution d'une procédure mémorisée.

table DTD_REF : table de référence DTD.

tables de gestion : Tables utilisées par une extension DB2 pour traiter les demandes d'utilisateurs portant sur des objets XML. Certaines tables de gestion identifient les tables et les colonnes utilisateur activées pour une extension. D'autres tables de gestion contiennent

des informations sur les attributs d'objets dans les colonnes activées. Synonyme de table de métadonnées.

table utilisateur : Table créée pour une application et utilisée par elle.

table XML : Table d'application comprenant une ou plusieurs colonnes XML.

type de données : Attribut de colonnes et de littéraux.

Type de données UDT XML : Type de données UDT fourni par l'Extension XML.

type défini par l'utilisateur (UDT) : Type de données qui n'est pas natif du gestionnaire de bases de données, créé par un utilisateur. Voir *type distinct*.

type distinct : Voir *type défini par l'utilisateur*.

UDF : Voir *fonction définie par l'utilisateur*.

UDT : Voir *type défini par l'utilisateur*.

Uniform Resource Locator (URL) : Adresse indiquant un serveur HTTP et éventuellement un répertoire et un nom de fichier, par exemple : http://www.ibm.com/data/db2/extenders.

UNION : Opération SQL combinant les résultats de deux instructions SELECT. UNION est souvent utilisée pour la fusion de listes de valeurs obtenues à partir de plusieurs tables.

URL : Uniform Resource Locator.

validation : Processus d'utilisation d'une DTD pour vérifier la validité d'un document XML et permettre des recherches structurées sur les données XML. La DTD est stockée dans le référentiel des DTD.

vue de jointure : Vue DB2 créée par l'instruction ″CREATE VIEW″, réalisant la jointure de plusieurs tables.

vue par défaut : Représentation de données dans laquelle une table XML et les tables annexes associées sont jointes.

XML : eXtensible Markup Language.

XML Path Language (XPath) : Langage permettant de se rapporter à des parties d'un document XML. XML Path Language est conçu pour être utilisé par XSLT. Tous les chemins d'emplacement peuvent être exprimés à l'aide de la syntaxe définie pour XPath.

XPath : Langage permettant de se rapporter à des parties d'un document XML.

XSL : XML Stylesheet Language.

XSLT : XML Stylesheet Language Transformation.

Index

Caractères spéciaux

A

B

C

D

E

F

G

I

J

L

M

N

O

P

R

S

T

U

V

X