Novell Pilote DirXML® pour JDBC*

1.6.2

5 mai 2004

www.novell.com

IMPLEMENTATION GUIDE
(GUIDE D'IMPLÉMENTATION)

Novell®

Mentions légales

Novell, Inc.
1800 South Novell Place
Provo, UT 84606
États-Unis

www.novell.com

DirXML Driver for JDBC Implementation Guide (Guide d'implémentation du pilote DirXML pour JDBC)
5 mai 2004

Documentation en ligne : pour accéder à la documentation en ligne de ce produit (et d'autres produits Novell) et obtenir les mises à jour, consultez le site www.novell.com/documentation.

Marques commerciales de Novell

DirXML est une marque déposée de Novell, Inc. aux États-Unis et dans d'autres pays.
eDirectory est une marque de Novell, Inc.
NetWare est une marque déposée de Novell, Inc. aux États-Unis et dans d'autres pays.
Novell est une marque déposée de Novell, Inc. aux États-Unis et dans d'autres pays.
Nsure est une marque de Novell, Inc.

Autres marques commerciales

Toutes les marques commerciales de fabricants tiers appartiennent à leur propriétaire respectif.

Sommaire

À propos de ce guide

Le pilote DirXML® pour JDBC* (Java* Database Connectivity) constitue une solution générique pour la synchronisation des données entre Novell® eDirectory™ et des bases de données relationnelles.

Ce guide présente la technologie du pilote et fournit des instructions de configuration.

Documentation supplémentaire

Pour la documentation sur l'utilisation d'Identity Manager et des autres pilotes, reportez-vous au site Web de la documentation d'Identity Manager (http://www.novell.com/documentation/french/dirxmldrivers).

Mises à jour de la documentation

Vous trouverez la version la plus récente de ce document sur le site Web de la documentation d'Identity Manager (http://www.novell.com/documentation/french/dirxmldrivers/index.html).

Conventions utilisées dans la documentation

Dans cette documentation, le symbole « supérieur à » (>) est utilisé pour séparer deux opérations dans une étape de procédure ainsi que deux éléments dans un chemin de références croisées.

Le symbole de marque (®, ™, etc.) indique une marque de Novell. L'astérisque (*) indique une marque commerciale de fabricant tiers.

Commentaires de l'utilisateur

Vos commentaires et suggestions sur le présent guide et sur les autres documents qui accompagnent Novell Nsure™ Identity Manager nous intéressent. Pour nous contacter, envoyez-nous un message électronique à l'adresse suivante : proddoc@novell.com.

1 Présentation du pilote DirXML pour JDBC

Le pilote DirXML® pour JDBC (Java Database Connectivity), ci-après désigné comme « le pilote », constitue une solution générique pour la synchronisation des données entre Novell® eDirectory™ et des bases de données relationnelles accessibles par JDBC.

La principale qualité de ce pilote réside dans sa nature générique. Contrairement à la plupart des pilotes DirXML qui assurent l'interface avec une application bien définie unique, ce pilote permet de communiquer avec la plupart des bases de données relationnelles et applications hébergées par une base de données.

Présentation

Cette section contient des informations sur les sujets suivants :

Nouvelles fonctionnalités

Fonctionnalités du pilote

- La prise en charge des attributs référentiels peut désormais être désactivée.

Corrections de bogues du pilote

- La prise en charge des attributs référentiels peut désormais être désactivée via le paramètre de pilote « Enable Referential Support? (Activer la prise en charge des référentiels ?) ». Ce pilote est ainsi compatible en amont avec la version précédente 1.0 du pilote.
- Il est désormais possible d'ajouter des objets sur le canal Éditeur sans éléments enfant <add-attr>.
- Les événements d'<ajout> de publication suivis d'événements de <modification> ne sont plus optimisés. Cela permet de garantir la compatibilité en amont avec le pilote 1.5.

Nouvelles fonctionnalités d'Identity Manager

Pour plus d'informations sur les nouvelles fonctionnalités d'Identity Manager, reportez-vous au *Nsure Identity Manager 2 Administration Guide (Guide d'administration de Nsure Identity Manager 2)* (http://www.novell.com/documentation/french/dirxml20/admin/data/alxnk27.html).

Concepts de pilote

Il est important de connaître les termes et les concepts suivants avant d'installer et de configurer le pilote :

Pilote DirXML pour JDBC

Le pilote se compose de trois fichiers : JDBCShim.jar, JDBCUtil.jar et CommonDriverShim.jar. Outre ces fichiers, il vous faudra un pilote JDBC tiers pour communiquer avec chaque base de données.

Pilote JDBC tiers

L'une des nombreuses implémentations JDBC utilisées par le pilote pour communiquer avec une base de données particulière. Par exemple, classes12.zip est l'un des pilotes JDBC d'Oracle.

Types de pilote JDBC

Il existe quatre types de pilote JDBC tiers :

1. Un pilote JDBC tiers partiellement Java qui communique indirectement avec une base de données via un pilote ODBC. Les pilotes de type 1 servent de pont JDBC-ODBC. Sun propose un pilote de pont JDBC-ODBC pour une utilisation expérimentale et pour les situations dans lesquelles aucun autre pilote JDBC tiers n'est disponible.
2. Un pilote JDBC tiers partiellement Java qui communique indirectement avec une base de données via son interface de programmation d'application cliente native.
3. Un pilote JDBC tiers pur Java qui communique indirectement avec une base de données via un serveur d'applications intermédiaire.
4. Un pilote JDBC tiers pur Java qui communique directement avec une base de données.

Les pilotes des types 3 et 4 sont généralement plus stables que les pilotes de types 1 et 2. Les pilotes des types 1 et 2 sont généralement plus rapides que les pilotes de types 3 et 4. Les pilotes des types 2 et 3 sont généralement plus sûrs que les pilotes de types 1 et 4. Si vous choisissez d'utiliser un pilote de type 1 ou de type 2, vous devez utiliser le chargeur distant pour garantir l'intégrité du processus d'annuaire.

Schéma d'annuaire

Ensemble des classes et attributs d'objet de l'annuaire. Par exemple, la classe User et l'attribut Given Name d'eDirectory font partie du schéma eDirectory.

Schéma d'application

Ensemble des classes et attributs d'une application. Comme les bases de données n'ont aucune notion de classes ou d'attributs, le pilote assigne les classes eDirectory à des vues ou des tables et les attributs eDirectory à des colonnes.

Schéma de synchronisation

Le schéma de base de données visible pour le pilote.

Classe de base de données logique

L'ensemble de tables ou de vues utilisées pour représenter une classe eDirectory dans une base de données.

Concepts de base de données

La section suivante définit les concepts importants relatifs aux bases de données ci-dessous :

- « Schéma de base de données », page 13
- « Langage de manipulation de données », page 13
- « Langage de définition de données », page 14
- « Transactions », page 14
- « Déclencheurs », page 15
- « Colonnes d'identité/Séquences », page 15
- « Procédures/fonctions stockées », page 16

Schéma de base de données

Un schéma de base de données est un ensemble d'objets de base de données tels que des tables, des vues, des procédures stockées, etc., qui sont la propriété d'un utilisateur de base de données particulier. La propriété est souvent exprimée par une notation à point du type `dirxml.emp`, où `dirxml` est le nom de l'utilisateur de base de données qui est propriétaire de la table `emp`. Tous les objets de base de données qui appartiennent à `dirxml` constituent le schéma de base de données dirxml.

Langage de manipulation de données

Les instructions en langage DML (Data Manipulation Language - Langage de manipulation de données) sont des éléments hautement normalisés qui permettent de manipuler des données de la base de données. Elles sont fondamentalement les mêmes quelle que soit la base de données utilisée.

L'exemple suivant contient plusieurs instructions DML :

```
SELECT * FROM emp;
INSERT INTO emp(lname) VALUES('Doe');
UPDATE emp SET fname = 'John' WHERE empno = 1;
```

Remarque : les exemples utilisés dans l'ensemble du présent guide d'implémentation s'appliquent à la base de données Oracle*.

Langage de définition de données

Les instructions en langage DDL (Data Definition Language - Langage de définition de données) permettent de manipuler des objets de base de données comme des tables, des index, des comptes utilisateur, etc. Les instructions DDL sont propriétaires et diffèrent largement selon les bases de données.

L'exemple suivant contient une instruction DDL :

```
CREATE TABLE emp
(
     empno NUBMER(8),
     fname VARCHAR2(64),
     lname VARCHAR2(64)
);

CREATE USER dirxml IDENTIFIED BY novell;
```

Transactions

Une transaction est une opération de base de données atomique qui se compose d'une ou de plusieurs instructions. Lorsqu'une transaction est terminée, toutes les instructions qu'elle contient sont validées. Lorsqu'une transaction est interrompue ou que l'une de ses instructions est erronée, elle est restaurée à son état initial. Dans ce cas, la base de données est rétablie à son état antérieur, celui qu'elle avait avant le début de la transaction.

Les transactions sont soit manuelles (définies par l'utilisateur), soit automatiques. Les transactions manuelles peuvent se composer d'une ou de plusieurs instructions et doivent être explicitement validées. Les transactions automatiques ne comportent qu'une seule instruction et sont implicitement validées après l'exécution de cette instruction.

Transactions manuelles

Les transactions manuelles contiennent généralement plusieurs instructions. En principe, il est impossible de combiner des instructions DDL et des instructions DML dans une transaction manuelle. L'exemple suivant illustre une transaction manuelle :

```
INSERT INTO emp(lname) VALUES('Doe');
UPDATE emp SET fname = 'John' WHERE empno = 1;
COMMIT; /* explicit */
```

Transactions automatiques

Les transactions automatiques se composent d'une seule instruction. Elles sont souvent appelées instructions auto-validées car les modifications sont implicitement validées après chaque instruction. Lorsqu'une instruction est exécutée automatiquement, elle est indépendante de toute autre instruction. L'exemple suivant illustre une transaction automatique :

```
INSERT INTO emp(lname) VALUES('Doe');
/* COMMIT; implicit */
```

Déclencheurs

Un déclencheur de base de données est une logique programmable associée à une table qui se déclenche ou s'exécute dans certaines conditions. Les déclencheurs servent souvent à créer des effets secondaires dans une base de données. Voici un exemple de déclencheur de base de données sur une table `emp`.

```
CREATE TABLE emp
(
     empno NUMBER(8),
     fname VARCHAR(64),
     lname VARCHAR(64)
);

CREATE TRIGGER t_emp_insert
     AFTER INSERT ON emp
     FOR EACH ROW

BEGIN
     UPDATE emp SET fname = 'John';
END;
```

Lorsqu'une instruction est exécutée sur une table qui comporte des déclencheurs, un déclencheur se déclenche si l'instruction répond aux conditions qu'il indique. Par exemple, dans la table ci-dessus, si l'instruction d'insertion suivante était exécutée,

```
INSERT INTO emp(LNAME) VALUES('Doe')
```

Le déclencheur `t_emp_insert` se déclencherait après l'exécution de l'instruction d'insertion et l'instruction de mise à jour suivante serait également exécutée :

```
UPDATE emp SET fname = 'John'
```

Un déclencheur peut généralement être exécuté avant ou après l'instruction qui l'a déclenché. Les instructions qui sont exécutées dans le cadre d'un déclencheur de base de données sont généralement incluses dans la même transaction que l'instruction déclenchante. Dans l'exemple ci-dessus, les instructions insert et update seraient validées ou annulées ensemble.

Colonnes d'identité/Séquences

Les colonnes d'identité et les séquences sont utilisées pour générer des valeurs de clé primaire uniques.

Une colonne d'identité est une colonne qui s'incrémente automatiquement et qui est utilisée pour identifier de manière unique une ligne dans une table. Les valeurs des colonnes d'identité sont automatiquement renseignées lors de l'insertion d'une ligne dans une table.

Un objet de séquence est un compteur qui peut être utilisé pour identifier de manière unique une ligne dans une table. Contrairement à une colonne d'identité, un objet de séquence n'est pas lié à une seule table. Si toutefois il est utilisé par une seule table, un objet de séquence peut être utilisé pour obtenir un résultat équivalent.

Voici un exemple d'objet de séquence :

```
CREATE SEQUENCE seq_empno
    START WITH 1
    INCREMENT BY 1
    NOMINVALUE
    NOMAXVALUE
    CACHE 100
    ORDER;
```

Procédures/fonctions stockées

Une procédure ou fonction stockée est une logique programmable enregistrée dans une base de données. Contrairement aux déclencheurs, les procédures ou fonctions stockées ne sont pas associées à une table. Elles peuvent être appelées à partir de n'importe quel contexte ou presque.

L'abonné peut utiliser des procédures ou des fonctions stockées pour extraire des valeurs de clé primaire à partir de lignes insérées dans des tables, afin de créer des associations. Ces procédures ou fonctions peuvent également être appelées à partir d'instructions SQL ou de déclencheurs incorporés.

La distinction entre procédures stockées et fonctions varie selon la base de données. En principe, les deux peuvent renvoyer des données en sortie. Ce sont leurs modes de renvoi qui diffèrent. En effet, les procédures stockées renvoient généralement des valeurs par l'intermédiaire de paramètres, tandis que les fonctions les renvoient par l'intermédiaire d'un ensemble de résultats.

Voici un exemple d'une procédure stockée :

```
CREATE
PROCEDURE sp_empno
(
    io_empno IN OUT NUMBER,
    i_fname IN VARCHAR2
)
IS
BEGIN
    IF (io_empno IS NULL) THEN
        SELECT seq_empno.nextval INTO io_empno FROM DUAL;
END IF;
END sp_empno;
```

Modèles de synchronisation de données

Le pilote prend en charge deux modèles de synchronisation de données : direct et indirect. Les sections suivantes décrivent le fonctionnement de la synchronisation directe et indirecte sur les canaux Abonné et Éditeur.

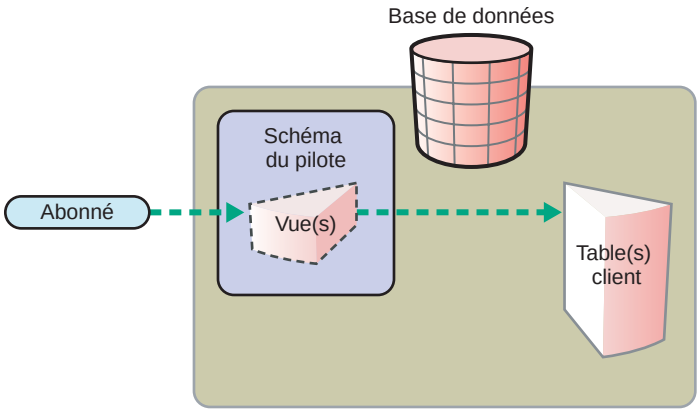
Synchronisation directe

Le mode direct utilise des vues pour synchroniser les informations entre eDirectory et une base de données. Les informations suivantes expliquent le fonctionnement de la synchronisation directe sur les canaux Abonné et Éditeur.

Les scénarios suivants peuvent concerner une ou plusieurs tables ou vues client.

Canal Abonné

Synchronisation directe sur le canal Abonné

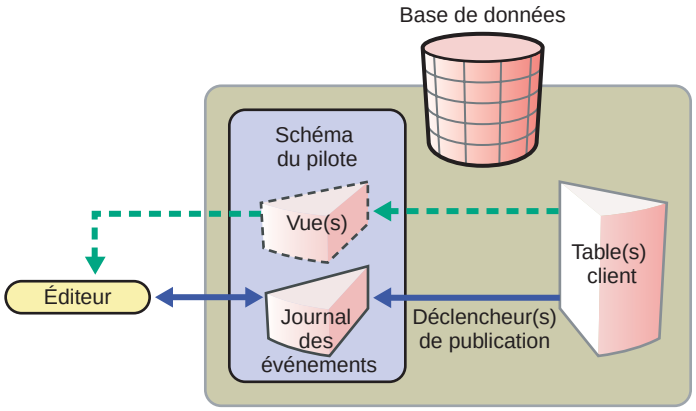


Le canal Abonné met à jour les tables client par l'intermédiaire d'une vue dans le schéma du pilote. Une vue peut servir à une synchronisation directe avec les tables client.

Remarque : la synchronisation directe sans vue n'est possible que si toutes les colonnes qui présentent un intérêt sont situées dans la même table et si cette table comporte une contrainte de clé primaire explicite.

Canal Éditeur

Synchronisation directe sur le canal Éditeur



Lorsqu'une table client est mise à jour, des déclencheurs du canal Éditeur insèrent des lignes dans la table de consignation des événements. Le canal Éditeur lit ensuite les lignes insérées puis met à jour eDirectory.

En fonction du contenu des lignes lues dans la table de consignation des événements, le canal Éditeur peut avoir à extraire des informations supplémentaires de la vue avant de mettre à jour eDirectory. Après la mise à jour d'eDirectory, le canal Éditeur supprime les lignes ou les marque comme traitées.

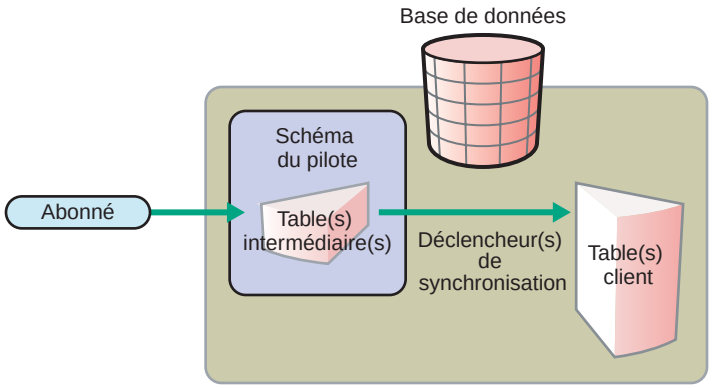
Synchronisation indirecte

Le mode indirect utilise des tables intermédiaires pour synchroniser les informations entre eDirectory et une base de données. Les informations suivantes expliquent le fonctionnement de la synchronisation indirecte sur les canaux Abonné et Éditeur.

Les scénarios suivants peuvent concerner une ou plusieurs tables ou tables intermédiaires client.

Canal Abonné

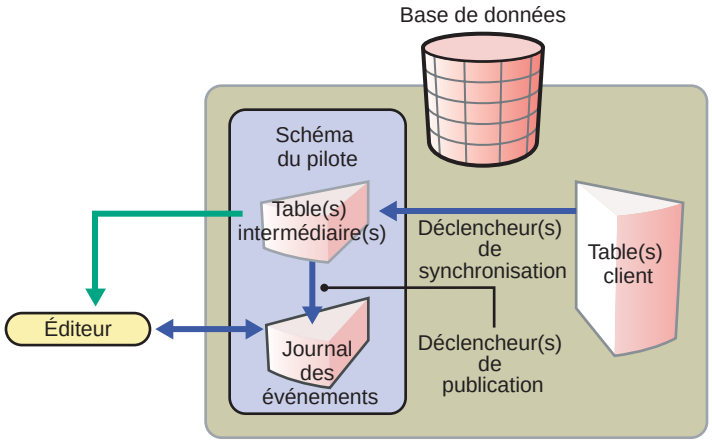
Synchronisation indirecte sur le canal Abonné



Le canal Abonné met à jour la table intermédiaire dans le schéma du pilote. Les déclencheurs de synchronisation mettent directement à jour les tables client dans le reste de la base de données.

Canal Éditeur

Synchronisation indirecte sur le canal Éditeur



Une fois les tables client mises à jour, les déclencheurs de synchronisation mettent à jour les tables intermédiaires. Les déclencheurs du canal Éditeur insèrent alors une ou plusieurs lignes dans la table de consignation des événements. Le canal Éditeur lit ensuite les lignes insérées puis met à jour eDirectory.

En fonction du contenu des lignes lues dans la table de consignation des événements, le canal Éditeur peut avoir à extraire des informations supplémentaires des tables intermédiaires avant de mettre à jour eDirectory. Après la mise à jour d'eDirectory, le canal Éditeur supprime les lignes ou les marque comme traitées.

2 Conditions préalables applicables au pilote

Les sections suivantes contiennent des informations importantes à consulter avant d'installer et de configurer le pilote.

Conditions préalables concernant le pilote

Le pilote DirXML® pour JDBC exige les éléments suivants :

- ❑ Novell Nsure™ Identity Manager 2
- ❑ Machine virtuelle Java (JVM*) 1.2 ou version ultérieure
- ❑ Un pilote JDBC d'un fabricant tiers

Plates-formes prises en charge

Le pilote s'exécute sur toutes les plates-formes dotées de fonctions Identity Manager, dont Windows* NT*/2000, NetWare®, Solaris* et Linux*.

Bases de données prises en charge

Le pilote utilise l'API JDBC 1.0 pour exécuter des instructions SQL et extraire des métadonnées d'une base de données. À ce titre, une base de données doit être accessible à JDBC. Les bases de données suivantes ont été testées et sont recommandées pour une utilisation avec ce produit :

Base de données	Version
IBM* DB2 Universal Database (UDB)	7.2 ou version ultérieure
Microsoft* SQL Server 2000	Service Pack 2 ou version ultérieure
Microsoft SQL Server 7	Service Pack 4

Base de données	Version
Oracle 8i	version 3 (8.1.7)
Oracle 9i	version 2 (9.2.0.1) ou ultérieure
Sybase* Adaptive Server Enterprise (ASE)	12.5 ou version ultérieure
MySQL*	3.23
Informix* Dynamic Server (IDS)	9.30 ou version ultérieure

Vous pouvez utiliser d'autres bases de données, à condition qu'elles répondent aux conditions minimales requises suivantes :

- Prise en charge de la grammaire de premier niveau SQL-92.
- Prise en charge des déclencheurs ou d'une fonctionnalité d'audit permettant la capture et la réplication d'événements (sur le canal Éditeur uniquement).

Pilotes JDBC de fabricants tiers recommandés

Nous vous recommandons d'utiliser des pilotes JDBC tiers de type 3 ou 4, chaque fois que c'est possible. Nous vous recommandons aussi d'utiliser les dernières versions de ces pilotes. Si vous choisissez d'utiliser un pilote de type 1 ou de type 2, vous devez utiliser le chargeur distant pour garantir l'intégrité du processus d'annuaire.

Les pilotes tiers suivants ont été testés et sont recommandés pour une utilisation avec le pilote DirXML pour JDBC :

Nom du pilote	Version
Pilote JDBC Oracle 8i	8.1.7.1
Pilote JDBC Oracle 9i	9.2.0.1 ou version ultérieure
Pilote jDriver de type 4 BEA* Weblogic* pour Microsoft SQL Server 7/2000	5.1.0, Service Pack 11 ou version ultérieure
Pilote JDBC jConnect Sybase	5.5 ou version ultérieure
Pilote Microsoft SQL Server 2000 pour JDBC	2.2 ou version ultérieure
Pilote JDBC Informix	9.3 ou version ultérieure
MySQL Connector/J	2.0.14 ou version ultérieure
Pilote JDBC de type 3 IBM pour DB2 UDB	7.2 ou version ultérieure

Les pilotes JDBC tiers suivants ont été testés, mais ne sont pas recommandés pour une utilisation avec ce produit :

- Pilote de pont JDBC-ODBC de type 1 Sun (JRE 1.2)

Nous vous conseillons vivement d'utiliser les pilotes tiers recommandés dans la mesure du possible.

Conditions minimales requises pour les pilotes JDBC de fabricants tiers

Il se peut que le pilote ne fonctionne pas avec tous les pilotes de fabricants tiers. Si vous choisissez d'utiliser un autre pilote de fabricant tiers, celui-ci doit répondre aux conditions suivantes pour fonctionner avec le pilote DirXML pour JDBC :

- Prise en charge des méthodes de métadonnées requises.

  Pour obtenir la dernière liste des appels de méthode java.sql.DatabaseMetaData obligatoires et facultatifs effectués par le pilote, reportez-vous à l'Annexe D, « Méthodes java.sql.DatabaseMetaData », page 91. Cette liste de conditions pourra être complétée dans les versions ultérieures. Toutes les méthodes java.sql.DatabaseMetaData en tant que telles doivent être prises en charge. Si le pilote tiers ne remplit pas ces conditions, il pourra être nécessaire d'en acquérir un autre par la suite.

- Renvoi de données précises des instructions de sélection.

- Exécution correcte des instructions d'insertion, de mise à jour et de suppression émises par le pilote.

Pour obtenir une liste des méthodes JDBC utilisées par le pilote, reportez-vous à l'Annexe E, « Méthodes JDBC 1.0 », page 93. Consultée parallèlement à la documentation des pilotes de fabricants tiers, cette liste peut permettre d'identifier d'éventuelles incompatibilités.

Considérations relatives à l'utilisation d'autres pilotes JDBC de fabricants tiers

- Comme le pilote dépend des pilotes de fabricants tiers, tout bogue survenant dans ces pilotes est susceptible d'empêcher le bon fonctionnement du pilote. Pour vous aider à déboguer les pilotes de fabricants tiers, la sortie de trace du pilote a été améliorée pour inclure des messages de trace JDBC de niveau API (niveau 5) et de pilote de fabricants tiers (niveau 6).

- La prise en charge des procédures ou fonctions stockées et les capacités de connexion (en particulier, la reconnexion) sont des sources d'échec possibles.

Utilisation du pilote de pont JDBC-ODBC de Sun

En raison de l'instabilité accrue liée à l'utilisation d'un pilote ODBC et des problèmes connus posés par le pilote de pont JDBC-ODBC du JRE (Java Runtime Environment - Environnement d'exécution Java) 1.3.*x* , nous vous recommandons vivement d'utiliser un pilote JDBC pur Java (type 3 ou 4) lorsque cela est possible. Si vous choisissez d'utiliser un pilote de type 1 ou de type 2, vous devez utiliser le chargeur distant pour garantir l'intégrité du processus d'annuaire.

Le principal inconvénient à utiliser un pilote de pont JDBC de type 1 et un pilote ODBC natif est une plus grande instabilité. Des erreurs dans les bibliothèques natives importées du pilote ODBC par l'intermédiaire du pont JDBC risquent de mettre l'annuaire hors service.

Il se peut que le pilote et le pilote de pont JDBC-ODBC ne fonctionnent pas avec tous les pilotes ODBC de fabricants tiers. La liste des conditions requises pour les pilotes JDBC de fabricants tiers s'applique également aux pilotes ODBC. Pour plus de détails, reportez-vous à « Pilotes JDBC de fabricants tiers recommandés », page 20.

- Si vous optez pour le pilote ODBC de Microsoft pour SQL Server (SQLSRV32.DLL), nous vous recommandons d'installer la dernière version de Microsoft Data Access Components (MDAC). Vous pouvez télécharger gratuitement MDAC sur le site Web de Microsoft. (http://www.microsoft.com/data/download.htm).

- Le pilote de pont compris dans l'environnement JRE 1.3.x contient un défaut connu relatif aux paramètres de procédure stockée IN OUT. L'appel d'une procédure stockée avec des paramètres IN OUT se traduit par une violation de l'accès à la mémoire et par la mise hors service de l'annuaire. Pour éviter ce problème, il est recommandé d'utiliser la version 1.2.x du JRE avec Identity Manager. Cette solution a néanmoins pour effet de réduire les performances de tous les pilotes exécutés sur le serveur. Identity Manager prend en charge uniquement l'utilisation de Hotspot avec JRE version 1.3.x ou ultérieure. JRE version 1.4.x JRE n'a pas encore été testé avec le pilote.

Sécurité

Pour garantir l'établissement d'une connexion sûre entre le pilote et un pilote tiers, nous vous recommandons d'exécuter le pilote à distance.

Quand il n'est pas possible d'exécuter le pilote à distance, vous souhaiterez peut-être utiliser un pilote JDBC de type 2 ou 3. Ces types de pilote offrent souvent un plus haut niveau de sécurité via des serveurs d'applications intermédiaires ou des interfaces de programmation clientes que les autres types de pilote JDBC.

Problèmes connus

Cette section répertorie les problèmes connus du pilote.

Généralités

- Certaines bases de données, telles que Sybase et DB2, ont des formats d'horodatage propriétaire qui ne peuvent pas être analysés par la classe java.sql.Timestamp.

  Lors de la synchronisation des colonnes d'horodatage issues de ces bases de données, les valeurs d'horodatage placées dans la table de consignation des événements doivent être au format réglementaire ODBC (c'est-à-dire, aaaa-mm-jj hh:mm:ss.fffffffff). Une alternative consiste à convertir ces valeurs au format réglementaire ODBC via des feuilles de style. Cependant, lorsque les horodatages sont utilisés comme clés primaires, les valeurs d'horodatage doivent être placées dans la table de consignation des événements au format réglementaire ODBC. Les valeurs d'horodatage peuvent être reformatées dans la base de données à l'aide d'un langage de programmation général, tel que Java, ou le langage de programmation SQL natif de la base de données.

- Quand les syntaxes du temps et de l'horodatage d'eDirectory sont interprétées comme des nombres entiers avec signes, elles ne peuvent stocker de dates antérieures au 1er janvier 1902 ou postérieures au 1er janvier 2038.

IBM DB2

- Après avoir appliqué un kit de mise à jour IBM à votre serveur DB2, vous devez utiliser le fichier db2java.zip mis à jour sur le serveur de base de données sur lequel le pilote est installé. Sinon, vous risquez de recevoir des erreurs de connexion, du type « CLI0601E Invalid statement handle or statement is closed. » (CLI0601E Identificateur d'instruction non valide ou instruction fermée)

Pont JDBC-ODBC

- Le pilote de pont compris dans l'environnement JRE 1.3.x contient un défaut connu relatif aux paramètres de procédure stockée IN OUT. L'appel d'une procédure stockée avec des paramètres IN OUT se traduit par une violation de l'accès à la mémoire et par la mise hors service de l'annuaire. Pour éviter ce problème, il est recommandé d'utiliser la version 1.2.2 du JRE avec Identity Manager. Cette solution a néanmoins pour effet de réduire les performances de tous les pilotes exécutés sur le serveur. Identity Manager prend en charge uniquement l'utilisation de Hotspot avec JRE version 1.3.x ou ultérieure.

Oracle

- Vous pouvez rencontrer des problèmes liés à une utilisation intensive de l'UC lors de l'exécution d'instructions SQL incorporées, sauf si vous associez un attribut `jdbc:type` à chaque élément `<jdbc:statement>`. De façon générale, le problème peut être évité en attribuant la valeur **`single`** au paramètre de pilote Gérer les résultats des instructions.
- La version 8.1.6 du pilote JDBC Oracle comporte un bogue qui influe sur l'acheminement des données via le canal Éditeur. Des valeurs NULL incorrectes sont renvoyées par certains champs dans la table de consignation des événements, alors que leurs valeurs sont en fait non NULL.

  La conséquence directe est que le pilote considère plusieurs lignes comme NOOP (« No Operation ») et les ignore, puis il génère un document de canal Éditeur incomplet. Les versions antérieures peuvent également présenter ce même problème. Nous vous recommandons donc d'utiliser la version 8.1.7 qui est compatible en amont avec la plupart des versions d'Oracle 8.
- Pour vous connecter à d'anciennes versions d'Oracle sur NetWare (par exemple, 8.0.3), vous devez utiliser le pilote JDBC CLASS111.zip fourni sur le CD d'installation d'Oracle.

Microsoft SQL Server

- Le pilote ODBC de Microsoft pour SQL Server renvoie un type java.sql.Types.OTHER ambigu pour les types de données NVARCHAR, NCHAR, NTEXT et UNIQUEIDENTIFIER. Or, le pilote suppose que le type java.sql.Types.OTHER est NVARCHAR, NCHAR ou NTEXT. C'est pour cette raison que le type UNIQUEIDENTIFIER n'est pas pris en charge.
- Le pilote JDBC pour Microsoft SQL Server 2000 génère l'erreur suivante quand le paramètre du pilote Réutiliser les instructions a une valeur autre que **`non`** : « Can't start manual transaction mode because there are cloned (Impossible de démarrer le mode de transaction manuel parce qu'il y a des éléments clonés). »

Sybase

- Pour assurer un comportement de remplissage et de troncature conforme à la norme ANSI des valeurs binaires, les colonnes binaires (autres que les images) doivent être exactement de la même taille que l'attribut eDirectory qui leur est assigné, contraintes NOT NULL et ajoutées à la règle de création de l'objet Éditeur ou Abonné. Si elles sont contraintes NULL, les zéros de droite, qui sont significatifs pour eDirectory, seront tronqués. Si les colonnes binaires dépassent la taille de leurs attributs eDirectory respectifs, des 0 supplémentaires seront ajoutés à la fin de la valeur.

MySQL

- Les colonnes TIMESTAMP (horodatage), lorsqu'elles sont mises à jour après avoir été initialement définies sur 0 ou NULL, sont toujours définies à la date et l'heure actuelles. Pour compenser ce comportement, nous vous recommandons d'assigner les syntaxes de temps et d'horodatage d'eDirectory aux colonnes DATETIME.
- La publication n'est pas prise en charge. MySQL ne prend pas en charge la requête utilisée par l'éditeur pour extraire des événements de la table de consignation des événements.

Informix

- Les colonnes NUMERIC (numérique) ou DECIMAL (décimale) ne peuvent être utilisées comme clés primaires à moins que l'échelle (c'est-à-dire le nombre de colonnes à droite du point décimal) soit explicitement définie sur zéro lors de la création de la table. Par défaut, l'échelle est définie sur 255.

Limites

La section suivante répertorie les limites connues du pilote.

- Le pilote ne prend pas en charge l'utilisation d'identificateurs de base de données délimités.
- La synchronisation directe (à l'aide de vues) ne prend pas en charge la synchronisation des attributs à valeurs multiples ou référentiels.
- Les bases de données Informix Dynamic Server créées avec l'option LOG MODE ANSI ne sont pas prises en charge. Les bases créées avec cette option utilisent des identificateurs délimités pour les noms d'utilisateur ou de schéma. Pour le moment, le pilote ne prend pas en charge les identificateurs délimités.
- La publication n'est pas prise en charge. MySQL ne prend pas en charge la requête utilisée par l'éditeur pour extraire des événements de la table de consignation des événements.
- Les types de données JDBC 2.0 ne sont pas pris en charge à l'exception de CLOB et BLOB.

3 Installation ou mise à niveau du pilote

Vous trouverez, dans cette section, les informations et les procédures utiles pour installer et mettre à niveau le pilote :

Installation du pilote

Le pilote DirXML® pour JDBC exige Novell® eDirectory™ et une configuration de la base de données. Il est recommandé de configurer votre base de données et de la tester indépendamment du pilote.

Suivez ces instructions s'il n'existe aucune installation précédente du pilote. Après avoir téléchargé l'image du CD, exécutez les instructions suivantes pour installer le pilote :

Installation du pilote

1. Arrêtez eDirectory.
2. Copiez JDBCShim.jar, JDBCUtil.jar et CommonDriverShim.jar dans le répertoire approprié pour votre plate-forme. Inspirez-vous du tableau ci-dessous pour déterminer le répertoire qui convient :

Plate-forme	Chemin de répertoire
NetWare®	SYS:\SYSTEM\LIB
Solaris ou Linux	/usr/lib/dirxml/classes
Windows NT/2000	NOVELL\NDS\LIB

3. Copiez les fichiers de pilote tiers appropriés dans le répertoire que vous avez indiqué à l'étape précédente.

   Remarque : ces fichiers de fabricants tiers ne sont pas fournis avec le pilote. Ils doivent faire l'objet d'une licence correcte, si nécessaire, pour être utilisés dans un environnement de production.

   Le tableau suivant contient des informations pour le téléchargement de pilotes de fabricants tiers, par fournisseur.

Fournisseur	Base de données	Nom(s) de fichier(s)	Instructions de téléchargement
Oracle	Oracle 8i	classes12.zip, nls_charset12.zip	Oracle Technology Network (http://otn.oracle.com/software/tech/java/sqlj_jdbc/content.html) Vous devez tout d'abord vous enregistrer gratuitement sur le site OTN (Oracle Technology Network) d'Oracle. Téléchargez la version 8.1.7.1 ou une version ultérieure.
Oracle	Oracle 9i	classes12.zip, nls_charset12.zip	Oracle Technology Network (http://otn.oracle.com/software/tech/java/sqlj_jdbc/content.html) Vous devez tout d'abord vous enregistrer gratuitement sur le site OTN (Oracle Technology Network) d'Oracle. Téléchargez la version 9.2.0.1 ou une version ultérieure.
BEA Systems	Microsoft SQL Server 7/ 2000	non précisé(s)	Centre de téléchargement de BEA (http://commerce.bea.com/downloads/weblogic_server.jsp) Ce pilote requiert une inscription gratuite qui expire chaque mois. Dans la section JDBC Drivers, téléchargez « MSSQLServer4 Kit, Version 7 and 2000 ». Vous devez transformer vous-même le fichier en archive zip/jar. Pour cela, procédez comme suit : 1. Placez le fichier weblogic\mssqlserver4v70\license\WeblogicLicense.xml dans le répertoire \weblogic\smsqlserver4v70\classes. 2. Zippez ou transformez en archive jar le répertoire...\classes\weblogic, y compris son contenu, ainsi que le fichier ...\classes\WeblogicLicense.xml. Ce répertoire et le fichier XML doivent se trouver à la racine du fichier archive zip/jar.
Microsoft	SQL Server 2000	msbase.jar mssqlserver.jar msutil.jar	Centre de téléchargement de Microsoft (http://www.microsoft.com/sql/downloads/2000/jdbc.asp) Cliquez sur l'option de téléchargement appropriée pour votre plate-forme.
IBM	DB2 Universal Database (UDB) 7.2	db2java.zip	Centre de téléchargement d'IBM (http://www-4.ibm.com/software/data/db2/udb/downloads.html) Ce pilote fait partie du téléchargement gratuit de DB2 UDB Personal Developer. Si vous appliquez des kits de mise à jour, veillez à utiliser le fichier db2java.zip situé sur le serveur qui a été modifié par le correctif et non le fichier obtenu par le téléchargement initial.
Sybase	Adaptive Server Enterprise (ASE) 12.5	jconn2.jar	Centre de téléchargement de Sybase (http://www.sybase.com/downloads) Pour télécharger ce pilote, sélectionnez jConnect for JDBC.

Fournisseur	Base de données	Nom(s) de fichier(s)	Instructions de téléchargement
MySQL	MySQL 3.23	mysql-connector-java-2.0.14-bin.jar	Centre de téléchargement de MySQL (http://www.mysql.com/downloads/api-jdbc.html) Sélectionnez MySQL Connector/J2 ou version ultérieure.
Informix	Informix Dynamic Server (IDS) 9.3	ifxjdbc.jar	non disponible pour téléchargement
Sun	N/D	N/D	Centre de téléchargement de Sun (http://java.sun.com/j2se/downloads.html) Le pilote de pont JDBC-ODBC de Sun est disponible gratuitement en tant que partie intégrante du JRE (Java Runtime Environment).

4. Redémarrez eDirectory.
5. Démarrez ConsoleOne ou iManager.

Importation d'une configuration de pilote

Les configurations de pilotes correspondent aux exemples de configurations uniquement. Il est recommandé d'installer une configuration de pilote et de l'exécuter avant de personnaliser le pilote. Des configurations de pilotes sont fournies pour les bases de données suivantes :

Base de données	1.6.1 ou nom de fichier antérieur	nom de fichier 1.6.2
Oracle 8i, 9i	JDBCOracleDirect.xml JDBCOracleIndirect.xml	JDBCOracle.xml
Microsoft SQL Server 7/2000	JDBCMSSQLDirect.xml JDBCMSSQLIndirect.xml	JDBCMSSQL.xml
IBM Universal Database (UDB) 7.2	JDBCDB2Direct.xml JDBCDB2Indirect.xml	JDBCDB2.xml
Sybase Adaptive Server Enterprise (ASE) 12.5	JDBCSybaseDirect.xml JDBCSybaseIndirect.xml	JDBCSybase.xml
MySQL 3.23	JDBCMySQLIndirect.xml	JDBCMySQL.xml
Informix Dynamic Server (IDS) 9.3	JDBCInformixDirect.xml JDBCInformixIndirect.xml	JDBCInformix.xml

Toutes les configurations utilisent les mêmes conventions, quelle que soit la base de données :

- Les champs de type chaîne comportent 64 caractères. Des champs de cette longueur peuvent contenir la plupart des attributs eDirectory. Vous pouvez adapter ces longueurs afin d'améliorer l'efficacité du stockage.

- Les champs de clé primaire comportent 8 chiffres.
- La colonne `record_id` de la table `eventlog` présente la précision numérique maximale autorisée par chaque base de données.
- Tous les noms de table, de déclencheur, de procédure stockée, d'index et de contrainte sont en minuscules. Cette convention de casse est la plus couramment utilisée.
- Les noms de déclencheurs sont précédés de `t_`, les noms de procédures stockées de `sp_`, les noms d'index de `i_`, les contraintes de vérification de `chk_`, les contraintes de clé primaire de `pk_` et les contraintes de clé étrangère de `fk_`.
- Les contraintes de vérification, de clé primaire et de clé étrangère respectent la convention de dénomination suivante : *préfixe_nom-table_nom-colonne* (par exemple, pk_emp_empno, fk_phone_empno, chk_eventlog_event_type)
- Les déclencheurs se conforment à la convention de dénomination suivante : t_*nom-table_opération* (par exemple, t_emp_insert)
- Les index se conforment à la convention de dénomination suivante : i_*nom_table_numéro* (par exemple, i_eventlog_1)
- Les colonnes d'identité et les objets de séquence peuvent placer 100 valeurs en mémoire cache.
- Les noms d'utilisateur se composent du nom de famille d'un utilisateur concaténé avec la valeur de clé primaire (par exemple, le nom d'utilisateur de Jean Dupont serait Dupont1).
- Les mots de passe de départ correspondent au nom de famille d'un utilisateur (par exemple, le mot de passe de Jean Dupont serait `Dupont`). Les mots de passe Sybase doivent comporter au moins 6 caractères. Lorsque les noms de famille sont plus courts, ils sont complétés par le caractère 'p' (par exemple, le mot de passe de Jean Bon serait `Bonppp`). Le caractère de remplissage peut être modifié dans la feuille de style Transformation de la commande du canal Abonné.

Importation de la configuration du pilote

Le fichier (XML) de configuration de pilote permet de créer et de configurer les objets nécessaires au bon fonctionnement du pilote. Il inclut également des exemples de règles et de feuilles de style que vous pouvez modifier pour faciliter l'implémentation de votre système.

1. Dans iManager, sélectionnez Gestion DirXML > Créer un pilote.
2. Sélectionnez un ensemble de pilotes

   Si vous placez ce pilote dans un nouvel ensemble de pilotes, vous devez spécifier un nom d'ensemble de pilotes, ainsi qu'un contexte et un serveur associé.
3. Cochez l'option Importer une configuration de pilote du serveur et sélectionnez le fichier .xml.

   Le fichier de configuration des pilotes est installé sur le serveur Web au moment de la configuration d'iManager.
4. Vous serez alors invité à entrer le nom du pilote. Entrez le nom du pilote puis cliquez sur Suivant pour continuer.
5. (Facultatif) Cliquez sur Définir les équivalences de sécurité.

   5a Cliquez sur Ajouter puis sélectionnez un objet disposant de droits Admin (ou de tout autre droit que vous voulez attribuer au pilote).

   5b Cliquez sur Appliquer, puis sur OK.

6 (Facultatif) Cliquez sur Exclure les rôles administratifs pour exclure de la réplication les objets correspondants.

6a Cliquez sur Ajouter puis sélectionnez les utilisateurs à exclure (l'utilisateur admin, par exemple).

6b Cliquez sur Appliquer, puis sur OK.

7 Cliquez sur Suivant pour afficher le résumé d'importation. Vérifiez que la configuration est correcte puis cliquez sur Terminer la présentation.

Les objets Identity Manager nécessaires ont été créés. Si vous n'avez pas défini les équivalences de sécurité ou exclu d'utilisateurs dotés de privilèges administratifs pendant l'importation, vous pouvez exécuter ces tâches en modifiant les propriétés de l'objet Pilote.

Installation d'objets de base de données

Les informations suivantes concernent l'installation et la configuration d'objets de base de données (tables, déclencheurs, index, etc.) pour la synchronisation avec la configuration de pilote par défaut.

Les scripts SQL se trouvent dans le répertoire tools\sql*base de données*.

Cette section contient des informations utiles pour :

- « Configuration des objets Oracle », page 29
- « Configuration des objets Microsoft SQL Server », page 30
- « Configuration des objets IBM DB2 », page 31
- « Configuration des objets Sybase », page 32
- « Configuration des objets MySQL », page 32
- « Configuration des objets Informix », page 32

Important : il est recommandé d'installer ou de désinstaller les pilotes préconfigurés et les scripts de base de données en bloc. Pour éviter tout problème de discordance, les scripts de base de données et les pilotes préconfigurés contiennent désormais un en-tête qui comprend un numéro de version, le nom de la base de données cible et la version de la base de données.

Pour des informations de désinstallation, reportez-vous au Chapitre 7, « Désinstallation des objets Pilote et Base de données », page 79.

Configuration des objets Oracle

1 À partir d'un client Oracle, comme SQL Plus, loguez-vous en tant qu'utilisateur **`SYSTEM`**. Par défaut, le mot de passe pour le compte `SYSTEM` est **`MANAGER`**.

2 Exécutez le premier script d'installation en vue d'une synchronisation directe ou indirecte. Exemple :

```
SQL> @c:\tools\sql\oracle\direct\INSTALL_DIRECT_1.sql
```

```
SQL> @c:\tools\sql\oracle\indirect\INSTALL_INDIRECT_1.sql
```

3 Loguez-vous en tant qu'utilisateur **`dirxml`** en utilisant **`dirxml`** comme mot de passe.

4 Exécutez le deuxième script d'installation en vue d'une synchronisation directe ou indirecte. Exemple :

```
SQL> @c:\tools\sql\oracle\direct\INSTALL_DIRECT_2.sql
```

```
SQL> @c:\tools\sql\oracle\indirect\INSTALL_INDIRECT_2.sql
```

Remarque : avant d'exécuter les essais de publication fournis en tant que `SYSTEM`, vous devez vous loguer et créer une nouvelle session. Sinon, vous ne serez pas en mesure de voir les objets de séquence qui sont la propriété de `dirxml`.

Si les scripts s'exécutent correctement, vous devez être averti de la création des objets de base de données. En cas d'erreurs, vérifiez que vous êtes logué sous le nom d'utilisateur correct. Avant de réexécuter les scripts d'installation, veillez à exécuter le script de désinstallation (par exemple, UNINSTALL_DIRECT.sql ou UNINSTALL_INDIRECT.sql).

Conseils de dépannage

- Lorsque vous générez des événements pour le canal Éditeur, assurez-vous que vous n'êtes pas connecté en tant qu'utilisateur `dirxml`. Si vous procédez à des modifications en tant qu'utilisateur `dirxml`, vos modifications ne seront pas publiées.
- Assurez-vous de bien valider vos modifications. Pour être publiée, une modification doit en effet être validée.

Configuration des objets Microsoft SQL Server

1. Démarrez l'Analyseur de requêtes.
2. Loguez-vous à votre serveur de base de données sous le nom d'utilisateur **`sa`**. Par défaut, le compte `sa` n'est protégé par aucun mot de passe.
3. Ouvrez et exécutez le premier script pour effectuer une synchronisation directe ou indirecte. Exemple :

   tools\sql\mssql\direct\INSTALL_DIRECT_1.sql

   tools\sql\mssql\indirect\INSTALL_INDIRECT_1.sql
4. Loguez-vous à votre serveur de base de données en tant qu'utilisateur **`dirxml`** en utilisant **`dirxml`** comme mot de passe.
5. Ouvrez et exécutez le deuxième script d'installation pour effectuer une synchronisation directe ou indirecte. Exemple :

   tools\sql\mssql\direct\INSTALL_DIRECT_2.sql

   tools\sql\mssql\indirect\INSTALL_INDIRECT_2.sql

Conseils de dépannage

- Lorsque vous générez des événements pour le canal Éditeur, assurez-vous que vous n'êtes pas connecté en tant qu'utilisateur `dirxml`. Si vous procédez à des modifications en tant qu'utilisateur `dirxml`, vos modifications ne seront pas publiées.
- Assurez-vous de bien valider vos modifications. Pour être publiée, une modification doit en effet être validée. Le mot clé de validation de Microsoft SQL Server est `go`.
- Avant d'exécuter les scripts, vérifiez que vous êtes connecté à la bonne base de données sous le bon nom d'utilisateur.

Configuration des objets IBM DB2

Pour DB2 Universal Database, vous devez créer manuellement un compte utilisateur de la base de données et une base de données avant d'exécuter les scripts fournis. Comme le processus de création des comptes utilisateur est différent d'un système d'exploitation à l'autre, l'étape 1 ci-après dépend du système d'exploitation. Ces instructions concernent un environnement d'exploitation Windows NT. Si vous réinstallez les objets de base de données, vous n'avez à répéter que les étapes 6 à 8.

1. Créez un compte pour l'utilisateur **`dirxml`** avec le mot de passe **`dirxml`** dans Gestionnaire des utilisateurs pour les domaines.
   - N'oubliez pas de désélectionner la case « L'utilisateur doit changer de mot de passe à la prochaine ouverture de session » pour ce compte.
   - Il peut être également utile de cocher la case Le mot de passe n'expire jamais.

   Remarque : les instructions suivantes sont indépendantes du système d'exploitation.
2. Démarrez le Control Center (Centre de contrôle).
3. Cliquez avec le bouton droit de la souris sur Databases (Bases de données) > cliquez sur Create Database Using Wizard (Création d'une base de données à l'aide de l'assistant).
4. Nommez la base de données **`dirxml`** > cliquez sur Finish (Terminer).
5. Copiez JDBCUtil.jar sur votre serveur DB2.
6. Démarrez le Command Center (Centre de commande) à partir du Control Center (Centre de contrôle).
7. Changez le nom du compte et le mot de passe administrateur de votre serveur avant d'exécuter le premier script d'installation.
8. Cliquez sur l'onglet Script > ouvrez le menu Script > importez et exécutez le script en vue d'une synchronisation directe ou indirecte. Exemple :

   tools\sql\db2\direct\INSTALL_DIRECT_1.sql

   tools\sql\db2\indirect\INSTALL_INDIRECT_1.sql
9. Importez le deuxième script d'installation en vue d'une synchronisation directe ou indirecte. Exemple :

   tools\sql\db2\direct\INSTALL_DIRECT_2.sql

   tools\sql\db2\indirect\INSTALL_INDIRECT_2.sql
10. Indiquez le chemin d'accès à JDBCUtil.jar et exécutez le script.

Conseils de dépannage

- Lorsque vous générez des événements pour le canal Éditeur, assurez-vous que vous n'êtes pas connecté en tant qu'utilisateur `dirxml`. Si vous procédez à des modifications en tant qu'utilisateur `dirxml`, vos modifications ne seront pas publiées.
- N'oubliez pas de valider vos modifications. Pour être publiée, une modification doit en effet être validée.

Configuration des objets Sybase

Cette section explique comment installer les objets de base de données Sybase Adaptive Server Enterprise (ASE).

Si vous n'avez pas installé la prise en charge JDBC sur votre serveur Sybase, vous devez le faire au préalable. Pour plus de détails, reportez-vous au manuel d'installation du serveur. Si l'installation est requise, vous devez exécuter le script sql_server*.sql pour installer la prise en charge de java.sql.DatabaseMetaData.

1. À partir d'un client Sybase, comme isql, loguez-vous en tant qu'utilisateur **`sa`** et exécutez le premier script d'installation en vue d'une synchronisation directe ou indirecte. Par défaut, le compte `sa` n'est protégé par aucun mot de passe. Exemple :

   isql -U sa -P -i tools\sql\sybase\direct\INSTALL_DIRECT_1.sql

   isql -U sa -P -i tools\sql\sybase\indirect\INSTALL_INDIRECT_1.sql

2. Loguez-vous en tant qu'utilisateur **`dirxml`** à l'aide de **`dirxml`** comme mot de passe et exécutez le deuxième script d'installation en vue d'une synchronisation directe ou indirecte. Exemple :

   isql -U dirxml -P dirxml -i tools\sql\sybase\direct\INSTALL_DIRECT_2.sql

   isql -U dirxml -P dirxml -i tools\sql\sybase\direct\INSTALL_INDIRECT_2.sql

Conseils de dépannage

- N'oubliez pas de valider vos modifications. Pour être publiée, une modification doit en effet être validée. Le mot clé de validation de Sybase est `go`.

Configuration des objets MySQL

1. À partir d'un client MySQL, comme mysql, loguez-vous en tant qu'utilisateur **`root`** ou tout autre utilisateur doté de privilèges d'administrateur. Par défaut, le compte `root` n'est protégé par aucun mot de passe.
2. Exécutez le premier script en vue d'une synchronisation indirecte. Exemple :

   mysql> \. c:\tools\sql\mysql\indirect\INSTALL_INDIRECT_1.sql

3. Ouvrez et exécutez le deuxième script d'installation pour effectuer une synchronisation indirecte. Exemple :

   mysql> \. c:\tools\sql\mysql\indirect\INSTALL_INDIRECT_2.sql

Configuration des objets Informix

Pour une installation Informix Dynamic Server, vous devez créer manuellement un compte utilisateur de base de données avant d'exécuter les scripts fournis.

Comme la procédure de création de comptes utilisateur diffère selon les systèmes d'exploitation, l'étape 1 ci-après dépend du système. Ces instructions concernent un environnement d'exploitation Windows NT. Si vous réinstallez les objets de base de données, vous n'avez à répéter que les étapes 2 à 6.

Instructions d'installation

1. Créez un compte pour l'utilisateur **dirxml** avec le mot de passe **dirxml** dans Gestionnaire des utilisateurs pour les domaines.
   - N'oubliez pas de désélectionner la case « L'utilisateur doit changer de mot de passe à la prochaine ouverture de session » pour ce compte.
   - Il peut être également utile de cocher la case Le mot de passe n'expire jamais.

   Remarque : les instructions suivantes sont indépendantes du système d'exploitation.
2. Démarrez l'Éditeur SQL.
3. Loguez-vous à votre serveur sous le nom d'utilisateur **informix**. Par défaut, le mot de passe pour le compte `informix` est **informix**.
4. Ouvrez et exécutez le premier script pour effectuer une synchronisation directe ou indirecte. Exemple :

   tools\sql\informix\direct\INSTALL_DIRECT_1.sql

   tools\sql\informix\indirect\INSTALL_INDIRECT_1.sql
5. Loguez-vous à votre serveur de base de données en tant qu'utilisateur **dirxml** en utilisant **dirxml** comme mot de passe.
6. Ouvrez et exécutez le deuxième script d'installation pour effectuer une synchronisation directe ou indirecte. Exemple :

   tools\sql\informix\direct\INSTALL_DIRECT_2.sql

   tools\sql\informix\indirect\INSTALL_INDIRECT_2.sql

Conseils de dépannage

- Lorsque vous générez des événements pour le canal Éditeur, assurez-vous que vous n'êtes pas connecté en tant qu'utilisateur `dirxml`. Si vous procédez à des modifications en tant qu'utilisateur `dirxml`, vos modifications ne seront pas publiées.
- Assurez-vous de bien valider vos modifications. Pour être publiée, une modification doit en effet être validée.
- Avant d'exécuter les scripts, vérifiez que vous êtes connecté à la bonne base de données sous le bon nom d'utilisateur.

Mise à niveau du pilote

Utilisez les informations et procédures suivantes si vous mettez le pilote à niveau à partir d'une version antérieure.

Conditions requises pour la mise à niveau

Pour les versions antérieures à la version 1.5, reportez-vous au DirXML Driver 1.5 for JDBC Implementation Guide (Guide d'implémentation du pilote DirXML 1.5 pour JDBC) (http://www.novell.com/documentation/french/dirxmldrivers/index.html). Assurez-vous d'utiliser l'outil d'association 1.6. Il remplace toutes les versions précédentes.

Mise à niveau de la version 1.5 à la version 1.6

Après avoir téléchargé l'image du CD, procédez comme suit pour une mise à niveau à partir d'une version antérieure du pilote :

1. Arrêtez les pilotes à mettre à niveau. Sélectionnez Manuel comme option de démarrage du pilote.
2. Arrêtez eDirectory.
3. Remplacez JDBCShim.jar, JDBCUtil.jar et CommonDriverShim.jar.
4. Redémarrez eDirectory.
5. (Facultatif) Installez les configurations de pilote.

   Il convient de désinstaller les pilotes préconfigurés précédents et d'exécuter les scripts de désinstallation de la base de données avant d'installer les nouveaux pilotes préconfigurés et les scripts.
6. Rétablissez les valeurs antérieures des options de démarrage des pilotes.
7. Redémarrez les pilotes.

Activation du pilote

L'activation doit être effectuée sous 90 jours après l'installation, sans quoi le pilote ne s'exécutera pas.

Pour obtenir des informations sur l'activation, reportez-vous à Activation des produits de gestion d'identité de Novell (http://www.novell.com/documentation/french/dirxml20/admin/data/afbx4oc.html).

4 Configuration du pilote

Cette section explique comment attribuer des valeurs possibles aux paramètres de configuration des pilotes. Avant de commencer, vous devez disposer des fichiers des pilotes appropriés et savoir utiliser Novell® eDirectory™ et iManager.

Définition des paramètres d'authentification du pilote

Une fois le pilote importé, vous devez fournir des informations d'authentification pour la base de données.

Configuration de l'authentification du pilote

1. Dans Novell iManager, cliquez sur Gestion DirXML > Présentation.
2. Recherchez l'ensemble qui contient le pilote puis cliquez sur l'icône du pilote.
3. Dans Présentation du pilote DirXML, cliquez sur l'objet Pilote. Les configurations de pilote s'affichent. Pour connaître les paramètres spécifiques à un pilote, reportez-vous au guide d'implémentation correspondant.
4. Entrez les informations d'authentification spécifiques au pilote choisi :

Nom du paramètre	Exemple de valeur de configuration	Valeur par défaut	Champ obligatoire
ID d'authentification	dirxml		oui
Contexte d'authentification	jdbc:oracle:thin:@255.255.255.255:1521:ora		oui
Mot de passe de l'application	dirxml		oui

ID d'authentification

L'ID d'authentification est le nom du compte utilisateur de base de données ou de login du pilote. Cet utilisateur doit exister et bénéficier de privilèges de login et de session sur la base de données, sinon aucune connexion ne pourra être établie. Par ailleurs, cet utilisateur doit disposer de droits de sélection, d'insertion, de mise à jour et de suppression sur les tables du schéma de synchronisation, sinon cette dernière échouera.

Contexte d'authentification

Le contexte d'authentification contient l'URL JDBC de la base de données cible.

Le format et le contenu de cette URL sont propriétaires et diffèrent selon les pilotes de fabricants tiers. Le contenu présente toutefois certaines ressemblances. En effet, chaque URL, quel que soit son format, comprend généralement une adresse IP ou un nom DNS, un numéro de port et un identificateur de base de données. Pour connaître la syntaxe exacte et le contenu requis pour votre pilote de fabricant tiers, consultez la documentation qui l'accompagne.

Le tableau suivant répertorie des exemples d'URL pour les pilotes de fabricants tiers. Il conviendra d'indiquer l'adresse IP, le numéro de port et les identificateurs de base ou de source de données appropriés pour votre base de données. Ces exemples utilisent tous l'adresse IP 255.255.255.255, le numéro de port par défaut de chaque base de données et `dirxml` comme identificateur de base ou de source de données.

Pilote de fabricant tiers	Exemple de syntaxe d'URL JDBC
Pilotes JDBC Oracle8i, 9i	jdbc:oracle:thin:@255.255.255.255:1521:dirxml
Pilote JDBC IBM DB2 UDB	jdbc:db2://255.255.255.255/dirxml
BEA Weblogic jDriver pour Microsoft SQL Server 7/2000	jdbc:weblogic:mssqlserver4:dirxml@255.255.255.255:1433
Pilote Microsoft SQL Server 2000 pour JDBC	jdbc:microsoft:sqlserver:// 255.255.255.255:1433;DatabaseName=dirxml
Sybase jConnect	jdbc:sybase:Tds:255.255.255.255:2048/dirxml
MySQL Connector/J	jdbc:mysql://255.255.255.255:3306/dirxml
Pilote JDBC Informix	jdbc:informix-sqli://255.255.255.255:1526/ dirxml:informixserver=server
Pilote de pont JDBC-ODBC de Sun	jdbc:odbc:dirxml

Mot de passe de l'application

Il s'agit du mot de passe du compte utilisateur de base de données ou de login qui est utilisé par le pilote. Vous devez créer un compte utilisateur ou de login sur la base de données et accorder des privilèges de login/session sur ce compte, sinon le pilote ne pourra pas se connecter.

Remarque : ConsoleOne® n'affiche plus les astérisques (*) dans les champs Nouveau mot de passe lorsque vous rouvrez la boîte de dialogue Propriétés du pilote. Pourtant, le mot de passe est bien conservé et ne doit pas être ressaisi.

Paramètres du pilote

Après avoir défini les paramètres d'authentification, vous devez définir les paramètres du pilote.

Ces paramètres se répartissent en trois catégories :

- Pilote
- Abonné
- Éditeur

Configuration du pilote

1. Dans iManager, cliquez sur Gestion DirXML > Présentation.
2. Recherchez l'ensemble qui contient le pilote puis cliquez sur l'icône du pilote.
3. Dans Présentation du pilote DirXML, cliquez sur l'objet Pilote. Les configurations de pilote s'affichent.

Le tableau suivant dresse la liste des paramètres avec des exemples de valeurs :

Nom du paramètre	Exemple de valeur de configuration	Valeur par défaut	Obligatoire	Balise
Nom de classe du pilote JDBC tiers	oracle.jdbc.driver.OracleDriver		oui	<jdbc-class>
Synchroniser le schéma	dirxml		oui	<sync-schema>
Synchroniser la ou les tables	emp		oui	<sync-tables>
Réutiliser les instructions ?	oui	oui	non	<reuse-statements>
Utiliser les transactions manuelles ?	oui	(déterminée dynamiquement)	non	<use-manual-transactions>
Utiliser la connexion simple ?	non	non	non	<use-single-connection>
Niveau d'isolation de transaction par défaut	lecture validée	(idem)	non	<transaction-isolation-level>
Nom de classe du testeur de connexion	com.novell.nds.dirxml.driver.jdbc.util.JDBCConnectionTester	(idem)	non	<connection-tester-class>
Instruction de test de la connexion	SELECT empno FROM dirxml.emp where -1 = 0		non	<connection-test-stmt>
Récupérer les métadonnées minimales ?	non	non	non	<minimal-metadata>
Gérer les résultats des instructions ?	oui	oui	non	<handle-stmt-results>
Chaîne d'initialisation de la connexion	USE dirxml		non	<connection-init>
Enable Referential Support (Activer la prise en charge des référentiels) ?	oui	oui	non	<enable-refs>

Nom de classe du pilote JDBC tiers

Le nom de classe du pilote JDBC tiers est un paramètre obligatoire avec distinction majuscules/minuscules. Ce nom fait référence au nom qualifié complet de la classe de votre pilote tiers. Le tableau suivant répertorie le nom de classe pour les pilotes de fabricants tiers testés.

Pilote de fabricant tiers	Valeur
Pilotes JDBC Oracle8i, 9i	oracle.jdbc.driver.OracleDriver
Pilote JDBC IBM DB2 UDB	COM.ibm.db2.jdbc.net.DB2Driver
BEA Weblogic jDriver pour MSSQL Server 7/2000	weblogic.jdbc.mssqlserver4.Driver
Pilote Microsoft SQL Server 2000 pour JDBC	com.microsoft.jdbc.sqlserver.SQLServerDriver
Sybase jConnect 5.5	com.sybase.jdbc2.jdbc.SybDriver
MySQL Connector/J	org.gjt.mm.mysql.Driver
Pilote JDBC Informix	com.informix.jdbc.IfxDriver
Pilote JDBC-ODBC Sun	sun.jdbc.odbc.JdbcOdbcDriver

Synchroniser le schéma

Synchroniser le schéma est un paramètre obligatoire avec distinction possible majuscules/minuscules. Ce paramètre identifie le schéma de base de données en cours de synchronisation. Un schéma de base de données est analogue au nom du propriétaire des tables en cours de synchronisation. Par exemple, si vous souhaitez synchroniser deux tables, `emp` et `phone`, qui appartiennent au même utilisateur de base de données `dirxml`, vous devez entrer `dirxml` dans ce champ. Lorsque ce paramètre est utilisé, le paramètre Synchroniser la ou les tables doit être laissé vide ou omis de la configuration d'un pilote.

Synchroniser la ou les tables

Synchroniser la ou les tables est un paramètre obligatoire avec distinction possible majuscules/minuscules. Ce paramètre permet de créer un schéma de base de données virtuel en dressant une liste des noms des classes de base de données logiques à synchroniser. Les noms de classes de base de données logiques sont les noms des tables et des vues parent. C'est une erreur de lister les noms de tables enfant. Ce paramètre est utile pour une synchronisation avec des bases de données qui ne prennent pas en charge la notion de schéma ou lorsque le schéma de synchronisation contient un grand nombre de tables dont quelques-unes seulement présentent un intérêt pour le pilote. Si vous synchronisez deux tables ou vues avec les mêmes noms dans différents schémas, assurez-vous de préfixer la table avec le schéma ou d'afficher les noms dans la règle d'assignation de schéma. Le pilote ne préfixe pas la table avec le schéma ou n'affiche pas les noms renvoyés par l'opération getSchema() par défaut. Lorsque ce paramètre est utilisé, le paramètre Synchroniser le schéma doit être laissé vide ou omis de la configuration d'un pilote.

Réutiliser les instructions ?

Réutiliser les instructions est un paramètre sans distinction majuscules/minuscules qui peut être obligatoire pour certains pilotes tiers. Si vous attribuez la valeur **oui** (valeur par défaut) à ce paramètre, le pilote alloue les objets java.sql.Statement, java.sql.PreparedStatement et java.sql.CallableStatement une seule fois, puis les réutilise. Lorsque ce paramètre a la valeur **non**, le pilote alloue ou libère les objets Instruction chaque fois qu'ils sont employés. La valeur **non** attribuée à ce paramètre entraîne un ralentissement des performances du pilote.

Il convient d'attribuer la valeur **non** à ce paramètre en cas d'utilisation avec le pilote Microsoft SQL Server 2000 pour JDBC.

Pour des performances maximales du pilote, il est recommandé d'utiliser la valeur par défaut ou d'omettre ce paramètre de la plupart des configurations de pilote.

Utiliser les transactions manuelles ?

Utiliser les transactions manuelles est un paramètre sans distinction majuscules/minuscules dont la valeur est tirée des métadonnées de la base de données en phase d'exécution. Ce paramètre ne doit être utilisé que lorsqu'il est nécessaire de remplacer le comportement par défaut du pilote. Par exemple, pour MySQL, la prise en charge des transactions est déterminée sur la base de la table plutôt que de la base de données. Dans ce cas, il est nécessaire de désactiver la prise en charge des transactions manuelles lors d'une synchronisation avec des tables sans prise en charge des transactions.

Lorsque ce paramètre a la valeur **oui**, le pilote prend en charge l'utilisation de transactions manuelles. Lorsqu'il a la valeur **non**, toute instruction exécutée par le pilote est une transaction automatique.

Pour assurer l'intégrité des données dans la base de données cible, nous vous recommandons d'omettre ce paramètre de la plupart des configurations de pilote.

Utiliser la connexion simple ?

Utiliser la connexion simple est un paramètre sans distinction majuscules/minuscules qui peut être obligatoire pour certains pilotes tiers. Lorsqu'il a la valeur **oui**, les canaux Abonné et Éditeur partagent une connexion unique. Lorsqu'il a la valeur **non**(par défaut), chaque canal emprunte une connexion séparée. La définition de ce paramètre sur **oui** réduit les performances du pilote. Ce paramètre ne doit être défini sur **oui** que lorsque les deux canaux Abonné et Éditeur sont utilisés.

Pour des performances maximales du pilote, il est recommandé d'utiliser la valeur par défaut ou d'omettre ce paramètre de la plupart des configurations de pilote.

Niveau d'isolation de transaction par défaut

Niveau d'isolation de transaction par défaut est un paramètre facultatif sans distinction majuscules/minuscules. Ce paramètre définit le niveau d'isolation de transaction par défaut pour les connexions utilisées par le pilote. Il existe cinq valeurs possibles, dont quatre correspondent aux constantes publiques définies dans l'interface java.sql.Connection :

- `aucune`
- `lecture non validée`
- `lecture validée`
- `lecture renouvelée`
- `sérialisable`

La valeur par défaut est `lecture validée`. Nous vous recommandons d'utiliser le niveau d'isolation de transaction `lecture validée`. Pour plus d'informations sur ces valeurs, reportez-vous au site Web de Sun (http://java.sun.com).

Étant donné que certains pilotes de fabricants tiers ne reconnaissent pas la valeur `aucune` du niveau d'isolation de transaction pour une connexion, le pilote accepte également la valeur supplémentaire `non prise en charge`.

Instruction de test de la connexion

L'instruction de test de la connexion est un paramètre facultatif avec distinction possible majuscules/minuscules. Ce paramètre est une solution de remplacement rapide à la création d'une classe de testeur de connexion. Pour détecter l'échec de la connexion, il suffit souvent d'envoyer une instruction SQL arbitraire via le réseau.

Lorsqu'il est présent, ce paramètre remplace le paramètre Nom de classe du testeur de connexion.

Nom de classe du testeur de connexion

Le nom de classe du testeur de connexion est un paramètre avec distinction majuscules/minuscules, qui peut être obligatoire pour certains pilotes tiers. Il s'agit du nom qualifié complet de la classe utilisée pour déterminer l'état de la connexion. Cette classe doit être publique, comporter un constructeur par défaut public et mettre en oeuvre l'interface com.novell.nds.dirxml.driver.jdbc.db.DBConnectionTester.

La valeur par défaut est `com.novell.nds.dirxml.driver.jdbc.util.JDBCConnectionTester`.

Pour le pilote Microsoft SQL pour JDBC, définissez la valeur suivante :
`com.novell.nds.dirxml.driver.jdbc.db.MSSQLConnectionTester`

Pour le pilote JDBC d'Informix, définissez la valeur :
`com.novell.nds.dirxml.driver.jdbc.db.InformixConnectionTester`

Pour le pilote Mysql Connector/J, définissez la valeur :
`com.novell.nds.dirxml.driver.jdbc.db.MySQLConnectionTester`

Ce paramètre est ignoré lorsqu'une valeur est définie pour le paramètre Instruction de test de la connexion.

Récupérer les métadonnées minimales ?

Récupérer les métadonnées minimales est un paramètre sans distinction majuscules/minuscules, qui peut être obligatoire pour certaines bases de données. Lorsqu'il a la valeur **`oui`**, le pilote n'appelle que des méthodes de métadonnées obligatoires. Lorsque la valeur **`non`** (par défaut) est définie pour ce paramètre, le pilote appelle les méthodes de métadonnées obligatoires et facultatives. Reportez-vous à Annexe D, « Méthodes java.sql.DatabaseMetaData », page 91 pour consulter une liste supplémentaire de méthodes de métadonnées obligatoires et facultatives. Les méthodes de métadonnées facultatives sont requises pour la synchronisation des attributs à valeurs multiples et référentiels.

Définissez cette valeur sur **`oui`** pour améliorer le temps de démarrage du pilote au détriment de son fonctionnement.

Gérer les résultats des instructions ?

Gérer les résultats des instructions est un paramètre facultatif sans distinction majuscules/minuscules. Ce paramètre indique au pilote le nombre d'ensembles de résultats qui peuvent être générés par une instruction SQL arbitraire. Trois valeurs sont possibles :

- `aucune`
- `unique`
- `multiple`

La valeur par défaut est `multiple`. Pour la compatibilité en amont, `oui` est égal à `multiple`non est égal à `aucune`.

Pour le pilote ODBC de Microsoft, Oracle ou Informix, vous devez attribuer la valeur `single` à ce paramètre. Pour les autres pilotes tiers, nous vous recommandons d'utiliser la valeur par défaut ou d'omettre ce paramètre de la plupart des configurations de pilote.

Chaîne d'initialisation de la connexion

La chaîne d'initialisation de la connexion est un paramètre facultatif avec distinction possible majuscules/minuscules. La chaîne d'initialisation de la connexion sert à définir des propriétés sur les connexions utilisées par le pilote. Vous devez séparer plusieurs valeurs d'instruction par des points-virgules. Ce paramètre permet d'ajuster les normes de compatibilité ANSI et le contexte de base de données.

Enable Referential Support (Activer la prise en charge des référentiels) ?

Enable referential support (Activer la prise en charge des référentiels) est un paramètre facultatif sans distinction majuscules/minuscules. Ce paramètre indique au pilote d'interpréter les contraintes de clé étrangère qui font référence aux tables parent d'autres classes de base de données en tant qu'attributs référentiels. Les attributs référentiels servent généralement à désigner un endiguement (par exemple, l'appartenance à un groupe). Lorsque la valeur **`oui`** (par défaut) est définie pour ce paramètre, le pilote interprète lesdites colonnes en tant que référentiels. Lorsque la valeur **`non`** est définie pour ce paramètre, le pilote interprète lesdites colonnes en tant que non-référentiels. L'objectif de ce paramètre est de garantir la compatibilité en amont avec la version 1.0 de ce pilote. Pour une compatibilité avec la version 1.0, ce paramètre doit avoir la valeur **`non`**.

Configuration de l'objet Abonné

Le tableau suivant dresse la liste des paramètres avec des exemples de valeurs.

Nom du paramètre	Exemple de valeur de configuration	Valeur par défaut	Obligatoire	Balise
Désactiver	oui	non	non	<disable>
Génération de clé primaire	emp("sp_empno(empno,fname)")		non	<key-gen>
Moment choisi pour la génération de la clé	après	avant	non	<key-gen-timing>
Vérifier le nombre de mises à jour ?	oui	oui	non	<check-update-count>

Désactiver

Désactiver est un paramètre facultatif sans distinction majuscules/minuscules. Lorsque ce paramètre a la valeur `oui`, le canal Abonné ne traite pas les événements, mais renvoie des avertissements à la place. Lorsque la valeur de ce paramètre est `non` (par défaut), le canal Abonné traite les événements.

Génération de clé primaire

La génération de clé primaire est un paramètre facultatif et complexe avec distinction possible majuscules/minuscules. Les identificateurs de base de données utilisés dans cette valeur ne doivent pas être délimités.

Lors du traitement d'événements d'`<ajout>`, le canal Abonné utilise des valeurs de clé primaire pour créer des associations. Ce paramètre indique comment le canal Abonné obtient les valeurs de clé primaire nécessaires pour générer des valeurs d'association. Il existe trois possibilités :

1. Les valeurs de clé primaire nécessaires sont déjà présentes dans l'événement XML.
2. Le canal Abonné doit générer les valeurs de clé primaire nécessaires.
3. Le canal Abonné doit se procurer ces valeurs en appelant une procédure stockée ou une fonction définie par l'utilisateur dans la base de données.

Méthode 1 : par défaut, le pilote suppose que les valeurs de clé primaire sont déjà présentes dans l'événement XML. Si c'est le cas, aucune valeur ne doit être générée. Cela est souhaitable lorsqu'un attribut eDirectory, comme un GUID, est explicitement assigné au schéma de la colonne de clé primaire d'une table ou d'une vue.

La syntaxe de la méthode 1 est la suivante : *nomclasse-basedonnées-logique*(none)

Exemple :

```
emp(none)

view_emp(none)
```

Méthode 2 : il est souvent souhaitable dans un environnement de test que le canal Abonné génère les valeurs de clé primaire avant qu'une procédure stockée ou une fonction soit disponible. Cette méthode peut également être utilisée avec des bases de données qui ne prennent pas en charge les procédures ou les fonctions stockées. Pour les colonnes de type numérique, le pilote utilise une fonction simple (MAX+1) pour générer des valeurs de clé primaire. Dans le cas de colonnes de type chaîne, le pilote génère une suite aléatoire de caractères alphanumériques. Les autres types de données ne sont pas pris en charge.

La syntaxe de la méthode 2 est la suivante : *nomclasse-basedonnées-logique*(driver)

Exemple :

```
emp(driver)

view_emp(driver)
```

Méthode 3 : les valeurs de clé primaire sont tirées d'une procédure stockée ou fonction définie par l'utilisateur.

La syntaxe des procédures stockées est la suivante : *nomclasse-basedonnées-logique (« signature-procédure-stockée »)*, où *signature-procédure-stockée = nom-procédure(nom-colonne, . . .).*

Exemple :

```
emp("sp_empno(empno, fname)")
```

```
view_emp("sp_empno(pk_empno, fname)")
```

La syntaxe des fonctions est la suivante : *nomclasse-basedonnées-logique("? = signature-fonction")*, où *signature-fonction = nom-fonction(nom-colonne, . . .)*.

Exemple :

```
emp("? = sp_empno(empno, fname)")
```

```
view_emp("? = sp_empno(pk_empno, fname)")
```

Cette notation assigne une table ou une vue parent à une procédure stockée ou une fonction définie par l'utilisateur. Les noms de colonne sont ceux de la classe de base de données logique qui doit être transmise à la procédure stockée ou fonction. L'ordre, le nombre et le type de données des paramètres doivent correspondre à l'ordre, au nombre et au type de données des paramètres attendus par la procédure ou fonction. Pour les procédures stockées, les colonnes de clé primaire doivent être transmises comme paramètres IN OUT. Les colonnes sans clé doivent être transmises comme paramètres IN.

Remarques supplémentaires sur la génération de la clé primaire

- Dans le cas de la première méthode, il convient d'assigner au schéma d'une colonne de clé primaire le GUID au lieu du CN.
- Avec la troisième méthode, les colonnes de clé primaire ne doivent pas être assignées au schéma, ni incluses dans les filtres Abonné ou Éditeur.
- Lors de la synchronisation de plusieurs classes, une méthode de génération de clé primaire doit être déclarée pour chaque classe de base de données logique. Les valeurs doivent être séparées par des espaces ou des virgules.

Moment choisi pour la génération de la clé

Le moment choisi pour la génération de la clé est un paramètre sans distinction majuscules/minuscules, qui est obligatoire pour la plupart des bases de données en cas d'utilisation des méthodes deux et trois de génération de la clé primaire.

Deux valeurs sont possibles :

- `avant`
- `après`

La valeur par défaut est `avant`.

Méthode 1 de génération de clé primaire : ce paramètre est ignoré.

Méthode 2 de génération de clé primaire : lorsque ce paramètre a la valeur **`avant`**, le pilote exécute une instruction select avant l'insertion d'une ligne dans une table ou vue parent. Lorsque ce paramètre a la valeur **`après`**, le pilote exécute une instruction select après l'insertion d'une ligne dans une table ou vue parent.

Méthode 3 de génération de clé primaire : avec la valeur **`avant`**, les procédures ou fonctions déclarées dans le paramètre Génération de clé primaire sont appelées avant l'insertion d'une ligne dans une table ou vue parent. Avec la valeur **`après`**, les procédures ou fonctions sont appelées après l'insertion d'une ligne dans une table ou vue parent.

Pour toutes les bases de données, sauf Oracle, la valeur `après` doit être attribuée à ce paramètre. Pour Oracle, il convient d'utiliser la valeur par défaut ou d'omettre ce paramètre.

Vérifier le nombre de mises à jour ?

Vérifier le nombre de mises à jour est un paramètre facultatif sans distinction majuscules/minuscules. Lorsque ce paramètre a la valeur `oui` (par défaut), un décompte des mises à jour est effectué afin de vérifier que lors de l'insertion, la mise à jour ou la suppression de lignes dans une table ou une vue, l'opération en question a bien été effectuée. Si ce paramètre a la valeur `oui` et que des lignes ne sont pas mises à jour, une erreur est émise. Lorsque ce paramètre a la valeur `non`, le nombre de mises à jour n'est pas vérifié. La valeur `non` doit être attribuée à ce paramètre lorsque des instructions sont redéfinies avant la logique de déclencheur dans une table ou à la place de cette logique dans une vue.

Lorsque vous effectuez une synchronisation avec Microsoft SQL Server, il est préférable d'utiliser la valeur par défaut car les erreurs contenues dans la logique de déclencheur (qui peuvent entraîner la restauration d'une transaction) ne sont pas toujours répercutées sur l'abonné.

Configuration de l'objet Éditeur

Le tableau suivant répertorie les paramètres de l'objet Éditeur, avec leurs valeurs par défaut et des exemples de configuration :

Nom du paramètre	Exemple de valeur de configuration	Valeur par défaut	Obligatoire	Balise
Désactiver	oui	non	non	<disable>
Nom de la table de consignation	eventlog		oui	<log-table>
Intervalle d'interrogation (en secondes)	1-604800 (1 semaine)	10	non	<polling-interval>
Intervalle de reconnexion (en secondes)	1-3600 (1 heure)	30	non	<reconnect-interval>
Optimiser les mises à jour	oui	non	non	<optimize-update>
Supprimer du journal	oui	oui	non	<delete-from-log>
Autoriser le retour en boucle ?	oui	non	non	<check-update-count>

Désactiver

Désactiver est un paramètre facultatif, sans distinction majuscules/minuscules, qui indique si le canal Éditeur doit ouvrir une connexion à la base de données et rechercher des événements de base de données dans la table de consignation des événements. Lorsque ce paramètre a la valeur `oui`, le canal Éditeur n'établit pas de connexion avec une base de données et n'interroge pas la table de consignation des événements. Lorsque ce paramètre a la valeur `non` (par défaut), le canal Éditeur se connecte à la base de données et interroge la table de consignation des événements.

Nom de la table de consignation

Le nom de la table de consignation est un paramètre obligatoire avec distinction possible majuscules/minuscules. Ce paramètre indique le nom de la table dans laquelle les événements de base de données sont stockés en vue de leur acheminement via le canal Éditeur. Cette valeur ne doit pas être délimitée.

Intervalle d'interrogation

L'intervalle d'interrogation est un paramètre facultatif, sans distinction majuscules/minuscules, qui indique la fréquence en secondes à laquelle l'objet Éditeur doit rechercher des événements dans la table de consignation des événements. La valeur par défaut est de 10 secondes.

Il est recommandé de ne pas définir de valeur inférieure à dix secondes.

Intervalle de reconnexion

L'intervalle de reconnexion est un paramètre facultatif, sans distinction majuscules/minuscules, qui indique la fréquence en secondes à laquelle l'objet Éditeur doit tenter de se reconnecter à la base de données cible. La valeur par défaut est de 30 secondes.

Il est recommandé de ne pas définir de valeur inférieure à dix secondes.

Optimiser les mises à jour

Ce paramètre facultatif, sans distinction majuscules/minuscules, indique si le canal Éditeur doit ou non ignorer les événements de type 2 qui contiennent les mêmes valeurs anciennes et nouvelles. L'égalité est déterminée par une opération de comparaison de chaînes avec distinction majuscules/minuscules. Les valeurs vides sont considérées comme égales. Cette option est utile si les déclencheurs du canal Éditeur ne sont pas optimisés. Lorsque cette option a la valeur `oui`, les événements de type 2 sont optimisés. Lorsque la valeur de ce paramètre est `non` (par défaut), les événements de type 2 ne sont pas optimisés.

Supprimer du journal

Ce paramètre facultatif, sans distinction majuscules/minuscules, indique si le canal Éditeur doit supprimer ou non les enregistrements traités de la table de consignation des événements. Lorsque ce paramètre a la valeur `non`, le canal Éditeur ne supprime pas les lignes traitées de la table. À la place, le canal Éditeur attribue au champ `état` la valeur 'S' (Réussi). Ce paramètre est utile à des fins de débogage. Lorsque la valeur de ce paramètre est `oui` (par défaut), les lignes traitées sont supprimées. Il s'agit du paramètre adapté pour un environnement de production. Les lignes dont le traitement échoue restent dans la table de consignation des événements quelle que soit la valeur de ce paramètre.

La valeur `non` ne doit être attribuée à ce paramètre qu'à des fins de débogage. La valeur `oui` de ce paramètre entraîne un ralentissement des performances d'acheminement via le canal Éditeur. Si une fonction d'audit est souhaitée dans un environnement de production, les lignes insérées dans la table de consignation des événements en vue de leur acheminement via le canal Éditeur doivent également être écrites dans une table miroir.

Niveaux de trace

Pour afficher les résultats de débogage de ce pilote, vous devez attribuer une valeur d'attribut DirXML-DriverTraceLevel comprise entre 1 et 6 à l'ensemble auquel appartient le pilote. Cet attribut est souvent confondu avec l'attribut DirXML-XSL TraceLevel. Pour plus d'informations sur les niveaux de trace du pilote, reportez-vous au Identity Manager Administration Guide (Guide d'administration d'Identity Manager) (http://www.novell.com/documentation).

Le pilote prend en charge les six niveaux de trace suivants :

1. Minimal
2. Propriétés de la base de données
3. État de la connexion, instructions SQL et enregistrements de la table de consignation
4. Verbeux
5. API JDBC (méthodes, arguments, valeurs renvoyées, etc.)
6. Pilote de fabricant tiers

Les niveaux 5 et 6 sont particulièrement utiles pour déboguer les pilotes de fabricants tiers.

Configuration de pilotes JDBC de fabricants tiers

Les instructions suivantes vous aideront à configurer des pilotes tiers. Pour des instructions de configuration particulières, reportez-vous à la documentation du pilote tiers.

- Utilisez la dernière version disponible du pilote.
- Lors de la configuration d'une source de données ODBC, veillez à ne remplacer aucun des paramètres d'authentification du pilote (par exemple, le nom d'utilisateur et le mot de passe).
- Il se peut que le comportement des pilotes tiers puisse être configuré. Dans de nombreux cas, il est possible de résoudre les problèmes d'incompatibilité en réglant les propriétés de configuration du pilote.
- Lors du traitement de caractères internationaux, il est souvent nécessaire d'indiquer explicitement aux pilotes tiers le codage de caractères utilisé par la base de données en ajoutant une chaîne de propriété à la fin de l'URL JDBC du pilote. Les propriétés se composent généralement d'un mot clé et d'une valeur de codage des caractères (par exemple, jdbc:odbc:mssql;charSet=Big5). Le mot clé de propriété peut varier selon les pilotes tiers.

  Les valeurs possibles de codage des caractères sont définies par Sun. Pour plus d'informations, reportez-vous à la page Supported Encoding du site Web de Sun (http://java.sun.com/products/jdk/1.1/docs/guide/intl/encoding.doc.html).

Le tableau suivant répertorie les paramètres recommandés pour une compatibilité maximale des pilotes. Ces paramètres sont utiles avec un pilote de fabricant tiers non testé.

Nom du paramètre	Valeur
Synchroniser la ou les tables	*table-list*
Réutiliser les instructions ?	non
Utiliser les transactions manuelles ?	non
Utiliser la connexion simple ?	oui
Niveau d'isolation de transaction par défaut	non pris en charge
Récupérer les métadonnées minimales ?	oui
Gérer les résultats des instructions ?	unique

5 Configuration avancée du pilote

Après avoir installé un exemple de script de préconfiguration et de base de données, vous devez personnaliser le pilote pour une utilisation spécialisée. Cette section présente des concepts importants, des exemples de configurations et d'autres informations qui vous aideront à configurer le pilote.

- « Assignation de schéma », page 47
- « Table de consignation des événements », page 58
- « Assignation d'événements », page 57
- « Utilisation du langage SQL dans des événements XML », page 65

Assignation de schéma

Le tableau suivant présente une vue générale du mode d'assignation d'objets eDirectory Novell à des objets de base de données par le pilote.

Objet eDirectory	Objet Base de données
Arborescence	Schéma
Classe	Table/Vue
Attribut	Colonne
Association	Clé primaire

Classes de base de données logiques

Par classe de base de données logique, on entend l'ensemble des tables ou des vues utilisées pour représenter une classe eDirectory dans une base de données. Une classe de base de données logique peut se composer d'une vue unique ou d'une table parent et de zéro ou plusieurs tables enfant. Le nom de la classe de base de données logique est le nom de la table ou vue parent.

Synchronisation indirecte

Dans un modèle de synchronisation indirecte, le pilote procède aux assignations suivantes :

Objet eDirectory	Objet Base de données
Classes	Tables
Attributs	Colonnes

Objet eDirectory	Objet Base de données
1 classe	1 table parent et 0 ou plus tables enfant
Attribut à valeur unique	Colonne de la table parent
Attribut à valeurs multiples	Colonne de la table parent ou Colonne de la table enfant (préférée)

Assignation de classes eDirectory à des classes de base de données logiques

Dans l'exemple suivant, la classe de base de données logique `emp` se compose d'une table parent `emp` et d'une table enfant `phone`. La classe logique `emp` est assignée à la classe eDirectory Utilisateur.

```
CREATE TABLE dirxml.emp
(
    empno NUMERIC(8) NOT NULL,
    fname VARCHAR(64),
    lname VARCHAR(64),
    pwdminlen NUMERIC(4),

    CONSTRAINT pk_emp_empno PRIMARY KEY(empno)
);

CREATE TABLE dirxml.phone
(
    empno NUMERIC(8) NOT NULL,
    phone VARCHAR(64) NOT NULL,

    CONSTRAINT fk_phone_empno FOREIGN KEY(empno) REFERENCES
    emp(empno)
);
```

```
<rule name="MappingRule">
     <attr-name-map>
           <class-name>
                 <nds-name>User</nds-name>
                 <app-name>emp</app-name>
           </class-name>
           <attr-name class-name="User">
                 <nds-name>Given Name</nds-name>
                  <app-name>fname</app-name>
           </attr-name>
           <attr-name class-name="User">
                 <nds-name>Surname</nds-name>
                  <app-name>lname</app-name>
           </attr-name>
          <attr-name class-name="User">
               <nds-name>Password Minimum Length</nds-name>
               <app-name>pwdminlen</app-name>
          </attr-name>
           <attr-name class-name="User">
                 <nds-name>Telephone Number</nds-name>
```

```
                    <app-name>phone.phoneno</app-name>
                </attr-name>
            </attr-name-map>
</rule>
```

Tables parent

Les tables parent sont des tables assorties d'une contrainte de clé primaire explicite qui contiennent une ou plusieurs colonnes. Dans une table parent, une contrainte de clé primaire explicite est requise pour indiquer au pilote les champs à inclure dans une valeur d'association.

```
CREATE TABLE dirxml.emp
(
    empno NUMERIC(8) NOT NULL,
    ...
    CONSTRAINT pk_emp_empno PRIMARY KEY(empno)
);
```

Le tableau suivant contient des exemples de données pour `dirxml.emp`.

empno	fname	lname
1	Jean	Untel

L'association résultante pour cette ligne serait la suivante :

```
empno=1,table=emp,schema=dirxml
```

Remarque : le sort des identificateurs de base de données dans les valeurs d'association est déterminé de manière dynamique lors de l'exécution, à partir des métadonnées de base de données.

Colonnes de la table parent

Les colonnes de table parent ne peuvent contenir qu'une valeur. À ce titre, elles conviennent parfaitement pour l'assignation d'attributs eDirectory à valeur unique. Ainsi, l'attribut eDirectory à valeur unique Password Minimum Length (longueur minimale du mot de passe) serait assigné à la colonne de table parent `pwdminlen`.

Les colonnes de table parent sont préfixées, de manière implicite, avec le nom de la table parent. Il n'est pas nécessaire de préfixer explicitement les colonnes de table parent. Ainsi, `emp.fname` est l'équivalent de `fname` à des fins d'assignation de schéma.

```
<rule name="MappingRule">
   <attr-name-map>
      <class-name>
         <nds-name>User</nds-name>
         <app-name>emp</app-name>
      </class-name>
      <attr-name class-name="User">
         <nds-name>Given Name</nds-name>
          <app-name>fname</app-name>
      </attr-name>
   </attr-name-map>
</rule>
```

Les données de type binaire et chaîne en grande quantité doivent en principe être assignées à des colonnes de table parent. Pour être assigné à une colonne de table enfant, un type de données doit pouvoir faire l'objet d'une comparaison dans une instruction SQL. Or, les types de données en grande quantité ne peuvent généralement pas être comparés dans des instructions SQL.

Les données de type binaire et chaîne en grande quantité peuvent être assignées à des colonnes de table enfant si les événements `<remove-value>` de ces types comprennent des éléments `<remove-all-values>` qui sont transformés dans les feuilles de style en éléments `<add-value>`, à raison d'un élément par valeur ajoutée.

Tables enfant

Une table enfant est une table qui comporte une contrainte de clé étrangère sur la clé primaire de sa table parent, ce qui relie les deux tables entre elles. Les colonnes qui composent la clé étrangère de la table enfant doivent porter le même nom que les colonnes contenues dans la clé primaire de la table parent. Ce nom de colonne commun est utilisé par le canal Éditeur pour identifier toutes les lignes de la table de consignation des événements qui se rapportent à une classe de base de données logique unique.

L'exemple suivant illustre la relation qui existe entre la table parent `emp` et sa table enfant `phone`. Notez que le même nom de colonne `empno` est utilisé dans les deux tables.

```
CREATE TABLE dirxml.emp
(
    empno NUMERIC(8) NOT NULL,
    ...
    CONSTRAINT pk_emp_empno PRIMARY KEY(empno)
);

CREATE TABLE dirxml.phone
(
    empno NUMERIC(8) NOT NULL,
    phoneno VARCHAR(64) NOT NULL,

    CONSTRAINT fk_phone_empno FOREIGN KEY(empno) REFERENCES
    emp(empno)
);
```

La colonne contrainte dans une table enfant identifie la table parent. Dans l'exemple ci-dessus, la colonne contrainte dans la table enfant `phone` est `empno`. Le seul objectif de cette colonne est de relier les tables `phone` et `emp`. Dans la mesure où les colonnes contraintes ne contiennent pas d'informations utiles, elles doivent être omises des déclencheurs de publication et de la règle d'assignation de schéma.

La colonne non-contrainte est celle qui nous intéresse. Elle représente un seul attribut à valeurs multiples. Dans l'exemple ci-dessus, la colonne non-contrainte est `phoneno`. Comme les colonnes non-contraintes peuvent contenir plusieurs valeurs, elles conviennent donc parfaitement à l'assignation d'attributs eDirectory à valeurs multiples. Par exemple, l'assignation de l'attribut eDirectory à valeurs multiples Telephone Number à `phone.phoneno`.

Toutes les colonnes d'une table enfant doivent être contraintes NOT NULL.

Remarque : chaque attribut eDirectory à valeurs multiples doit être assigné à une colonne de table enfant différente.

Le tableau suivant contient des exemples de données pour `dirxml.phone`.

empno	phoneno
1	111-1111
1	222-2222

En cas d'assignation d'un attribut eDirectory à valeurs multiples à une colonne de table enfant, le nom de cette dernière doit être précédé de manière explicite du nom de la table enfant (par exemple, `phone.phoneno`). Sinon, le pilote interprétera implicitement `phoneno` en tant que `emp.phoneno`, et pas `phone.phoneno`.

```
<rule name="MappingRule">
   <attr-name-map>
      <class-name>
         <nds-name>User</nds-name>
         <app-name>emp</app-name>
      </class-name>
      <attr-name class-name="User">
         <nds-name>Telephone Number</nds-name>
          <app-name>phone.phoneno</app-name>
      </attr-name>
   </attr-name-map>
</rule>
```

Attributs référentiels

L'endiguement référentiel peut être représenté dans la base de données grâce à l'utilisation de contraintes de clé étrangère. Les attributs référentiels sont des colonnes au sein d'une classe de base de données logique qui font référence aux colonnes de clé primaire des tables parent d'autres classes de base de données logiques.

Attributs référentiels à valeur unique

Deux tables parent peuvent être mises en relation grâce à une colonne unique de table parent. Cette colonne doit avoir une contrainte de clé étrangère qui pointe sur la clé primaire de l'autre table parent. L'exemple suivant met en relation une table parent unique `utilisateur` avec elle-même.

```
CREATE TABLE user
(
    idu NUMBER(8) NOT NULL,
    manager NUMBER(8),

    CONSTRAINT pk_user_idu PRIMARY KEY(idu),
    CONSTRAINT fk_user_idu FOREIGN KEY(manager)REFERENCES
    user(idu)
);
```

```
<rule name="Mapping Rule">
    <attr-name-map>
        <class-name>
            <nds-name>User</nds-name>
            <app-name>user</app-name>
        </class-name>
        <attr-name class-name="User">
            <nds-name>manager</nds-name>
```

```
            <app-name>manager</app-name>
        </attr-name>
    </attr-name-map>
</rule>
```

Les colonnes référentielles à valeur unique doivent pouvoir être égales à NULL.

Attributs référentiels à valeurs multiples

Deux tables parent peuvent être mises en relation grâce à une table enfant commune. Cette table enfant doit avoir une contrainte de clé étrangère qui pointe sur la clé primaire de chaque table parent. L'exemple suivant met en relation deux tables parent `utilisateur` et `groupe` par l'intermédiaire d'une table enfant commune `membre`.

```
CREATE TABLE user
(
    idu NUMBER(8) NOT NULL,
    lname VARCHAR(64) NOT NULL,

    CONSTRAINT pk_user_idu PRIMARY KEY(idu)
);

CREATE TABLE group
(
    idg NUMBER(8) NOT NULL,

    CONSTRAINT pk_group_idg PRIMARY KEY(idg)
);

CREATE TABLE member
(
    idg NUMBER(8)NOT NULL,
    idu NUMBER(8)NOT NULL,

    CONSTRAINT fk_member_idg FOREIGN KEY(idg) REFERENCES
    group(idg),
    CONSTRAINT fk_member_idu FOREIGN KEY(idu) REFERENCES
    user(idu)
);

<rule name="Mapping Rule">
    <attr-name-map>
        <class-name>
            <nds-name>User</nds-name>
            <app-name>user</app-name>
        </class-name>
        <attr-name class-name="User">
            <nds-name>Surname</nds-name>
            <app-name>lname</app-name>
        </attr-name>
        <class-name>
            <nds-name>Group</nds-name>
            <app-name>group</app-name>
        </class-name>
        <attr-name class-name="Group">
            <nds-name>Member</nds-name>
            <app-name>member.idu</app-name>
        </attr-name>
    </attr-name-map>
</rule>
```

La première colonne contrainte dans une table enfant détermine la propriété. Dans l'exemple ci-dessus, membre est considéré comme faisant partie de la classe groupe. membre est considéré comme étant un enfant propre de groupe. La deuxième colonne contrainte dans une table enfant est l'attribut référentiel à valeurs multiples. Les deux colonnes doivent être contraintes NOT NULL.

Dans l'exemple suivant, l'ordre des colonnes contraintes a été inversé pour que membre soit une partie d'utilisateur. Pour mieux refléter cette relation, membre a été renommée membre_de.

```
CREATE TABLE user
(
    idu NUMBER(8) NOT NULL,
    lname VARCHAR(64) NOT NULL,

    CONSTRAINT pk_user_idu PRIMARY KEY(idu)
);

CREATE TABLE group
(
    idg NUMBER(8) NOT NULL,

    CONSTRAINT pk_group_idg PRIMARY KEY(idg)
);

CREATE TABLE member_of
(
    idu NUMBER(8)NOT NULL,
    idg NUMBER(8)NOT NULL,

    CONSTRAINT fk_member_idg FOREIGN KEY(idg) REFERENCES
    group(idg),
    CONSTRAINT fk_member_idu FOREIGN KEY(idu) REFERENCES
    user(idu)
);
```

```
<rule name="Mapping Rule">
    <attr-name-map>
        <class-name>
            <nds-name>User</nds-name>
            <app-name>user</app-name>
        </class-name>
        <attr-name class-name="User">
            <nds-name>Surname</nds-name>
            <app-name>lname</app-name>
        </attr-name>
        <attr-name class-name="User">
            <nds-name>Group Membership</nds-name>
            <app-name>member_of.idg</app-name>
        </attr-name>
        <class-name>
            <nds-name>Group</nds-name>
            <app-name>group</app-name>
        </class-name>
    </attr-name-map>
</rule>
```

Dans les bases de données dans lesquelles la position n'a aucune importance, l'ordre est déterminé par comparaison lexicographique.

En général, il est seulement nécessaire de synchroniser les attributs référentiels à valeurs multiples dans l'une ou l'autre classe, pas dans les deux. Pour synchroniser les attributs référentiels des deux classes, il faudrait construire deux tables enfant, une par classe. Si par exemple vous souhaitiez synchroniser Adhésion au groupe et Membre, vous auriez besoin de deux tables enfant : `membre_de` et `membre`.

En pratique, pour la synchronisation des objets Utilisateur et Groupe, nous vous recommandons de synchroniser l'attribut Adhésion au groupe de la classe Utilisateur au lieu de l'attribut Membre de la classe Groupe. Lorsque vous synchronisez Membre, des événements sont générés pour les utilisateurs non associés qui sont ajoutés à des groupes associés. Lorsque vous synchronisez Adhésion au groupe, des événements ne sont générés que pour les utilisateurs associés qui sont ajoutés à des groupes associés.

Synchronisation directe

Dans un modèle de synchronisation directe, le pilote procède aux assignations suivantes :

Objet eDirectory	Objet Base de données
Classes	Vues
Attributs	Afficher les colonnes
1 classe	Vue
Attribut à valeur unique	Afficher la colonne
Attribut à valeurs multiples	Afficher la colonne

Une vue est une table logique. Contrairement aux tables parent ou enfant, elle n'existe pas physiquement dans la base de données. En tant que telles, les vues ne peuvent pas comporter de contraintes de clé primaire/clé étrangère. Pour indiquer au pilote les champs à utiliser lors de la génération de valeurs d'association, une ou plusieurs colonnes de vue doivent porter le préfixe `pk_` (sans distinction majuscules/minuscules).

Remarque : les vues doivent être construites de telle sorte que les colonnes dotées du préfixe `pk_` identifient une ligne de façon unique.

Les fonctionnalités de mise à jour des vues sont très variables selon les bases de données. La plupart des bases permettent une mise à jour des vues dans certaines conditions. Si les vues sont strictement en lecture seule, il est impossible de les utiliser pour l'acheminement des données via le canal Abonné. Sous Microsoft SQL Server 2000 et Oracle 8i et 9i, il est possible de définir une logique de mise à jour sur les vues dans instead-of-triggers, ce qui permet à une vue de joindre plusieurs tables tout en restant modifiable.

```
CREATE TABLE dirxml.emp
(
    empno NUMERIC(8) NOT NULL UNIQUE,
    fname VARCHAR(64),
    lname VARCHAR(64),
    pwdminlen NUMERIC(4),
    phoneno VARCHAR(64)
);

CREATE VIEW dirxml.view_emp
(pk_empno, fname, lname, pwdminlen, phoneno)
AS
SELECT empno, fname, lname, pwdminlen, phoneno FROM dirxml.emp;
```

```
<rule name="MappingRule">
     <attr-name-map>
           <class-name>
              <nds-name>User</nds-name>
              <app-name>view_emp</app-name>
          </class-name>
          <attr-name class-name="User">
              <nds-name>Given Name</nds-name>
              <app-name>fname</app-name>
          </attr-name>
          <attr-name class-name="User">
              <nds-name>Surname</nds-name>
              <app-name>lname</app-name>
          </attr-name>
          <attr-name class-name="User">
              <nds-name>Password Minimum Length</nds-name>
              <app-name>pwdminlen</app-name>
          </attr-name>
          <attr-name class-name="User">
              <nds-name>Telephone Number</nds-name>
              <app-name>phoneno</app-name>
          </attr-name>
     </attr-name-map>
</rule>
```

Synchronisation des colonnes de clé primaire

Quand la base de données est la source experte de colonnes de clé primaire, celles-ci doivent généralement être omises des filtres Éditeur et Abonné, de la règle d'assignation de schéma et des déclencheurs de publication.

Quand eDirectory est la source experte de colonnes de clé primaire, celles-ci doivent généralement être incluses dans le filtre Abonné et la règle d'assignation de schéma et omises du filtre Éditeur et des déclencheurs de publication. Il est également recommandé d'utiliser GUID plutôt que CN comme clé primaire. CN est à valeurs multiples et peut être modifié. GUID est à valeur unique et statique.

Synchronisation de plusieurs classes

Lors de la synchronisation de plusieurs classes eDirectory, il est nécessaire de synchroniser chaque classe avec une table ou une vue parent différente. Chaque classe de base de données logique doit avoir un nom de colonne de clé primaire unique. Ce nom de colonne commun est utilisé par le canal Éditeur pour identifier toutes les lignes de la table de consignation des événements qui se rapportent à une classe de base de données logique unique. Ainsi, les classes de base de données logique `utilisateur` et `groupe` ont chacune un nom de colonne de clé primaire unique.

```
CREATE TABLE user
(
    idu NUMBER(8) NOT NULL,
    lname VARCHAR(64) NOT NULL,

    CONSTRAINT pk_user_idu PRIMARY KEY(idu)
);

CREATE TABLE group
(
    idg NUMBER(8) NOT NULL,

    CONSTRAINT pk_group_idg PRIMARY KEY(idg)
);
```

Assignation d'attributs à valeurs multiples à des champs de base de données à valeur unique

Par défaut, le pilote suppose que tous les attributs eDirectory assignés aux colonnes d'une table parent ou d'une vue sont à valeur unique. Comme le pilote ne reconnaît pas le schéma eDirectory, il n'a aucun moyen de savoir si un attribut eDirectory est à valeur unique ou à valeurs multiples. Par conséquent, les assignations d'attributs à valeur unique et à valeurs multiples sont traitées de la même manière.

Le pilote met en oeuvre l'algorithme MRT (Most Recently Touched - Modifié en dernier) vis-à-vis des colonnes de table parent ou de vue à valeur unique. Un algorithme MRT garantit le stockage dans la base de données de la valeur d'attribut ajoutée ou supprimée en dernier. Cet algorithme fonctionne si l'attribut en question est à valeur unique et a des effets indésirables si l'attribut comporte plusieurs valeurs.

Lorsqu'une valeur est supprimée d'un attribut à valeurs multiples, le champ de base de données auquel elle est assignée sera défini comme vide (NULL) et le restera jusqu'à l'ajout d'une nouvelle valeur. Plusieurs solutions à ce comportement indésirable sont exposées ci-après.

- La solution préférée consiste à élargir le schéma eDirectory afin que seuls des attributs à valeur unique soient assignés aux colonnes de table parent ou de vue.
- Pour une synchronisation indirecte, assignez chaque attribut à valeurs multiples à sa propre table enfant.
- Pour la synchronisation directe ou indirecte, utilisez des feuilles de style pour déterminer plusieurs valeurs avant de les insérer dans une colonne de table parent ou de vue.

- Mettez en oeuvre une règle de première ou dernière valeur par réplique dans les feuilles de style, à l'aide des méthodes fournies dans la classe com.novell.nds.dirxml.driver.jdbc .util.MappingPolicy. Avec une règle de première valeur par réplique, la première valeur d'attribut de la réplique Identity Manager est toujours synchronisée. Avec une règle de dernière valeur par réplique, la dernière valeur d'attribut d'une réplique est toujours synchronisée. Tous les pilotes préconfigurés présentent une règle de première valeur par réplique. Ils assignent les attributs eDirectory à valeurs multiples Given name (Prénom), Surname (Nom) et Facsimile Telephone Number (Numéro de télécopie) aux colonnes à valeur unique `fname`, `lname` et `faxno` respectivement.

Assignation d'événements

Le tableau suivant est un résumé de la manière dont l'Abonné assigne les événements XML aux instructions SQL.

Événement XML	Équivalent SQL
<add>	1 ou plusieurs instructions d'insertion ; 0 ou 1 instruction de sélection ; 0 ou 1 procédure stockée ou appel de fonction
<modify>	0 ou 1 instruction de mise à jour ; 0 ou plus instructions d'insertion ; 0 ou plus instructions de suppression
<delete>	1 ou plusieurs instructions de suppression : 0 ou plus instructions de mise à jour
<requête>	1 ou plusieurs instructions de sélection
<move> ou <rename>	0 instruction

Événements d'ajout

Les événements d'ajout sont assignés à une instruction d'insertion pour la table ou la vue parent et zéro ou plusieurs instructions d'insertion pour chaque table enfant. Pour la méthode 2 de génération de clé primaire, une instruction de sélection est exécutée. Pour la méthode 3 de génération de clé primaire, une procédure stockée ou un appel de fonction est exécuté(e).

Événements de modification

Les événements de modification sont assignés à zéro ou une instruction de mise à jour pour la table ou la vue parent et zéro ou plusieurs instructions d'insertion et de suppression pour chaque table enfant.

Événements de suppression

Les événements de suppression sont assignés à une instruction de suppression pour la table ou la vue parent et zéro ou une instruction de mise à jour pour chaque colonne de table parent référentielle à valeur unique.

Les événements de suppression sont assignés à zéro ou plusieurs instructions de suppression pour chaque colonne de table enfant référentielle à valeurs multiples.

Événements d’interrogation

Les événements d’interrogation sont assignés à une instruction de sélection pour la table ou la vue parent et zéro ou une instruction de sélection pour chaque table enfant.

Événements de déplacement et de réassignation de nom

Les événements de déplacement et de réassignation de nom sont NOOP (« No Operation »). Ils ne sont jamais assignés à des instructions.

Table de consignation des événements

C’est dans la table de consignation des événements que les événements du canal Éditeur sont stockés. Cette section décrit la structure et les limites de cette table.

Vous pouvez personnaliser le nom de la table de consignation des événements et ses colonnes, afin d’éviter des conflits avec les mots de base de données réservés. L’ordre, le nombre et les types de données de ses colonnes doivent cependant rester constants. Dans les bases de données dans lesquelles la position n’a aucune importance, l’ordre est déterminé par comparaison lexicographique.

Colonnes de la table de consignation des événements

- `record_id`

  La colonne `record_id` sert à identifier de façon unique les lignes de la table de consignation des événements. Cette colonne doit contenir des valeurs entières uniques séquentielles, croissantes et positives.

- `status`

  La colonne `status` indique l’état d’une ligne donnée. Ses valeurs possibles sont les suivantes :

  - 'N' = nouveau
  - 'U' = inconnu
  - 'S' = réussi
  - 'W' = avertissement
  - 'F' = fatal
  - 'E' = erreur

  Toutes les lignes insérées dans la table de consignation des événements doivent avoir la valeur `status` 'N' pour être traitées. Les autres caractères d’état sont utilisés uniquement par le canal Éditeur. Tous les autres caractères sont réservés à un usage ultérieur.

  Remarque : ces valeurs d’état respectent la casse.

- `event_type`

  Les valeurs de cette colonne doivent être comprises entre 1 et 8. Les types d'événements appartiennent à deux catégories principales : par champ (1 à 3, 7 et 8) et par ligne (4 à 6). Les événements par champ présentent une plus grande granularité que les événements par ligne, mais prennent davantage de place dans la table de consignation. Les événements par ligne présentent une granularité plus fine et exigent moins d'espace. Les types d'événements par champ peuvent être considérés comme par attribut. Les types d'événements par ligne peuvent être considérés comme par objet.

  Les types d'événements peuvent également être regroupés en deux autres catégories : avec retour d'interrogation (5 à 8) et sans retour d'interrogation (1 à 4). Les événements avec retour d'interrogation sont utiles lors de la synchronisation de données de type binaire et chaîne en grande quantité.

  En général, une combinaison de types d'événements de chaque catégorie permet d'obtenir les meilleurs résultats en termes de temps, d'espace et de complexité.

  Les valeurs suivantes permettent une classification des types d'événements. Tous les autres nombres sont réservés à un usage ultérieur.

  - 1 = insertion de champ
  - 2 = mise à jour de champ
  - 3 = mise à jour de champ (avec suppression de toutes les valeurs)
  - 4 = suppression de ligne
  - 5 = insertion de ligne (avec retour d'interrogation)
  - 6 = mise à jour de ligne (avec retour d'interrogation)
  - 7 = insertion de champ (avec retour d'interrogation)
  - 8 = mise à jour de champ (avec retour d'interrogation)

- `event_time`

  Réservé à un usage ultérieur. Cette valeur ne doit pas être vide.

- `perpetrator`

  L'utilisateur qui est à l'origine de l'événement. Une valeur vide est interprétée comme désignant un autre utilisateur que l'utilisateur du pilote. À ce titre, les enregistrements avec `perpetrator` = NULL ou !*nomutilisateur du pilote* sont publiés. Les enregistrements avec `perpetrator` = *nomutilisateur du pilote* ne sont pas publiés, sauf si le paramètre du canal Éditeur Autoriser le retour en boucle ? a la valeur oui.

- `table_name`

  Le nom de la table ou de la vue dans laquelle l'événement s'est produit.

- `table_key`

  Les valeurs de cette colonne doivent être formatées exactement de la même façon dans tous les déclencheurs d'une classe de base de données logique. Par exemple,

  nom de colonne de clé primaire = valeur + nom de colonne de clé primaire = valeur . . .

  - Pour des pilotes préconfigurés indirects, par exemple, la valeur de cette colonne serait `empno=1`.
  - Pour des pilotes préconfigurés directs, par exemple, la valeur de cette colonne serait `pk_empno=1`.

Remarque : les valeurs de clé primaire placées dans le champ `table_key` doivent être délimitées (c'est-à-dire, mises entre guillemets) si elles contiennent les caractères suivants : , ; ' + = \ " < >

Des différences de remplissage ou de formatage peuvent entraîner un traitement des événements dans le désordre. Dans un souci de performance, il convient de supprimer tout espace inutile dans les valeurs numériques. (Par exemple, « empno=1 » est préférable à « empno= 1 ».)

- `column_name`

  Le nom de la colonne qui a été modifiée. Cette colonne n'est utilisée que par les types d'événements par champ (1 à 3, 7 et 8). Bien que cette colonne soit utilisée uniquement pour les types d'événements par champ, elle doit toujours figurer dans la table de consignation des événements. Dans le cas contraire, l'objet Éditeur provoquera l'arrêt du pilote.

- `old_value`

  Le nom de colonne de l'ancienne valeur n'est utilisé que par les types d'événements par champ, sans retour d'interrogation (1 à 3). Bien que cette colonne soit utilisée uniquement pour ces types d'événements, elle doit toujours figurer dans la table de consignation des événements. Dans le cas contraire, l'objet Éditeur provoquera l'arrêt du pilote.

- `new_value`

  La nouvelle valeur du champ. Le nom de colonne de la nouvelle valeur n'est utilisé que par les types d'événements par champ, sans retour d'interrogation (1 à 3). Bien que cette colonne soit utilisée uniquement pour ces types d'événements, elle doit toujours figurer dans la table de consignation des événements. Dans le cas contraire, l'objet Éditeur provoquera l'arrêt du pilote.

Types d'événement

Cette section décrit de façon plus détaillée les différents types d'événements et leur interprétation par l'objet Éditeur.

Le tableau ci-dessous présente la corrélation de base entre les types d'événements du canal Éditeur et le code XML généré par l'objet Éditeur.

Type d'événement	XML résultant
insert	<add>
update	<modify>
delete	<delete>

L'exemple ci-dessous illustre le code XML généré par l'objet Éditeur pour chaque type possible des événements consignés dans la table `emp`.

```
CREATE TABLE dirxml.emp
(
    empno NUMERIC(8) NOT NULL,
    fname VARCHAR2(64),
    photo LONGRAW,

    CONSTRAINT pk_emp_empno PRIMARY KEY(empno)
);
```

Le tableau ci-dessous montre le contenu initial de `emp` après l'insertion d'une nouvelle ligne :

empno	fname	lname	photo
1	Jacques	Frost	0xAAAA

Le tableau ci-dessous montre le contenu actuel de `emp` après la mise à jour de la ligne :

empno	fname	lname	photo
1	Jean	Untel	0xBBBB

1. Insertion de champ

   Le tableau ci-dessous montre le contenu de la table de consignation des événements après l'insertion d'une nouvelle ligne dans la table `emp`. La valeur de la colonne `photo` a été chiffrée au format Base64. L'équivalent codé au format Base64 de 0xAAAA est qqo=.

event_type	table	table_key	column_name	old_value	new_value
1	emp	empno=1	fname	NULL	Jacques
1	emp	empno=1	lname	NULL	Frost
1	emp	empno=1	photo	NULL	qqo=

   Le XML généré par l'Éditeur serait :

```
<add class-name="emp">
    <association>empno=1,table=emp,schema=dirxml
    </association>
    <add-attr attr-name="fname">
        <value type="string">Jack</value>
    </add-attr>
    <add-attr attr-name="lname">
        <value type="string">Frost</value>
    </add-attr>
    <add-attr attr-name="photo">
        <value type="octet">qqo=</value>
    </add-attr>
</add>
```

2. Mise à jour de champ

   Le tableau ci-dessous montre le contenu de la table de consignation des événements après la mise à jour de la ligne dans la table `emp` : Les valeurs de la colonne `photo` ont été chiffrées au format Base64. L'équivalent codé au format Base64 de 0xBBBB est u7s=.

event_type	table	table_key	column_name	old_value	new_value
2	emp	empno=1	fname	Jacques	Jean
2	emp	empno=1	lname	Frost	Untel
2	emp	empno=1	photo	qqo=	u7s=

Le XML généré par l'Éditeur serait :

```
<modify class-name="emp">
    <association>empno=1,table=emp,schema=dirxml
    </association>
    <modify-attr attr-name="fname">
        <remove-value>
            <value type="string">Jack</value>
        </remove-value>
        <add-value>
            <value type="string">John</value>
        </add-value>
    </modify-attr>
    <modify-attr attr-name="lname">
        <remove-value>
            <value type="string">Frost</value>
        </remove-value>
        <add-value>
            <value type="string">Doe</value>
        </add-value>
    </modify-attr>
    <modify-attr attr-name="photo">
        <remove-value>
            <value type="octet">qqo=</value>
        </remove-value>
        <add-value>
            <value type="octet">u7s=</value>
        </add-value>
    </modify-attr>
</modify>
```

3. Mise à jour de champ (avec suppression de toutes les valeurs)

   Le tableau ci-dessous montre le contenu de la table de consignation des événements après la mise à jour de la ligne dans la table `emp` : La valeur de la colonne `photo` a été chiffrée au format Base64.

event_type	table	table_key	column_name	old_value	new_value
3	emp	empno=1	fname	Jacques	Jean
3	emp	empno=1	lname	Frost	Untel
3	emp	empno=1	photo	qqo=	u7s=

Le XML généré par l'Éditeur serait :

```
<modify class-name="emp">
    <association>empno=1,table=emp,schema=dirxml
    </association>
    <modify-attr attr-name="fname">
        <remove-all-values/>
        <add-value>
            <value type="string">John</value>
        </add-value>
    </modify-attr>
    <modify-attr attr-name="lname">
        <remove-all-values/>
        <add-value>
            <value type="string">Doe</value>
```

```
        </add-value>
    </modify-attr>
    <modify-attr attr-name="photo">
        <remove-all-values/>
        <add-value>
            <value type="octet">u7s=</value>
        </add-value>
    </modify-attr>
</modify>
```

4. Suppression de ligne

   Le tableau ci-dessous montre le contenu de la table de consignation des événements après la suppression de la ligne dans la table `emp` :

event_type	table	table_key	column_name	old_value	new_value
4	emp	empno=1	NULL	NULL	NULL

   Le XML généré par l'Éditeur serait :

```
<delete class-name="emp">
    <association>empno=1,table=emp,schema=dirxml
    </association>
</delete>
```

5. Insertion de ligne (avec retour d'interrogation)

   Le tableau ci-dessous montre le contenu de la table de consignation des événements après l'insertion d'une nouvelle ligne dans la table `emp`.

event_type	table	table_key	column_name	old_value	new_value
5	emp	empno=1	NULL	NULL	NULL

   Le XML généré par l'Éditeur figure ci-dessous. Notez que les valeurs reflètent le contenu actuel de la table `emp` et pas son contenu initial.

```
<add class-name="emp">
    <association>empno=1,table=emp,schema=dirxml
    </association>
    <add-attr attr-name="fname">
        <value type="string">John</value>
    </add-attr>
    <add-attr attr-name="lname">
        <value type="string">Doe</value>
    </add-attr>
    <add-attr attr-name="photo">
        <value type="octet">u7s=</value>
    </add-attr>
</add>
```

6. Mise à jour de ligne (avec retour d'interrogation)

   Le tableau ci-dessous montre le contenu de la table de consignation des événements après la mise à jour de la ligne dans la table `emp` :

event_type	table	table_key	column_name	old_value	new_value
6	emp	empno=1	NULL	NULL	NULL

Le XML généré par l'Éditeur figure ci-dessous. Notez que les valeurs reflètent le contenu actuel de la table `emp` et pas son contenu initial.

```
<modify class-name="emp">
    <association>empno=1,table=emp,schema=dirxml
    </association>
    <modify-attr attr-name="fname">
        <remove-all-values/>
        <add-value>
            <value type="string">John</value>
        </add-value>
    </modify-attr>
    <modify-attr attr-name="lname">
        <remove-all-values/>
        <add-value>
            <value type="string">Doe</value>
        </add-value>
    </modify-attr>
    <modify-attr attr-name="photo">
        <remove-all-values/>
        <add-value>
            <value type="octet">u7s=</value>
        </add-value>
    </modify-attr>
</modify>
```

7. Insertion de champ (avec retour d'interrogation)

   Le tableau ci-dessous montre le contenu de la table de consignation des événements après l'insertion d'une nouvelle ligne dans la table `emp`. Les valeurs anciennes et nouvelles sont omises car elles ne sont pas utilisées.

event_type	table	table_key	column_name	old_value	new_value
7	emp	empno=1	fname	NULL	NULL
7	emp	empno=1	lname	NULL	NULL
7	emp	empno=1	photo	NULL	NULL

   Le XML généré par l'Éditeur figure ci-dessous. Notez que les valeurs reflètent le contenu actuel de la table `emp` et pas son contenu initial.

```
<add class-name="emp">
    <association>empno=1,table=emp,schema=dirxml
    </association>
    <add-attr attr-name="fname">
        <value type="string">John</value>
    </add-attr>
    <add-attr attr-name="lname">
        <value type="string">Doe</value>
    </add-attr>
    <add-attr attr-name="photo">
        <value type="octet">u7s=</value>
    </add-attr>
</add>
```

8. Mise à jour de champ (avec retour d'interrogation)

   Le tableau ci-dessous montre le contenu de la table de consignation des événements après la mise à jour de la ligne dans la table `emp` : Les valeurs anciennes et nouvelles sont omises car elles ne sont pas utilisées.

event_type	table	table_key	column_name	old_value	new_value
8	emp	empno=1	fname	NULL	NULL
8	emp	empno=1	lname	NULL	NULL
8	emp	empno=1	photo	NULL	NULL

   Le XML généré par l'Éditeur figure ci-dessous. Notez que les valeurs reflètent le contenu actuel de la table `emp` et pas son contenu initial.

```
<modify class-name="emp">
    <association>empno=1,table=emp,schema=dirxml
    </association>
    <modify-attr attr-name="fname">
        <remove-all-values/>
       <add-value>
            <value type="string">John</value>
        </add-value>
    </modify-attr>
    <modify-attr attr-name="lname">
        <remove-all-values/>
       <add-value>
            <value type="string">Doe</value>
        </add-value>
    </modify-attr>
    <modify-attr attr-name="photo">
        <remove-all-values/>
       <add-value>
            <value type="octet">u7s=</value>
        </add-value>
    </modify-attr>
</modify>
```

Utilisation du langage SQL dans des événements XML

La section suivante contient des informations qui vous permettront d'insérer du code SQL dans les événements XML.

Tous les exemples font référence à la table `emp` ci-dessous. La méthode de génération de clé primaire utilisée pour obtenir des valeurs de clé primaire importe peu dans les exemples de cette section.

```
CREATE TABLE emp
(
    empno NUMERIC(8) NOT NULL,
    fname VARCHAR2(64),
    lanem VARCHAR2(64),

    CONSTRAINT pk_emp_empno PRIMARY KEY(empno)
);
```

Remarque : le préfixe d'espace de nom `jdbc` utilisé dans toute cette section est implicitement lié à l'espace de nom `urn:dirxml:jdbc` lorsqu'il est cité en dehors d'un document XML.

Introduction

Vos pouvez utiliser du code SQL incorporé dans des événements XML. Tout comme il est possible d'installer des déclencheurs de base de données sur une table pour provoquer des effets secondaires dans une base de données, le code SQL incorporé dans des événements XML joue le rôle d'un déclencheur virtuel doté des mêmes fonctionnalités.

Le code SQL est incorporé aux événements XML par l'intermédiaire des éléments `<jdbc:statement>` et `<jdbc:sql>`. L'élément `<jdbc:statement>` peut contenir un ou plusieurs éléments `<jdbc:sql>`.

L'exemple de code XML suivant comporte une instruction SQL incorporée.

```
<input xmlns:jdbc="urn:dirxml:jdbc">
     <add class-name="emp">
          <add-attr name="lname">
               <value>Doe</value>
          </add-attr>
     </add>
     <jdbc:statement>
           <jdbc:sql> UPDATE dirxml.emp SET fname = 'John'
           </jdbc:sql>
     </jdbc:statement>
</input>
```

Comme l'objet Abonné résout les événements d'`<ajout>` pour une ou plusieurs instructions d'insertion, le code XML ci-dessus serait résolu comme suit :

```
INSERT INTO dirxml.emp(lname)VALUES('Doe');
UPDATE dirxml.emp SET fname = 'John';
```

Important : utilisez des éléments et attributs qui indiquent l'espace de nom en préfixe pour incorporer du code SQL (sinon, le pilote ne le reconnaîtra pas). Dans l'exemple ci-dessus, l'espace de nom est `urn:dirxml:jdbc`. Le préfixe est l'identificateur qui figure à droite de l'identificateur `xmlns`. Dans l'exemple ci-dessus, le préfixe est `jdbc`. En pratique, le préfixe peut être ce que vous voulez, à condition qu'il soit lié au bon espace de nom.

Substitution de variables

Au lieu de vous imposer une analyse syntaxique des valeurs des champs d'une association, l'objet Abonné prend en charge la substitution de variable dans les instructions SQL incorporées. Exemple :

```
<input xmlns:jdbc="urn:dirxml:jdbc">
    <modify class-name="emp">
        <association>empno=1,table=emp,schema=dirxml
        </association>
        <modify-attr name="lname">
            <add-value>
                <value>DoeRaeMe</value>
            </add-value>
        </modify-attr>
    </modify>
    <jdbc:statement>
           <jdbc:sql>UPDATE dirml.emp SET fname = 'John' WHERE
           empno = {$empno}</jdbc:sql>
    </jdbc:statement>
</input>
```

Les marques de réservation variables doivent respecter la syntaxe modèle des valeurs d'attribut XSLT : {$*nom-champ*} et l'élément d'association doit précéder l'élément <jdbc:statement> dans le document XML ou figurer en tant qu'enfant de l'élément <jdbc:statement>. La marque *nom-champ* doit faire référence à l'un des noms d'attribut RDN dans la valeur d'association. Dans l'exemple ci-dessus, il n'existe qu'un seul attribut de dénomination, empno.

Un événement d'<ajout> est le seul qui n'exige pas d'élément d'association pour traiter les instructions SQL incorporées avec substitution de variable, parce que l'association n'a pas encore été créée. Par ailleurs, les instructions SQL incorporées qui utilisent la substitution de variable doivent suivre, et non précéder, l'événement d'<ajout>. Exemple :

```
<input xmlns:jdbc="urn:dirxml:jdbc">
     <add class-name="emp">
          <add-attr name="lname">
               <value>Doe</value>
          </add-attr>
     </add>
     <jdbc:statement>
          <jdbc:sql>UPDATE dirxml.emp SET fname = 'John' WHERE
          empno = {$empno}</jdbc:sql>
     </jdbc:statement>
</input>
```

Pour empêcher le suivi des informations personnelles, il est possible d'utiliser {$$password} pour désigner le contenu d'un élément <password> dans le même document.

```
<input xmlns:jdbc="urn:dirxml:jdbc">
   <add class-name="emp">
      <add-attr name="lname">
         <value>Doe</value>
      </add-attr>
      <password>Dupont{$empno}</password>
   </add>
   <jdbc:statement>
      <jdbc:sql>CREATE USER Dupont IDENTIFIED BY
      {$$password}</jdbc:sql>
   </jdbc:statement>
</input>
```

Placement des instructions

De la même manière que les déclencheurs de base de données peuvent être exécutés avant ou après une instruction qui les déclenche, le code SQL incorporé peut être placé avant ou après l'événement XML déclenchant. Les exemples suivants illustrent comment incorporer du code SQL avant ou après un événement XML.

Avant le déclencheur

```
<input xmlns:jdbc"urn:dirxml:jdbc">
      <jdbc:statement>
          <association>empno=1,table=emp,schema=dirxml
          </association>
          <jdbc:sql>UPDATE dirxml.emp SET fname = 'John' WHERE
            empno = ${empno}</JDBC:SQL>
      </jdbc:statement>
     <modify class-name="emp">
          <association>empno=1,table=emp,schema=dirxml
```

```
          </association>
          <modify-attr name="lname">
               <remove-all-values/>
               <add-value>
                    <value>Doe</value>
               </add-value>
          </modify-attr>
     </modify>
</input>
```

Les données XML ci-dessus sont converties comme suit :

```
UPDATE dirxml.emp SET fname = 'John' WHERE empno = 1;
UPDATE dirxml.emp SET lname = 'Doe' WHERE empno = 1;
```

Après le déclencheur

```
<input xmlns:jdbc"urn:dirxml:jdbc">
     <modify class-name="emp">
          <association>empno=1,table=emp,schema=dirxml
          </association>
          <modify-attr name="lname">
               <remove-all-values/>
               <add-value>
                    <value>Doe</value>
               </add-value>
          </modify-attr>
     </modify>
     <jdbc:statement>
          <jdbc:sql>UPDATE dirxml.emp SET fname = 'John' WHERE
            empno = {$empno}</jdbc:sql>
     </jdbc:statement>
</input>
```

Les données XML ci-dessus sont converties comme suit :

```
UPDATE dirxml.emp SET lname = 'Doe' WHERE empno = 1;
UPDATE dirxml.emp SET fname = 'John' WHERE empno = 1;
```

Transactions manuelles et automatiques

Vous pouvez regrouper manuellement du code SQL et des événements XML à l'aide des deux attributs personnalisés suivants :

- jdbc:transaction-type
- jdbc:transaction-id

jdbc:transaction-type

Cet attribut possède deux valeurs : `manual` et `auto`. Par défaut, la plupart des événements XML qui présentent un intérêt sont des transactions de type manual. Le paramètre « manual » permet aux événements XML d'être convertis en plusieurs instructions SQL.

Le type de transaction auto est attribué par défaut aux événements SQL incorporés, car certaines instructions SQL ne peuvent pas être incluses dans une transaction manuelle.

```
<input xmlns:jdbc="urn:dirxml:jdbc">
     <add class-name="emp" jdbc:transaction-type="auto">
           <add-attr name="lname">
```

```
                <value>Doe</value>
            </add-attr>
      </add>
      <jdbc:statement>
            <jdbc:sql>UPDATE dirxml.emp SET fname = 'John' WHERE
            empno = {$empno}</jdbc:sql>
      </jdbc:statement>
</input>
```

Les données XML ci-dessus sont converties comme suit :

```
INSERT INTO dirxml.emp(lname) VALUES('Doe');
/* COMMIT; implicit */

UPDATE dirxml.emp SET fname = 'John' WHERE empno = 1;
/* COMMIT; implicit */
```

jdbc:transaction-id

L'objet Abonné ignore cet attribut, sauf si l'attribut `jdbc:transaction-type` de l'élément a par défaut ou explicitement la valeur **`manual`**. Le code XML suivant représente un exemple de transaction manuelle :

```
<input xmlns:jdbc="urn:dirxml:jdbc">
      <add class-name="emp" jdbc:transaction-id="0">
            <add-attr name="lname">
                  <value>Doe</value>
            </add-attr>
      </add>
      <jdbc:statement jdbc:transaction-type="manual"
        jdbc:transaction-id="0">
            <jdbc:sql>UPDATE dirxml.emp SET fname = 'John' WHERE
            empno = {$empno}</jdbc:sql>
      </jdbc:statement>
</input>
```

Le code XML ci-dessus est converti comme suit :

```
INSERT INTO dirxml.emp(lname) VALUES('Dupont');
UPDATE dirxml.emp SET fname = 'John' WHERE empno = 1;
COMMIT; /* explicit */
```

Niveau d'isolation de transaction

Outre le regroupement d'instructions, les transactions permettent de préserver l'intégrité des données d'une base de données. Les transactions peuvent verrouiller les données afin d'empêcher tout accès concurrent ou toute modification. Le réglage des verrous est déterminé par le niveau d'isolation d'une transaction. En général, le niveau d'isolation par défaut utilisé par le pilote est suffisant et ne doit pas être modifié.

L'attribut personnalisé `jdbc:isolation-level` vous permet de régler le niveau d'isolation de transaction en cas de besoin. Cinq valeurs possibles sont définies dans l'interface java.sql.Connection :

- `aucune`
- `lecture non validée`
- `lecture validée`

- lecture renouvelée
- sérialisable

Le niveau d'isolation de transaction par défaut du pilote est `lecture validée`. En cas de transaction manuelle, l'attribut `jdbc:isolation-level` doit être placé sur le premier élément de la transaction. Cet attribut est ignoré sur les éléments qui suivent. Exemple :

```
<input xmlns:jdbc="urn:dirxml:jdbc">
     <add class-name="emp" jdbc:transaction-id="0"
      jdbc:isolation-level="serializable">
           <add-attr name="lname">
                <value>Doe</value>
           </add-attr>
      </add>
      <jdbc:statement jdbc:transaction-type="manual"
       jdbc:transaction-id="0">
           <jdbc:sql>UPDATE dirxml.emp SET fname = 'John'
             WHERE empno = {$empno}</jdbc:sql>
      </jdbc:statement>
</input>
```

Les données XML ci-dessus sont converties comme suit :

```
INSERT INTO dirxml.emp(lname) VALUES('Doe');
UPDATE dirxml.emp SET fname = 'John' WHERE empno = 1;
COMMIT; /* explicit */
```

Type d'instruction

Le pilote exécute les instructions SQL incorporées, sans les comprendre. L'interface JDBC définit plusieurs méthodes d'exécution de différents types d'instructions SQL. Le tableau suivant recense ces méthodes :

Type d'instruction	Méthode d'exécution
SELECT	Statement.executeQuery(String)
INSERT	Statement.executeUpdate(String)
UPDATE	Statement.executeUpdate(String)
DELETE	Statement.executeUpdate(String)
CALL ou EXECUTE	Statement.execute(String)
L'une quelconque des instructions ci-dessus	

La solution la plus simple est d'assigner toutes les instructions SQL à la méthode execute(). Par défaut, c'est la méthode retenue par le pilote. Certains pilotes de fabricants tiers, notamment le pilote JDBC d'Oracle, mettent incorrectement en œuvre les méthodes utilisées pour déterminer le nombre de résultats générés par la méthode execute(). En conséquence, le pilote peut être pris dans une boucle infinie qui entraîne une forte utilisation de l'unité centrale. Pour éviter ce problème, il est possible d'utiliser l'attribut `jdbc:type` sur n'importe quel élément `<jdbc:statement>` pour en assigner les instructions SQL aux méthodes executeQuery() ou executeUpdate() au lieu de la méthode execute() par défaut.

L'attribut jdbc:type possède deux valeurs : update et query. Il convient de définir la valeur update pour les instructions insert, update ou delete et la valeur query pour les instructions select. En l'absence de cet attribut, le pilote assigne toutes les instructions SQL à la méthode execute(). S'il est placé sur un autre élément que <jdbc:statement>, cet attribut est ignoré.

Nous vous recommandons d'affecter la valeur d'attribut jdbc:type="query" à toutes les instructions select et l'attribut jdbc:type="update" à toutes les instructions insert, update et delete.

Le code XML suivant contient un exemple d'attribut jdbc:type :

```
<input xmlns:jdbc="urn:dirxml:jdbc">
     <add class-name="emp">
          <add-attr name="lname">
               <value>Doe</value>
          </add-attr>
     </add>
     <jdbc:statement jdbc:type="update">
          <jdbc:sql>UPDATE dirxml.emp SET fname = 'John'
           WHERE empno = {$empno}</jdbc:sql>
     </jdbc:statement>
</input>
```

Requêtes SQL

Pour prendre pleinement en charge les fonctionnalités de requête d'une base de données et éviter la difficile conversion des requêtes SQL natives au format XML, le pilote prend en charge le traitement des requêtes SQL natives. Les instructions select peuvent être incorporées à des documents XML exactement comme les autres instructions SQL.

Par exemple, si le contenu de la table emp était le suivant :

empno	fname	lname
1	Jean	Untel

Le document XML ci-dessous donnerait un document de sortie contenant un seul ensemble de résultats.

```
<input xmlns:jdbc="urn:dirxml:jdbc">
     <jdbc:statement jdbc:type="query">
          <jdbc:sql>SELECT * FROM dirxml.emp</jdbc:sql>
     </jdbc:statement>
</input>

<output xmlns:jdbc="urn:dirxml:jdbc">
     <jdbc:result-set jdbc:number-of-rows="1">
          <jdbc:row jdbc:number="1">
               <jdbc:column jdbc:name="empno"
                            jdbc:position="1"
                            jdbc:type="java.sql.Types.DECIMAL"
                    <jdbc:value>1</jdbc:value>
                </jdbc:column>
                <jdbc:column jdbc:name="fname"
                             jdbc:position="2"
                             jdbc:type="java.sql.Types.VARCHAR">
                    <jdbc:value>John</jdbc:value>
                 </jdbc:column>
```

```
                    <jdbc:column jdbc:name="lname"
                              jdbc:position="3"
                                jdbc:type="java.sql.Types.VARCHAR>
                          <jdbc:value>Doe</jdbc:value>
                     </jdbc:column>
                </jdbc:row>
           </jdbc:result-set>
           <status level="success"/>
</output>
```

Les requêtes SQL produisent toujours un élément `<jdbc:result-set>` unique, que l'ensemble de résultats contienne ou non des lignes. Si l'ensemble de résultats est vide, l'attribut `jdbc:number-of-rows` sera défini sur zéro.

Il est possible d'incorporer plusieurs requêtes dans un document. Les requêtes SQL n'exigent pas que les tables référencées soient connues du pilote, alors que les requêtes XML l'exigent.

Instructions en langage DDL (Data Definition Language - Langage de définition de données)

Il est généralement impossible d'exécuter une instruction DDL dans un déclencheur de base de données, car la plupart des bases n'autorisent pas les transactions DML et DDL mixtes. Bien que les déclencheurs virtuels ne permettent pas de surmonter cette limite de transaction, ils permettent l'exécution d'instructions DDL en tant qu'effets secondaires d'un événement XML. Exemple :

```
<input xmlns:jdbc="urn:dirxml:jdbc">
     <add class-name="emp">
          <add-attr name="lname">
               <value>Doe</value>
          </add-attr>
     </add>
     <jdbc:statement>
          <jdbc:sql>CREATE USER dirxml IDENTIFIED BY novell
          </jdbc:sql>
     </jdbc:statement>
</input>
```

Les données XML ci-dessus sont converties comme suit :

```
INSERT INTO dirxml.emp(lname) VALUES('Doe');
/* COMMIT;  implicit */

CREATE USER dirxml IDENTIFIED BY novell;
/* COMMIT;  implicit */
```

L'utilisation des attributs `jdbc:transaction-id` et `jdbc:transaction-type` pour regrouper des instructions DML et DDL en une seule transaction entraînerait l'annulation de cette transaction dans la plupart des bases de données. Comme les instructions DDL sont généralement exécutées en tant que transactions distinctes, il est possible que l'instruction d'insertion de l'exemple ci-dessus aboutisse et que l'instruction de création d'utilisateur soit restaurée à son état initial. Par contre, il est impossible que l'instruction d'insertion échoue et que l'instruction de création d'utilisateur aboutisse. Le pilote arrête d'exécuter des transactions en chaîne dès que la première transaction est annulée.

Opérations logiques

Comme il est généralement impossible de mélanger des instructions DML et DDL dans une seule transaction, un événement unique peut se composer d'une ou de plusieurs transactions. Les attributs `jdbc:op-id` et `jdbc:op-type` peuvent être utilisés pour regrouper plusieurs transactions en une seule opération logique. Dans un regroupement de ce type, tous les membres de l'opération sont traités comme une seule entité en ce qui concerne leur état. Autrement dit, si un membre de l'opération échoue, tous les membres renvoient le même niveau d'état. De même, tous les membres partagent le même type d'état.

```
<input xmlns:jdbc="urn:dirxml:jdbc">
     <add class-name="emp" jdbc:op-id="0"
          jdbc:op-type="password-set-operation">
          <add-attr name="lname">
               <value>Doe</value>
          </add-attr>
          <password>Doe{$empno}</password>
     </add>
     <jdbc:statement jdbc:op-id="0">
          <jdbc:sql>CREATE USER Doe IDENTIFIED BY {$$password}
          </jdbc:sql>
     </jdbc:statement>
</input>
```

L'attribut `jdbc:op-type` est ignoré sur tous les éléments à l'exception du premier élément de l'opération.

Meilleures pratiques

Dans un souci de performance, il est préférable d'appeler une seule procédure stockée qui contient plusieurs instructions plutôt que d'incorporer plusieurs instructions SQL dans un document XML. Exemple :

```
<input xmlns:jdbc="urn:dirxml:jdbc">
      <add class-name="emp">
           <add-attr name="lname">
                <value>Doe</value>
           </add-attr>
      </add>
      <jdbc:statement>
           <jdbc:sql>CALL PROCEDURE set_fname('John', 'Joe',
           'Jimmy')</jdbc:sql>
      </jdbc:statement>
</input>
```

est préférable à :

```
<input xmlns:jdbc="urn:dirxml:jdbc">
      <add class-name="emp">
           <add-attr name="lname">
                <value>Doe</value>
           </add-attr>
     </add>
     <jdbc:statement>
          <jdbc:sql>UPDATE dirxml.emp SET fname = 'John'
           WHERE empno = {$empno}</jdbc:sql>
     </jdbc:statement>
     <jdbc:statement>
          <jdbc:sql>UPDATE dirxml.emp SET fname = 'Joe'
           WHERE empno = {$empno}</jdbc:sql>
```

```
        </jdbc:statement>
        <jdbc:statement>
            <jdbc:sql>UPDATE dirxml.emp SET fname = 'Jimmy'
             WHERE empno = {$empno}</jdbc:sql>
        </jdbc:statement>
    </input>
```

6 Utilitaire d'association JDBC

Cette section contient des informations sur l'utilisation de l'outil d'association JDBC. Cet utilitaire est conçu pour normaliser les associations d'objets effectuées avec les pilotes des versions 1.0 ou ultérieures. Il offre également plusieurs autres fonctions qui permettent de simplifier l'administration du pilote.

Cette version de l'utilitaire est compatible avec les anciennes versions du pilote JDBC jusqu'à la 1.0 ; elle remplace toutes les versions précédentes de l'utilitaire.

Fonctionnement de l'utilitaire

Cet utilitaire prend en charge sept opérations indépendantes :

1. Liste des objets associés à un pilote (par défaut)
2. Liste des objets qui comportent plusieurs associations à un pilote
3. Liste des objets qui comportent des associations incorrectes à un pilote

   Une association est incorrecte dans les cas suivants :

   - Elle est mal formée. (Par exemple, le RDN du schéma ou de la table est absent, ou le mot clé du schéma est mal orthographié.)
   - Elle contient des identificateurs de base de données qui ne sont pas assignés à des identificateurs dans la base de données cible. (Par exemple, une association comprend une assignation à une table qui n'existe pas.)
   - Elle n'est assignée à aucune ligne ou à plusieurs lignes. Or, une association est rompue si elle n'est pas assignée à une ligne. Par ailleurs, les associations ne sont pas uniques si elles sont assignées à plusieurs lignes.
4. Liste des objets qui doivent être normalisés
   - Une association normalisée est correcte, bien ordonnée et utilise la casse qui convient. La casse normale est en majuscules pour les bases de données qui ne font pas la distinction majuscules/minuscules et en casse mixte pour les bases de données qui font cette distinction.
5. Normalisation des associations d'objets listées par l'opération précédente
6. Liste des associations d'objets à modifier
   - Permet le remplacement global des noms de schéma, de table et de colonne en fonction de critères de recherche.
7. Modification des associations d'objets listées par l'opération précédente

Le tableau suivant dresse la liste des opérations en indiquant si elles sont en lecture seule ou en écriture.

Opération	Lecture seule ou écriture
1. Liste des objets associés à un pilote	Lecture seule
2. Liste des objets qui comportent plusieurs associations à un pilote	Lecture seule
3. Liste des objets qui comportent des associations incorrectes à un pilote	Lecture seule
4. Liste des objets qui doivent être normalisés	Lecture seule
5. Normalisation des associations d'objets listées par l'opération précédente	Écriture
6. Liste des associations d'objets à modifier	Lecture seule
7. Modification des associations d'objets listées par l'opération précédente	Écriture

Avant de commencer

La modification d'associations peut provoquer des problèmes. Si des associations sont corrompues, Identity Manager arrête de fonctionner ; n'utilisez donc des opérations d'écriture que lorsque c'est nécessaire. Pour éviter toute altération involontaire d'associations, cet utilitaire crée un fichier ldiff d'annulation pour toutes les opérations d'écriture.

Lisez les recommandations suivantes avant d'utiliser cet utilitaire :

- Cet utilitaire, comme le pilote, suppose que les identificateurs de base de données ne sont pas délimités (sans guillemets et sans caractères spéciaux).
- Il est extrêmement important que les associations d'objets liées à un pilote soient mises à jour en même temps.
  - Pour afficher tous les objets associés à un pilote particulier, cet utilitaire doit être exécuté sur le serveur eDirectory™ Novell® sur lequel le pilote fonctionne ou à partir duquel il s'exécute à distance.
  - Tous les objets associés à un pilote particulier doivent être endigués par la base de recherche LDAP.

    Remarque : pour assurer un endiguement complet, nous vous recommandons d'utiliser le conteneur racine de votre arborescence comme base de recherche.
- Assurez-vous que l'URL JDBC de la base de données cible fournie à cet utilitaire est la même que celle du pilote. En effet, si vous appliquez cet utilitaire à une base de données sans distinction majuscules/minuscules, alors que la base opère en fait cette distinction, les associations risquent d'être normalisées dans une casse incorrecte.
- Comme cet utilitaire est exécuté en local, il utilise une connexion non sécurisée, si bien que le serveur LDAP eDirectory doit être temporairement configuré pour accepter les mots de passe en texte clair. En fonction du pilote JDBC tiers que vous utilisez, la connexion de base de données établie par cet utilitaire risque de ne pas être sécurisée.

  Remarque : nous vous recommandons de changer le mot de passe d'authentification du module d'interface sur la base de données après avoir exécuté cet utilitaire.

Utilisation de l'utilitaire

Cet utilitaire doit être exécuté une fois pour chaque instance du pilote installée sur le serveur cible.

Un fichier de propriétés est fourni pour chaque base de données prise en charge ; son chemin est tools\sql*basedonnées*\properties.txt.

Remarque : pour plus d'informations sur l'exécution de l'utilitaire à partir de la ligne de commande, reportez-vous au fichier run.bat dans le répertoire tools\util de l'image de téléchargement.

1. Arrêtez le pilote.
2. Identifiez et supprimez les associations superflues (opérations 2 et 3).

   Aucun objet associé par le pilote JDBC ne doit comporter plusieurs associations. Les associations superflues doivent être supprimées manuellement pour chaque objet. L'opération 3 peut vous aider à identifier laquelle des diverses associations est effectivement valable. Une fois cette association identifiée, vous pouvez supprimer les autres.
3. Identifiez et corrigez les associations incorrectes (opération 3, éventuellement opérations 6 et 7).

   En règle générale, si le problème est isolé, modifiez chaque association incorrecte manuellement. Si le problème est répétitif et porte sur un grand nombre d'associations, envisagez d'utiliser les opérations 6 et 7. Cet utilitaire permet de remplacer globalement les identificateurs incorrects, mais ne peut pas en insérer ou en supprimer lorsqu'il n'en existe pas.
4. Normalisez les associations (opérations 4 et 5).

Édition d'associations

Cet utilitaire exige deux paramètres (*oldRDN* et *newRDN*) pour les opérations 6 et 7. Cette section explique comment utiliser ces paramètres.

La première valeur est le critère de recherche, tandis que la seconde est la valeur de remplacement. Le caractère joker `*` peut être employé dans certains cas pour généraliser le critère de recherche ou la valeur de remplacement.

Trois types d'opérations de recherche et de remplacement sont possibles :

1. Remplacement du nom de schéma

   Les caractères joker ne sont acceptés que du côté droit. Par exemple,

   - Remplacer le schéma `old` (ancien) par le schéma `new` (nouveau)

     oldRDN : **`schema=old`**

     newRDN : **`schema=new`**
2. Remplacement du nom de table

   Les caractères joker ne sont pas pris en charge. Par exemple,

   - Remplacer la table `old` (ancienne) par la table `new` (nouvelle)

     oldRDN : **`table=old`**

     newRDN : **`table=new`**
3. Remplacement du nom de colonne

   Les caractères joker sont obligatoires du côté droit, mais ne sont pas pris en charge du côté gauche. Par exemple,

   - Remplacer la colonne `old` (ancienne) par la colonne `new` (nouvelle)

     oldRDN : **`old=*`**

     newRDN : **`new=*`**

7 Désinstallation des objets Pilote et Base de données

Cette section indique comment désinstaller un pilote et les objets de base de données correspondants.

Désinstallation du pilote

Cette section fournit des informations sur la désinstallation du pilote.

Lors de la suppression d'objet eDirectory® Novell, vous devez supprimer tous les objets enfant avant de pouvoir supprimer un objet parent. Ainsi, vous devez supprimer toutes les règles et feuilles de style sur le canal Éditeur avant de pouvoir supprimer l'objet Éditeur lui-même. De même, vous devez supprimer les objets Éditeur et Abonné avant de pouvoir supprimer l'objet Pilote.

Pour supprimer un objet Pilote d'eDirectory :

1. Dans Novell iManager, cliquez sur Gestion DirXML > Présentation.
2. Dans Présentation, localisez l'ensemble de pilotes dans lequel figure le pilote, puis cliquez sur Supprimer le pilote.
3. Cliquez sur le pilote à supprimer, puis cliquez sur ok.

Désinstallation d'objets de base de données

Cette section fournit des informations et des procédures qui permettent de désinstaller les objets de base de données.

Cette section contient des informations utiles pour :

- « Désinstallation des objets Oracle », page 80
- « Désinstallation des objets Microsoft SQL Server », page 80
- « Désinstallation des objets IBM DB2 UDB », page 80
- « Désinstallation des objets Sybase », page 80
- « Désinstallation des objets MySQL », page 81
- « Désinstallation des objets Informix », page 81

Important : il est recommandé d'installer et de désinstaller les pilotes préconfigurés et les scripts de base de données en bloc. Pour éviter tout problème de discordance, les scripts de base de données et les pilotes préconfigurés contiennent désormais un en-tête qui comprend un numéro de version, le nom de la base de données cible et la version de la base de données.

Désinstallation des objets Oracle

1. À partir d'un client Oracle, comme SQL Plus, loguez-vous en tant qu'utilisateur **SYSTEM**. Par défaut, le mot de passe pour l'utilisateur SYSTEM est **MANAGER**.
2. Exécutez le script de désinstallation en vue d'une synchronisation directe ou indirecte. Exemple :

   SQL> @c:\tools\sql\oracle\direct\UNINSTALL_DIRECT.sql

   SQL> @c:\tools\sql\oracle\indirect\UNINSTALL_INDIRECT.sql

Désinstallation des objets Microsoft SQL Server

1. Démarrez l'Analyseur de requêtes.
2. Loguez-vous à votre serveur sous le nom d'utilisateur **sa**. Par défaut, le compte sa n'est protégé par aucun mot de passe.
3. Ouvrez et exécutez le script de désinstallation en vue d'une synchronisation directe ou indirecte. Exemple :

   tools\sql\mssql\direct\UNINSTALL_DIRECT.sql

   tools\sql\mssql\indirect\UNINSTALL_INDIRECT.sql

Désinstallation des objets IBM DB2 UDB

1. Démarrez le Command Center (Centre de commande).
2. Cliquez sur l'onglet Script > ouvrez le menu Script > importez le script de désinstallation en vue d'une synchronisation directe ou indirecte. Exemple :

   tools\sql\db2\direct\UNINSTALL_DIRECT.sql

   tools\sql\db2\indirect\UNINSTALL_INDIRECT.sql
3. Changez le nom du compte et le mot de passe administrateur de votre serveur avant d'exécuter le script de désinstallation.
4. Exécutez le script.

Remarque : le script de désinstallation ne détruit ni la base de données dirxml, ni le compte utilisateur du système dirxml.

Désinstallation des objets Sybase

1. À partir d'un client Sybase, comme isql, loguez-vous en tant qu'utilisateur **sa** et exécutez le script de désinstallation en vue d'une synchronisation directe ou indirecte. Par défaut, le compte sa n'est protégé par aucun mot de passe. Exemple :

   ```
   isql -U sa -P -i
   c:\tools\sql\sybase\direct\UNINSTALL_DIRECT.sql
   ```

   ```
   isql -U sa -P -i
   c:\tools\sql\sybase\indirect\UNINSTALL_INDIRECT.sql
   ```

Désinstallation des objets MySQL

1. À partir d'un client MySQL, comme mysql, loguez-vous en tant qu'utilisateur **`root`** et exécutez le script de désinstallation en vue d'une synchronisation indirecte. Par défaut, le compte `root` n'est protégé par aucun mot de passe. Exemple :

   mysql> \. c:\tools\sql\oracle\indirect\UNINSTALL_INDIRECT.sql

Désinstallation des objets Informix

1. Démarrez l'Éditeur SQL.
2. Loguez-vous à votre serveur sous le nom d'utilisateur **`informix`**. Par défaut, le mot de passe pour l'utilisateur `informix` est **`informix`**.
3. Exécutez le script de désinstallation en vue d'une synchronisation directe ou indirecte. Exemple :

   tools\sql\informix\direct\UNINSTALL_DIRECT.sql

   tools\sql\informix\indirect\UNINSTALL_INDIRECT.sql

Remarque : le script de désinstallation ne détruit pas le compte utilisateur du système `dirxml`.

A Meilleures pratiques

La section suivante répertorie les meilleures pratiques à adopter pour l'utilisation du pilote. Vous trouverez des informations complémentaires aux chapitres Chapitre 4, « Configuration du pilote », page 35 et Chapitre 5, « Configuration avancée du pilote », page 47.

- Pour une synchronisation directe, vous devez préfixer un ou plusieurs noms de colonnes par `pk_` (sans distinction de majuscules/minuscules).
- Pour une synchronisation indirecte, assurez-vous que toutes les tables qui incluent une classe de base de données logique comportent les mêmes noms de colonnes de clé primaire et étrangère.
- Pour une synchronisation à la fois directe et indirecte, veillez à utiliser des noms de clé primaire et étrangère différents entre les classes de base de données logiques.
- Les valeurs de clé primaire placées dans le champ `table_key` doivent être délimitées (c'est-à-dire, mises entre guillemets) si elles contiennent les caractères suivants : , ; ' + = \ " < >. Ce n'est normalement un problème que si la colonne de clé primaire est du type binaire. Un exemple est fourni dans le répertoire tools\sql\example\pbx.
- Quand eDirectory est la source experte des valeurs de clé primaire, il est recommandé d'utiliser GUID plutôt que CN comme clé primaire. En effet, contrairement à CN, la clé GUID est à valeur unique et ne change pas.
- Les colonnes de clé étrangère doivent toujours être omises des déclencheurs de publication.
- DSTrace ne doit pas être utilisé dans un environnement de production.
- N'insérez pas de colonnes de clé primaire dans des déclencheurs du canal Éditeur si elles sont statiques (c'est-à-dire qu'elles ne changent pas).
- Nous vous recommandons d'affecter la valeur d'attribut `jdbc:type="query"` à toutes les instructions select incorporées et l'attribut `jdbc:type="update"` à toutes les instructions incorporées insert, update et delete.
- Pour des raisons de performances et de sécurité, exécutez le pilote à distance, chaque fois que possible.

B Questions fréquentes

La section suivante répond à certaines questions fréquentes que vous risquez de vous poser lors de l'installation ou de la configuration du pilote. Ces questions sont les suivantes :

Pourquoi le pilote ne voit-il pas mes tables ou vues ?

Le pilote est seulement capable de synchroniser les tables qui possèdent des contraintes de clé primaire explicites. Il se sert de ces contraintes pour déterminer quels champs utiliser pour générer des associations. C'est pour cette raison qu'il ignore toutes les tables non-contraintes.
Si vous tentez d'effectuer une synchronisation avec des tables qui ne comportent pas de contraintes explicites, vous devrez soit ajouter ces contraintes, soit effectuer la synchronisation avec des tables intermédiaires qui contiennent les contraintes requises. Cette seconde solution est la meilleure.

Pour être visible du pilote, une vue doit contenir au moins un nom de colonne qui porte le préfixe `pk_` (sans distinction majuscules/minuscules).

Comment synchroniser des tables situées dans plusieurs schémas ?

Il vous faudra attribuer un alias aux tables dans le schéma de ce pilote, synchroniser avec des tables intermédiaires dans le schéma du pilote et déplacer les données entre les schémas, utiliser une vue ou créer un schéma virtuel par l'intermédiaire du nouveau paramètre de pilote Synchroniser la ou les tables.

Pourquoi le pilote ne traite-t-il pas les enregistrements dans le journal des événements ?

Il y a plusieurs explications à ce comportement. Tout d'abord, vous devez vérifier le champ `perpetrator` des lignes en question et vous assurer que la valeur est différente du nom d'utilisateur du pilote. Le pilote ne vérifie le champ `perpetrator` que si le paramètre de l'éditeur Autoriser le retour en boucle est défini sur `non`. Le pilote empêche le retour en boucle des événements en ignorant tous les enregistrements pour lesquels la valeur du champ `perpetrator` est égale au nom d'utilisateur du pilote.

Vous devez également vous assurer que le champ `status` de l'enregistrement a la valeur 'N' (nouveau). Les enregistrements dont les champs `status` contiennent autre chose que 'N' ne seront pas traités. En outre, veillez à valider explicitement les modifications. En effet, les modifications restent temporaires tant que vous ne les validez pas.

Le pilote peut-il gérer les comptes utilisateur de la base de données ?

Oui, il est possible de gérer les comptes de la base de données à l'aide de code SQL incorporé. Pour plus d'informations, reportez-vous à « Utilisation du langage SQL dans des événements XML », page 65.

Le pilote peut-il synchroniser les données de type binaire et chaîne en grande quantité ?

Oui. Il est possible d'inscrire et de publier des données de type binaire et chaîne en grande quantité. L'acheminement de ces données via le canal Éditeur s'effectue à l'aide de types d'événements avec retour d'interrogation.

Pourquoi l'acheminement des données via le canal Éditeur est-il si lent ?

Si la table de consignation des événements contient un grand nombre de lignes, elle doit être indexée. Tous les scripts d'installation de base de données fournissent des exemples d'index. Les instructions utilisées par le pilote pour tenir à jour le journal des événements peuvent être visualisées à l'aide du niveau de trace 3. Des exemples sont également fournis dans le fichier tools\sql\example\STATEMENTS.sql.

Les index des scripts d'installation peuvent encore être affinés pour améliorer les performances de l'acheminement des données via le canal Éditeur. Le déplacement des index dans un autre espace table ou sur un autre disque physique que le journal des événements contribue également à augmenter ces performances.

En outre, dans un environnement de production, la valeur **non** doit être attribuée au paramètre du canal Éditeur Supprimer du journal ?.

Le pilote peut-il synchroniser plusieurs classes ?

Oui. Les noms de colonne de clé primaire, cependant, doivent être uniques dans les classes de base de données logiques. Par exemple, si *class1* est assignée à *table1* avec le nom de colonne de clé primaire *key1* et *class2* est assignée à *table2* avec le nom de colonne de clé primaire *key2*, le nom de *key1* ne peut pas être identique à *key2*. Cette condition peut toujours être satisfaite si des tables ou vues intermédiaires sont utilisées.

Pourquoi les colonnes de clé étrangère et de clé primaire doivent-elles porter le même nom ?

Dans chaque classe de base de données logique, les noms de colonne de clé primaire et étrangère doivent concorder. Entre classes de base de données logiques, ils doivent être différents. Ce nom commun est utilisé par le canal Éditeur pour identifier tous les enregistrements de la table de consignation des événements qui se rapportent à un objet de base de données logique unique, même si cet objet couvre plusieurs tables.

Le pilote prend-il en charge le codage SSL ?

Non. Le mode de communication du pilote avec une base de données dépend du pilote de fabricant tiers utilisé. Certains pilotes tiers prennent en charge les sockets SSL et d'autres non. Même si le codage SSL est pris en charge, il n'existe aucune méthode normalisée qui permette de l'activer entre pilotes de fabricants tiers. La solution de ce problème, en général, consiste à exécuter à distance le pilote et votre pilote de fabricant tiers, ce qui permet au pilote et à votre pilote de fabricant tiers d'être exécutés en local sur le serveur de base de données. Toutes les données qui transitent par le réseau entre le moteur et le pilote seront alors codées au format SSL.

Une autre possibilité consiste à utiliser un pilote JDBC tiers de type 3 ou 2. Les serveurs d'applications intermédiaires de base de données et les interfaces de programmation client garantissent généralement un certain niveau de sécurité en ce qui concerne la connectivité.

Comment assigner des attributs à valeurs multiples à des champs de base de données à valeur unique ?

Pour plus de détails sur l'assignation d'attributs à valeurs multiples à des champs de base de données à valeur unique, reportez-vous à « Assignation d'attributs à valeurs multiples à des champs de base de données à valeur unique », page 56.

Pourquoi le pilote synchronise-t-il des chaînes incorrectes ?

La base de données et le pilote tiers utilisent sans doute des codages de caractères incompatibles. Pour y remédier, vous pouvez modifier le codage des caractères utilisé par votre pilote tiers.

Pour plus d'informations, reportez-vous aux valeurs de codage des caractères (http://java.sun.com/products/jdk/1.1/docs/guide/intl/encoding.doc.html) définies par Sun.

C Types de données pris en charge

Le pilote est capable de synchroniser les types de données chaîne, numériques, horaires et binaires JDBC 1.0. L'assignation des types de données JDBC aux types de données natives d'une base de données varie en fonction de la base de données. La liste suivante indique les types java.sql pris en charge :

- Types numériques
  - java.sql.Types.BIGINT
  - java.sql.Types.BIT
  - java.sql.Types.DECIMAL
  - java.sql.Types.DOUBLE
  - java.sql.Types.NUMERIC
  - java.sql.Types.REAL
  - java.sql.Types.FLOAT
  - java.sql.Types.INTEGER
  - java.sql.Types.SMALLINT
  - java.sql.Types.TINYINT
- Types chaîne
  - java.sql.Types.CHAR
  - java.sql.Types.LONGCHAR
  - java.sql.Types.VARCHAR
- Types horaires
  - java.sql.Types.DATE
  - java.sql.Types.TIME
  - java.sql.Types.TIMESTAMP
- Types binaires
  - java.sql.Types.BINARY
  - java.sql.Types.VARBINARY
  - java.sql.Types.LONGVARBINARY

D Méthodes java.sql.DatabaseMetaData

Cette section répertorie les méthodes java.sql.DatabaseMetaData obligatoires et facultatives actuellement utilisées par le pilote. Pour plus d'informations sur ces méthodes, reportez-vous à la page Interface MetaData du site Web de Sun (http://java.sun.com/products/jdk/1.2/docs/api).

Méthodes obligatoires :

- java.sql.ResultSet getColumns(java.lang.String catalog, java.lang.String schemaPattern, java.lang.String tableNamePattern, java.lang.String columnNamePattern)
- java.sql.ResultSet getPrimaryKeys(java.lang.String catalog, java.lang.String schema, java.lang.String table)
- java.sql.ResultSet getTables(java.lang.String catalog, java.lang.String schemaPattern, java.lang.String tableNamePattern, java.lang.String[] types)
- boolean storesLowerCaseIdentifiers()
- boolean storesMixedCaseIdentifiers()
- boolean storesUpperCaseIdentifiers()

Méthodes facultatives :

- boolean dataDefinitionCausesTransactionCommit()
- boolean dataDefinitionIgnoredInTransactions()
- java.sql.ResultSet getExportedKeys(java.lang.String catalog, java.lang.String schema, java.lang.String table)
- int getMaxConnections()
- int getMaxColumnsInSelect()
- int getMaxStatements()
- int getMaxStatementLength()
- java.sql.ResultSet getTableTypes()
- java.lang.String getUserName()
- boolean supportsDataDefinitionAndDataManipulationTransactions()
- boolean supportsDataManipulationTransactionsOnly()
- boolean supportsSchemasInDataManipulation()
- boolean supportsSchemasInProcedureCalls()
- boolean supportsTransactions()
- boolean supportsMultipleTransactions()
- boolean supportsTransactionIsolationLevel(int level)

E Méthodes JDBC 1.0

Cette section répertorie les méthodes JDBC 1.0 (autres que les méthodes DatabaseMetaData) utilisées par le pilote. Les méthodes sont organisées par classes. Il est fréquent que les fournisseurs de pilotes tiers recensent les défauts ou les problèmes connus méthode par méthode. Cette section, associée à la documentation des pilotes de fabricants tiers, permet de corriger ou de prévoir les éventuels problèmes d'interfonctionnement.

- java.sql.DriverManager

  java.sql.Connection getConnection(java.lang.String url, java.lang.String user, java.lang.String password)

- java.sql.PreparedStatement

  void clearParameters()
  void setNull(int parameterIndex, int sqlType)
  void setString(int parameterIndex, java.sql.String x)
  void setBoolean(int parameterIndex, boolean x)
  void setBigDecimal(int parameterIndex, java.math.BigDecimal x)
  void setLong(int parameterIndex, long x)
  void setDouble(int parameterIndex, double x)
  void setInt(int parameterIndex, int x)
  void setFloat(int parameterIndex, float x)
  void setShort(int parameterIndex, short x)
  void setByte(int parameterIndex, byte x)
  void setTimestamp(int parameterIndex, java.sql.Timestamp x)
  void setTime(int parameterIndex, java.sql.Time x)
  void setDate(int parameterIndex, java.sql.Date x)
  void setBytes(int parameterIndex, bytes[] x)

- java.sql.Statement

  void clearWarnings()
  void close()
  boolean execute(String sql)
  java.sql.ResultSet executeQuery(String sql)
  int executeUpdate(String sql)
  boolean getMoreResults()
  int getUpdateCount()
  java.sql.ResultSet getResultSet()

- java.sql.CallableStatement

  void registerOutParameter(int parameterIndex, int sqlType)

- java.sql.Connection

  void close()
  void commit()
  void rollback()
  int getTransactionIsolation()
  void setAutoCommit(boolean autoCommit)
  java.sql.PreparedStatement prepareStatement(String sql)
  java.sql.CallableStatement prepareCall(String sql)
  java.sql.Statement createStatement()

- java.sql.ResultSet

  void close()
  boolean next()
  java.lang.String getString(int columnIndex)
  java.lang.String getString(java.lang.String columnName)
  java.math.BigDecimal getBigDecimal(int columnIndex, int scale)
  long getLong(int columnIndex)
  double getDouble(int columnIndex)
  int getInt(int columnIndex)
  float getFloat(int columnIndex)
  short getShort(int columnIndex)
  byte getByte(int columnIndex)
  boolean getBoolean(int columnIndex)
  byte[] getBytes(int columnIndex)
  byte[] getBytes(java.lang.String columnName)
  java.sql.Timestamp getTimestamp(int columnIndex)
  java.sql.Time getTime(int columnIndex)
  java.sql.Date getDate(int columnIndex)
  java.io.InputStream getBinaryStream(String columnName)