

Novell Pilote DirXML pour JDBC*

1.6

www.novell.com

GUIDE D'IMPLÉMENTATION

103-000270-001



Novell®

Notices légales

Novell exclut toute garantie relative au contenu ou à l'utilisation de cette documentation. En particulier, Novell ne garantit pas que cette documentation est exhaustive ni exempte d'erreurs. Novell se réserve en outre le droit de réviser cette publication à tout moment et sans préavis.

Par ailleurs, Novell exclut toute garantie relative à tout logiciel, notamment toute garantie, expresse ou implicite, que le logiciel présenterait des qualités spécifiques ou qu'il conviendrait à un usage particulier. Novell se réserve en outre le droit de modifier à tout moment tout ou partie des logiciels Novell, sans notification préalable de ces modifications à quiconque.

L'exportation ou la réexportation de ce produit est interdite dès lors qu'elle enfreint les lois et réglementations applicables, y compris, de façon non limitative, les réglementations des États-Unis en matière d'exportation ou la législation en vigueur dans votre pays de résidence.

Copyright © 1993-2002 Novell, Inc. Tous droits réservés. Cette publication ne peut être reproduite, photocopiée, stockée sur un système de recherche documentaire ou transmise, même en partie, sans le consentement écrit explicite préalable de l'éditeur.

Brevets américains Nos. 5,608,903 ; 5,671,414 ; 5,677,851 ; 5,758,344 ; 5,784,560 ; 5,794,232 ; 5,818,936 ; 5,832,275 ; 5,832,483 ; 5,832,487 ; 5,870,739 ; 5,873,079 ; 5,878,415 ; 5,884,304 ; 5,913,025 ; 5,919,257 ; 5,933,826. Brevets américains et étrangers en cours d'homologation.

Novell, Inc.
1800 South Novell Place
Provo, UT 84606
U.S.A.

www.novell.com

Pilote DirXML pour JDBC
Octobre 2002

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

ConsoleOne est une marque déposée de Novell, Inc. aux États-Unis et dans d'autres pays.

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.

Autres marques commerciales

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

Tables des matières

	À propos de ce guide	9
1	Présentation du pilote DirXML pour JDBC	11
	Présentation	11
	Nouvelles fonctionnalités	11
	Corrections de bogues	12
	Concepts de pilote	12
	Concepts de base de données	13
	Schéma de base de données	13
	Langage de manipulation de données	14
	Langage de définition de données	14
	Transactions	14
	Déclencheurs de base de données	15
	Procédures/fonctions stockées	16
	Modèles de synchronisation de données	17
	Synchronisation directe	17
	Synchronisation indirecte	19
2	Conditions préalables applicables au pilote	21
	Conditions préalables concernant le pilote	21
	Plates-formes prises en charge	22
	Bases de données prises en charge	22
	Pilotes JDBC de fabricants tiers recommandés	22
	Utilisation d'un pilote ODBC	24
	Utilisation du pilote de pont JDBC-ODBC de Sun	24
	Sécurité	25
	Problèmes connus	25
	Limites	27
	Meilleures pratiques	27

3	Installation ou mise à niveau du pilote	29
	Installation du pilote en cours	29
	Installation du pilote en cours	29
	Installation d'objets de base de données	35
	Configuration des objets de base de données Oracle	36
	Configuration des objets Microsoft SQL Server	37
	Configuration des objets IBM DB2	38
	Configuration des objets Sybase	39
	Configuration des objets MySQL	40
	Configuration des objets Informix	41
	Mise à niveau du pilote	42
	Conditions requises pour la mise à niveau	42
	Mise à niveau de la version 1.5 à la version 1.6	42
	Activation du pilote	43
4	Configuration du pilote	45
	Définition des paramètres d'authentification du pilote	45
	Configuration de l'authentification du pilote	45
	ID d'authentification	46
	Contexte d'authentification	46
	Mot de passe de l'application	47
	Paramètres du pilote	47
	Configuration du pilote	48
	Configuration de l'objet Abonné	54
	Configuration de l'objet Éditeur	57
	Définition des propriétés de l'ensemble de pilotes	60
	Configuration de pilotes de fabricants tiers	60
5	Configuration avancée du pilote	63
	Assignation de schéma	63
	Classes de base de données logiques	64
	Synchronisation indirecte	64
	Synchronisation directe	70
	Assignation d'attributs à valeurs multiples à des champs de base de données à valeur unique	72
	Table de consignment des événements	73
	Colonnes de la table de consignment des événements	73
	Types d'événement	75
	Utilisation du langage SQL dans des événements XML	80
	Introduction	80
	Substitution de variables	81
	Placement des instructions	82
	Transactions manuelles et automatiques	84
	Niveau d'isolation de transaction	85

Type d'instruction	86
Requêtes SQL	87
Instructions en langage DDL (Data Definition Language - Langage de définition de données).	88
Opérations logiques	89
Meilleures pratiques	90
6 Questions fréquentes	93
Réponses aux questions fréquentes sur le pilote	94
Pourquoi le pilote ne voit-il pas mes tables ou vues ?	94
Comment synchroniser des tables situées dans plusieurs schémas ?	94
Pourquoi le pilote ne traite-t-il pas les enregistrements dans le journal des événements ?	94
Le pilote peut-il gérer les comptes utilisateur de la base de données ?	95
Le pilote peut-il synchroniser les données de type binaire et chaîne en grande quantité ?	95
Pourquoi l'acheminement des données via le canal Éditeur est-il si lent ?	95
Le pilote peut-il synchroniser plusieurs classes ?	95
Pourquoi les colonnes de clé étrangère et de clé primaire doivent-elles porter le même nom ?	96
Le pilote prend-il en charge le codage SSL ?	96
Comment assigner des attributs à valeurs multiples à des champs de base de données à valeur unique ?	96
Pourquoi le pilote synchronise-t-il des chaînes incorrectes ?	96
7 Utilitaire d'association JDBC	97
Fonctionnement de l'utilitaire	97
Avant de commencer	99
Utilisation de l'outil	100
Édition d'associations.	101
8 Désinstallation des objets Pilote et Base de données	103
Désinstallation du pilote	103
Désinstallation d'objets de base de données	104
Désinstallation des objets Oracle*	104
Désinstallation des objets Microsoft* SQL Server	104
Désinstallation des objets IBM* DB2 UDB	105
Désinstallation des objets Sybase*.	105
Désinstallation des objets MySQL	106
Désinstallation des objets Informix	106
A Types de données pris en charge	107
B Méthodes java.sql.DatabaseMetaData	109
C Méthodes JDBC	111

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 obtenir une documentation sur l'utilisation de DirXML et des autres pilotes DirXML, accédez au (<http://www.novell.com/documentation/french/dirxml11a/index.html>) site Web de la documentation relative à DirXML.

Mises à jour de la documentation

Vous trouverez la version la plus récente de ce document sur le [site Web de la documentation relative aux pilotes DirXML](http://www.novell.com/documentation/french/dirxmldrivers/index.html) (<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 DirXML 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 », page 11
- ♦ « Corrections de bogues », page 12
- ♦ « Concepts de pilote », page 12
- ♦ « Concepts de base de données », page 13
- ♦ « Modèles de synchronisation de données », page 17

Nouvelles fonctionnalités

- ♦ Le pilote prend maintenant en charge la définition de mot de passe.
- ♦ Les conditions minimales requises pour les bases de données ont été encore réduites.

- ♦ Une prise en charge de MySQL* et d'Informix* Dynamic Server a été ajoutée.
- ♦ Il est désormais possible de publier des données de type binaire et chaîne en grande quantité.
- ♦ Les valeurs de clé primaire peuvent être extraites de fonctions.

Corrections de bogues

- ♦ Les requêtes XDS qui contiennent une association mais aucune valeur d'attribut d'étendue sont automatiquement réinterprétées de façon à avoir une étendue de type entrée.
- ♦ Le pilote effectue maintenant un remplissage des valeurs comparées dans les requêtes à des types de colonnes de caractères de longueur fixe (par ex. CHAR).
- ♦ L'utilitaire d'association ne génère plus de NullPointerException dans certains cas pour les types d'événement 6 et 7.
- ♦ L'objet Éditeur autorise maintenant plusieurs colonnes de table enfant dans son filtre.

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, classes111.zip est l'un des pilotes JDBC d'Oracle.

- ♦ Schéma eDirectory

Ensemble des classes et attributs d'objet de l'annuaire. Par exemple, la classe « User » et l'attribut « Given Name » de 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.

- ♦ Classe de base de données logique

Par classe de base de données logique, on entend l'ensemble des tables et des 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 14
- ♦ « Langage de définition de données », page 14
- ♦ « Transactions », page 14
- ♦ « Déclencheurs de base de données », 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'utilisateurs de base de données particuliers.

La propriété est souvent exprimée par une notation à points 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. 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 (EMPNO, LNAME) VALUES (1, 'Last Name') ;  
UPDATE EMP SET FNAME='First Name' 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 NUMBER (8) ,  
    FNAME VARCHAR2 (32) ,  
    LNAME VARCHAR2 (32)  
);  
  
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(EMPNO,LNAME) VALUES(1,'LAST NAME');  
UPDATE EMP SET FNAME='First Name' WHERE EMPNO=1;  
COMMIT;
```

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(EMPNO,LNAME) VALUES(1,'LAST NAME');
```

Déclencheurs de base de données

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 hypothétique.

```
TABLE EMP  
(  
    EMPNO NUMBER(8)  
    FNAME VARCHAR(32)  
    LNAME VARCHAR(32)  
) ;  
  
CREATE TRIGGER EMP_INSERT  
    AFTER INSERT ON EMP  
    FOR EACH ROW
```

```
BEGIN
    UPDATE EMP SET FNAME = 'First Name' WHERE EMPNO =
        :new.EMPNO;

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 (EMPNO, LNAME) VALUES (1, 'Last Name')
```

le déclencheur 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 = 'First Name' WHERE EMPNO = 1
```

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.

Procédures/fonctions stockées

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

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

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.

Modèles de synchronisation de données

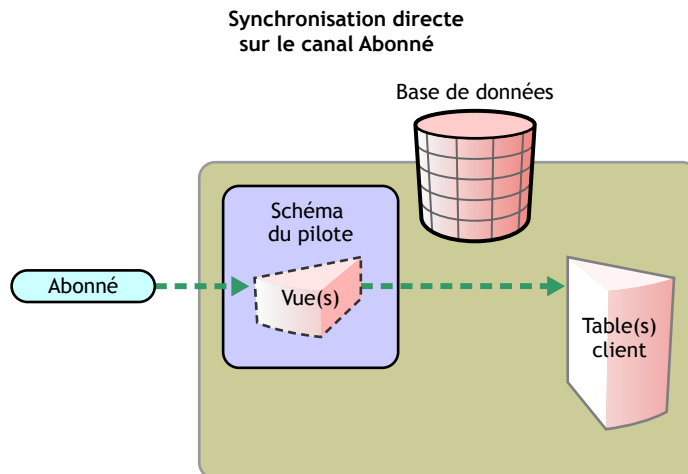
Le pilote prend en charge deux modèles de synchronisation de données : direct et indirect. Les sections suivantes décrivent le fonctionnement de la synchronisation directe et indirecte sur les canaux Abonné et Éditeur.

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.

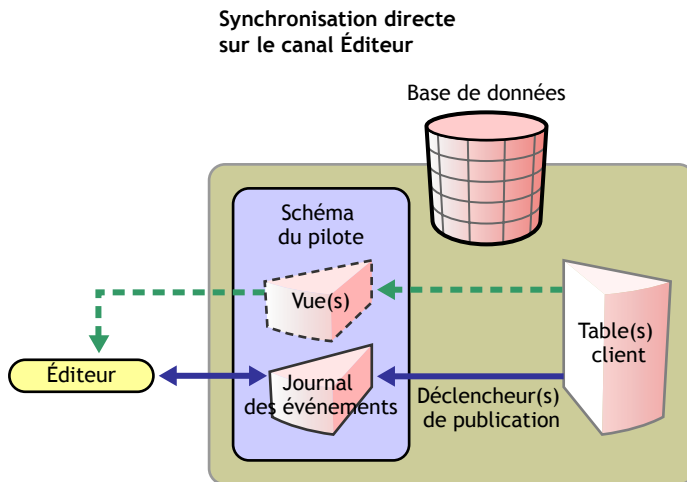
Canal Abonné



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



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

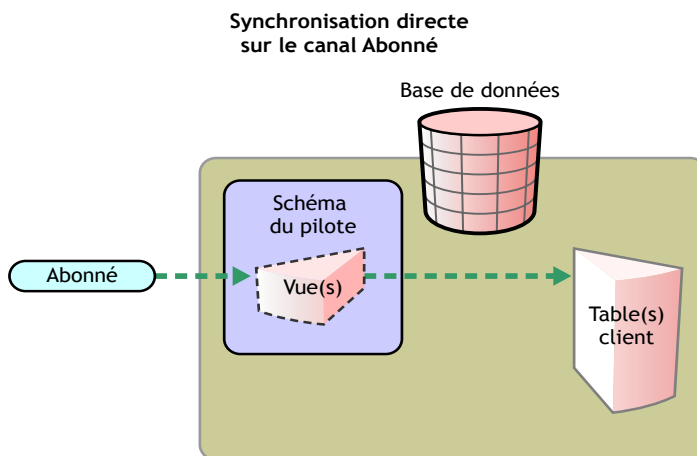
En fonction du contenu des lignes lues dans la table de consignation des événements, le canal Éditeur peut avoir à extraire des informations supplémentaires de la vue avant de mettre à jour eDirectory. Après la mise à jour de eDirectory, le canal Éditeur supprime les lignes ou les marque comme traitées.

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.

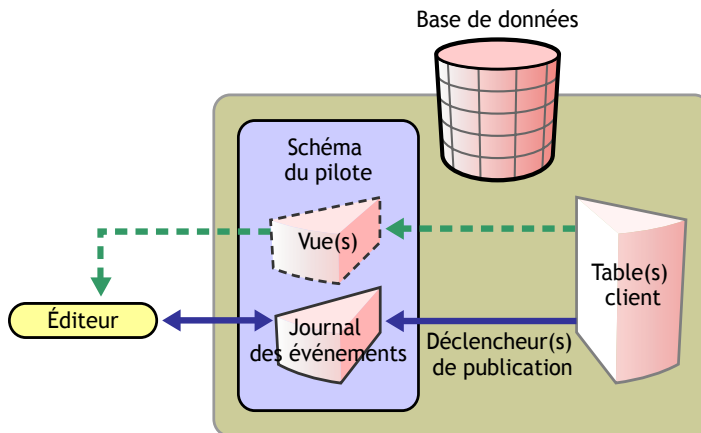
Les scénarios suivants peuvent concerner une ou plusieurs tables ou tables intermédiaires client.

Canal Abonné



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.

Synchronisation directe 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 consignment 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 consignment 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 de eDirectory, le canal Éditeur supprime les lignes ou les marque comme traitées.

2

Conditions préalables applicables au pilote

La section suivante contient des informations importantes à consulter avant d'installer et de configurer le pilote.

Les sujets abordés sont les suivants :

- ♦ « Conditions préalables concernant le pilote », page 21
- ♦ « Plates-formes prises en charge », page 22
- ♦ « Bases de données prises en charge », page 22
- ♦ « Pilotes JDBC de fabricants tiers recommandés », page 22
- ♦ « Utilisation d'un pilote ODBC », page 24
- ♦ « Utilisation du pilote de pont JDBC-ODBC de Sun », page 24
- ♦ « Sécurité », page 25
- ♦ « Problèmes connus », page 25
- ♦ « Limites », page 27
- ♦ « Meilleures pratiques », page 27

Conditions préalables concernant le pilote

Le pilote DirXML[®] pour JDBC exige les éléments suivants :

- ♦ eDirectory[™] 8.6.1 ou version ultérieure
- ♦ DirXML 1.1a ou version ultérieure
- ♦ ConsoleOne[®] 1.3.4
- ♦ Un pilote JDBC/ODBC d'un fabricant tiers

Plates-formes prises en charge

Le pilote s'exécute sur toutes les plates-formes dotées de fonctions DirXML, 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 :

- ♦ IBM* DB2 Universal Database (UDB) 7.2 ou version ultérieure
- ♦ Microsoft* SQL Server 2000, Service Pack 2 ou version ultérieure
- ♦ 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) 9.3 ou version ultérieure

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

- ♦ 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

Les pilotes de fabricants tiers doivent être compatibles avec JDBC 1.0.

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

- ♦ Pilote JDBC Oracle 8i (8.1.7.1)
- ♦ Pilote JDBC Oracle 9i (9.2.0.1)
- ♦ Pilote JDBC IBM pour DB2 UDB

- ♦ BEA* Weblogic Type 4 jDriver pour Microsoft SQL Server 7/2000.
Ce pilote doit être acquis séparément.
- ♦ Pilote JDBC Sybase jConnect 5.5
- ♦ Pilote Microsoft* SQL Server 2000 pour JDBC

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

- ♦ Pilote ODBC Microsoft pour SQL Server

Novell conseille vivement d'utiliser les pilotes tiers recommandés dans la mesure du possible.

Conditions minimales requises pour les pilotes 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 B, « Méthodes `java.sql.DatabaseMetaData` », page 109](#). Cette liste de conditions pourra être augmentée dans les versions ultérieures. Toutes les méthodes `java.sql.DatabaseMetaData` en tant que telles doivent être prises en charge. Si le pilote tiers ne satisfait pas à 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 une liste complète des méthodes JDBC utilisées par le pilote, reportez-vous à l'[Annexe C, « Méthodes JDBC », page 111](#). Consultée parallèlement à la documentation des pilotes de fabricants tiers, cette liste peut permettre d'identifier d'éventuelles incompatibilités.

Considérations relatives à l'utilisation de pilotes 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 et de pilote de fabricants tiers.
- ♦ La prise en charge des procédures stockées et les capacités de connexion (en particulier, la reconnexion) sont des sources d'échec possibles.

Utilisation d'un pilote ODBC

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, Novell recommande vivement d'utiliser un pilote JDBC pur Java (type 3 ou 4) lorsque cela est possible.

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

Utilisation du pilote de pont JDBC-ODBC de Sun

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 de fabricants tiers s'applique également aux pilotes ODBC et JDBC. Pour plus de détails, reportez-vous à « **Pilotes JDBC de fabricants tiers recommandés** », page 22.

- ♦ Si vous optez pour le pilote ODBC de Microsoft pour SQL Server (SQLSRV32.DLL), Novell recommande d'installer la dernière version de Microsoft Data Access Components (MDAC).
- ♦ La dernière version du 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 DirXML. Cette solution a néanmoins pour effet de réduire les performances de tous les pilotes exécutés sur le serveur. DirXML prend uniquement en charge l'utilisation de Hotspot avec JRE version 1.3.x ou ultérieure.

Sécurité

Pour garantir l'établissement d'une connexion sûre entre le pilote et un pilote JDBC tiers, Novell recommande d'exécuter le pilote à distance.

Problèmes connus

Cette section liste les problèmes connus du pilote.

Généralités

- ♦ Lors de l'utilisation de colonnes d'horodatage comme clés primaires avec les types d'événement 5 et 6, 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.ffffff). Certaines bases de données, comme Sybase et DB2, possèdent des formats d'horodatage propriétaire dont la syntaxe ne peut pas être analysée par la classe `java.sql.Timestamp`.

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

- ♦ La dernière version du pilote de pont JDBC-ODBC 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 DirXML. Cette solution a néanmoins pour effet de réduire les performances de tous les pilotes exécutés sur le serveur. DirXML 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 <handle-stmt-results>.
- ♦ 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 le journal 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. Il est donc recommandé 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, 8.0.3x), 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 renvoie un type java.sql.Types.OTHER ambigu pour les types de données NVARCHAR, NTEXT et UNIQUEIDENTIFIER. Or, le pilote suppose que le type java.sql.Types.OTHER est NVARCHAR ou NTEXT. En tant que tel, le type UNIQUEIDENTIFIER n'est pas pris en charge.

Sybase

- ♦ Pour assurer un comportement de remplissage/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.

Limites

La section suivante liste 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 exigent l'utilisation d'identificateurs délimités pour les noms d'utilisateur ou de schéma. Pour le moment, le pilote ne prend pas en charge les identificateurs délimités.
- ♦ L'acheminement des données via le canal Éditeur n'est pas pris en charge sur MySQL en raison de l'absence de prise en charge des déclencheurs dans la version 3.23.
- ♦ Les types de données JDBC 2.0 ne sont pas pris en charge.

Meilleures pratiques

La section suivante liste les meilleures pratiques à adopter pour l'utilisation du pilote. Vous trouverez des informations complémentaires aux chapitres [Chapitre 4, « Configuration du pilote », page 45](#) et [Chapitre 5, « Configuration avancée du pilote », page 63](#).

- ♦ Pour une synchronisation directe, vous devez préfixer une ou plusieurs colonnes par 'PK_' (sans distinction de majuscules/minuscules). Les colonnes préfixées servent à générer des associations. Exemple :

View EMP

PK_EMPNO, (utilisée en association)

FNAME,

LNAME,

- ♦ 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/étrangère. Exemple :
 - ♦ Classe de base de données logique User :
 - Table parent EMP, nom de colonne de clé primaire : EMPNO
 - Table enfant EMP, nom de colonne de clé étrangère : EMPNO

- ♦ Classe de base de données logique Other :

Table parent P, nom de colonne de clé primaire : N

Table enfant C(1), nom de colonne de clé étrangère : N

...

Table enfant C(N), nom de colonne de clé étrangère : N

- ♦ Pour une synchronisation à la fois directe et indirecte, veillez à utiliser des noms de clé primaire/étrangère différents entre les classes de base de données logiques. L'exemple ci-dessus illustre ce principe.
- ♦ 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 :

, ; ' + = \ " < >

Cela ne pose généralement un problème que si la colonne de clé primaire est de type binaire. Un exemple est fourni dans le dossier :

install\example\pbx

- ♦ Pour les pilotes du canal Abonné seulement, 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.
- ♦ DSTrace ne doit pas être utilisé dans un environnement de production parce que les messages de trace sont susceptibles de contenir des informations personnelles. S'il est utilisé, le niveau de trace doit être fixé à une valeur égale ou inférieure à 2.
- ♦ 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).
- ♦ Novell recommande 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.
- ♦ Pour des raisons de performances et de sécurité, exécutez le pilote à distance.

3

Installation ou mise à niveau du pilote

Vous trouverez dans cette section les informations et les procédures qui vous permettront d'effectuer les opérations suivantes :

- ♦ « Installation du pilote en cours », page 29
- ♦ « Installation d'objets de base de données », page 35
- ♦ « Mise à niveau du pilote », page 42
- ♦ « Activation du pilote », page 43

Installation du pilote en cours

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 en cours

- 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

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

- 3** Copiez le fichier de pilote JDBC tiers approprié dans le répertoire que vous avez indiqué à l'étape précédente.

Remarque : Ces fichiers de fabricants tiers ne sont pas fournis avec le pilote DirXML pour JDBC. 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	<p>Oracle Technology Network (http://otn.oracle.com/software/tech/java/sqlj_jdbc/content.html)</p> <p>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.</p>
Oracle	Oracle 9i	classes12.zip, nls_charset12.zip	<p>Oracle Technology Network (http://otn.oracle.com/software/tech/java/sqlj_jdbc/content.html)</p> <p>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.</p>

Fournisseur	Base de données	Nom(s) de fichier(s)	Instructions de téléchargement
BEA Systems	Microsoft SQL Server 7/2000	non précisé(s)	<p>BEA Download Center (http://commerce.bea.com/downloads/weblogic_server.jsp)</p> <p>Ce pilote requiert une inscription gratuite qui expire chaque mois. Dans la section JDBC Drivers, sélectionnez « MSSQLServer4 Kit, Version7 and 2000 » dans la liste déroulante.</p> <p>Vous devez transformer vous-même le fichier en archive jar. Procédez comme suit :</p> <ol style="list-style-type: none"> 1. Placez le fichier weblogic\mssqlserver4v70\license\WeblogicLicense.xml dans le répertoire \weblogic\smsqlserver4v70\classes. 2. Zippez 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	<p>Téléchargements Microsoft (http://www.microsoft.com/sql/downloads/2000/jdbc.asp)</p> <p>Cliquez sur l'option de téléchargement appropriée pour votre plate-forme.</p>
IBM	DB2 Universal Database (UDB) 7.2	db2java.zip	<p>Téléchargements IBM (http://www-4.ibm.com/software/data/db2/udb/downloads.html)</p> <p>Ce pilote fait partie du téléchargement gratuit de DB2 UDB Personal Developer.</p> <p>Si vous appliquez des kits de mise à jour, veuillez 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.</p>

Fournisseur	Base de données	Nom(s) de fichier(s)	Instructions de téléchargement
Sybase	Adaptive Server Enterprise (ASE) 12.5	jconn2.jar	Téléchargements Sybase (http://www.sybase.com/downloads) Pour télécharger ce pilote, sélectionnez jConnect for JDBC.
MySQL	MySQL 3.23	mysql-connector-java-2.0.14-bin.jar	Téléchargements 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

4 Redémarrez eDirectory.

5 Démarrez ConsoleOne ou iManager.

Importation d'un pilote préconfiguré

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

Base de données	Nom du fichier d'importation
Oracle 8i, 9i	JDBCOracleDirect.xml
	JDBCOracleIndirect.xml
Microsoft SQL Server 7/2000	JDBCMSQLDirect.xml
	JDBCMSQLIndirect.xml
IBM Universal Database (UDB) 7.2	JDBCDB2Direct.xml
	JDBCDB2Indirect.xml

Base de données	Nom du fichier d'importation
Sybase Adaptive Server Enterprise (ASE) 12.5	JDBCSybaseDirect.xml
	JDBCSybaseIndirect.xml
MySQL 3.23	JDBCMySQLIndirect.xml
Informix Dynamic Server (IDS) 9.3	JDBCInformixDirect.xml
	JDBCInformixIndirect.xml

Toutes les préconfigurations 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 qui présentent un intérêt. Vous pouvez adapter ces longueurs afin d'améliorer l'efficacité du stockage.
- ♦ Les champs de clé primaire comportent 8 chiffres. Cela doit suffire à des fins de test.
- ♦ 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>* (c'est-à-dire, « 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 ex. « t_emp_insert »)
- ♦ Les index se conforment à la convention de dénomination suivante : *i_<nom-table>_<numéro>* (par ex. « i_eventlog_1 »)

- ♦ Les colonnes d'identité et les objets de séquence peuvent placer 100 valeurs en mémoire cache. Cela doit suffire à des fins de test.
- ♦ 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 un pilote préconfiguré à partir du serveur et sélectionnez le fichier .xml.

Le fichier de configuration des pilotes est installé sur le serveur Web au moment de la configuration de 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 récapitulatif d'importation. Vérifiez que la configuration est correcte puis cliquez sur Terminer la présentation.

Les objets DirXML nécessaires ont été créés. Si vous n'avez pas défini les équivalences de sécurité ou exclu d'utilisateurs 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 le pilote préconfiguré par défaut.

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

Cette section contient des informations qui vous permettront d'effectuer les opérations suivantes :

- ♦ « Configuration des objets de base de données Oracle », page 36
- ♦ « Configuration des objets Microsoft SQL Server », page 37
- ♦ « Configuration des objets IBM DB2 », page 38
- ♦ « Configuration des objets Sybase », page 39
- ♦ « Configuration des objets MySQL », page 40
- ♦ « Configuration des objets Informix », page 41

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 à [Chapitre 8, « Désinstallation des objets Pilote et Base de données », page 103](#).

Configuration des objets de base de données Oracle

Cette section explique comment installer et dépanner les objets de base de données Oracle.

Par défaut, le mot de passe pour le compte « SYSTEM » est « MANAGER ».

Instructions d'installation

- 1 À partir d'un client Oracle, comme SQL Plus, loguez-vous en tant qu'utilisateur « SYSTEM ».
- 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
```

ou

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

- 3 Loguez-vous en tant qu'utilisateur « dirxml » avec le mot de passe « dirxml ».
- 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
```

ou

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

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 êtes connecté 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 Microsoft SQL Server

Cette section explique comment installer et dépanner les objets de base de données Microsoft SQL Server.

Par défaut, le compte « sa » n'est protégé par aucun mot de passe.

Instructions d'installation

- 1** Démarrez l'Analyseur de requêtes.
- 2** Loguez-vous à votre serveur sous le nom d'utilisateur « sa ».
- 3** Ouvrez et exécutez le premier script pour effectuer une synchronisation directe ou indirecte. Exemple :

```
tools\sql\mssql\direct\INSTALL_DIRECT_1.sql
```

ou

```
tools\sql\mssql\indirect\INSTALL_INDIRECT_1.sql
```

- 4** Utilisez l'utilisateur « dirxml » pour établir une autre connexion entre la base de données et votre serveur.
- 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
```

ou

```
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 êtes connecté 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.
- ♦ 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

Cette section explique comment installer et dépanner les objets de base de données IBM DB2 Universal Database (UDB).

Instructions d'installation

Pour une installation DB2 Universal Database, vous devez créer manuellement un compte utilisateur de base de données et une 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 6 à 8.

Pour installer les objets de base de données :

- 1** Créez un compte pour l'utilisateur « dirxml » avec le mot de passe « dirxml » dans Gest. 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`

ou

`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`

ou

`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 êtes connecté 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 et dépanner 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`.

Par défaut, le compte « sa » n'est protégé par aucun mot de passe.

Instructions d'installation

- 1 À partir d'un client Sybase, comme isql, exécutez le premier script d'installation en vue d'une synchronisation directe ou indirecte :

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

ou

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

- 2 Exécutez le deuxième script d'installation en vue d'une synchronisation directe ou indirecte :

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

ou

```
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.

Configuration des objets MySQL

Cette section explique comment installer et dépanner les objets de base de données MySQL.

Par défaut, le compte « root » n'est protégé par aucun mot de passe.

Instructions d'installation

- 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.

- 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

Cette section explique comment installer et dépanner les objets de base de données Informix Dynamic Server (IDS).

Par défaut, le compte « informix » est protégé par le mot de passe « informix ».

Instructions d'installation

- 1** Démarrez l'Éditeur SQL.
- 2** Connectez-vous à votre serveur sous le nom d'utilisateur « informix ».
- 3** Ouvrez et exécutez le premier script pour effectuer une synchronisation directe ou indirecte. Exemple :

```
tools\sql\informix\direct\INSTALL_DIRECT_1.sql
```

ou

```
tools\sql\informix\indirect\INSTALL_INDIRECT_1.sql
```

- 4** Utilisez l'utilisateur « dirxml » pour établir une autre connexion entre la base de données et votre serveur.
- 5** Ouvrez et exécutez le deuxième script d'installation pour effectuer une synchronisation directe ou indirecte. Exemple :

```
tools\sql\informix\direct\INSTALL_DIRECT_2.sql
```

ou

```
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 êtes connecté 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.
- ♦ 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. Cette section contient les informations suivantes :

Conditions requises pour la mise à niveau

Vous devez effectuer une mise à niveau vers la version 1.5 avant la version 1.6.

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 pilotes préconfigurés.
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 scripts.
- 6** Rétablissez les valeurs antérieures des options de démarrage des pilotes.
- 7** Redémarrez les pilotes.

Activation du pilote

DirXML et les pilotes DirXML doivent être activés dans les 90 jours qui suivent l'installation. S'ils ne le sont pas, ils seront arrêtés. Pendant cette période de 90 jours (ou ultérieurement), vous pouvez activer les produits DirXML dans le cadre d'une licence complète.

Pour activer le pilote, vous devez :

- ♦ Acquérir les licences DirXML
- ♦ Créer une requête d'activation de produit
- ♦ Soumettre la requête d'activation de produit
- ♦ Installer la référence d'activation de produit envoyée par Novell

Pour plus d'informations sur l'exécution de ces tâches, reportez-vous à [Activation de produit DirXML \(http://www.novell.com/documentation/french/dirxml11a/index.html\)](http://www.novell.com/documentation/french/dirxml11a/index.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 ConsoleOne®.

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

Nom du paramètre	Exemple de valeur de configuration	Valeur par défaut	Champ obligatoire
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. Ce compte 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, ce compte 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 champ du 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 liste des exemples d'URL pour les pilotes JDBC 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 l'identificateur de base ou de source de données DIRXML ou dirxml.

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

Pilote de fabricant tiers	Exemple de syntaxe d'URL JDBC
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 5.5	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 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 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 ouvrez 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 séries :

- ♦ 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.JDBCTestConnectionTester	(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>

Nom du paramètre	Exemple de valeur de configuration	Valeur par défaut	Obligatoire	Balise
Utiliser des noms de table qualifiés ?	non	non	non	<qualified-table-names>
Gérer les résultats des instructions ?	oui	oui	non	<handle-stmt-results>
Chaîne d'initialisation de la connexion	USE dirxml		non	<connection-init>

Nom de classe du pilote JDBC tiers

Le nom de classe du pilote JDBC tiers est un paramètre obligatoire avec distinction majuscules/minuscules. Il désigne le nom de classe qualifié complet du pilote JDBC tiers que vous utilisez pour communiquer avec votre base de données. Le tableau suivant donne des exemples de nom de classe du pilote :

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 correspond au 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 tables parent (ou des noms de classe de base de données logiques) à synchroniser. 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. 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 certaines bases de données.

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. Sinon, pour des performances maximales du pilote, Novell recommande 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, Novell recommande 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 certaines bases de données. 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 valeur « oui » attribuée à ce paramètre entraîne un ralentissement des performances du pilote. Ce paramètre ne doit prendre la valeur « oui » que lorsque les 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

Le niveau d'isolation de transaction par défaut est un paramètre facultatif sans distinction majuscules/minuscules, qui comporte quatre valeurs possibles en fonction des 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 ». Novell recommande 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/docs/books/tutorial/jdbc/basics/transactions.html>)

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. Pour plus de détails, reportez-vous à [Annexe B, « Méthodes java.sql.DatabaseMetaData », page 109](#). Lorsque la valeur « non » (par défaut) est définie pour ce paramètre, le pilote appelle les méthodes de métadonnées obligatoires et facultatives. Les méthodes de métadonnées facultatives sont requises pour la synchronisation des attributs à valeurs multiples et référentiels.

La valeur « oui » définie pour cette valeur entraîne une amélioration du temps de démarrage du pilote.

Utiliser des noms de table qualifiés

Ce paramètre a été désapprouvé et n'a aucun effet. Tous les noms de table sont désormais des noms qualifiés de schéma dans la mesure où un nom de schéma est disponible.

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. Trois valeurs sont possibles : « none », « single », « multiple » (valeur par défaut).

Pour des raisons de compatibilité amont, la valeur « yes » équivaut à « multiple » et la valeur « no », à « none ».

Pour le pilote ODBC de Microsoft, Oracle ou Informix, vous devez attribuer la valeur « single » à ce paramètre. Pour les autres bases de données ou pilotes tiers, Novell recommande 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. Il convient de 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.

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 ou non	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 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 <add> (ajouter), le canal Abonné utilise des valeurs de clé primaire pour créer des associations DirXML. Ce paramètre indique comment le canal Abonné obtient les valeurs de clé primaire nécessaires pour générer une valeur 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) : 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 cette première méthode 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 soit disponible. Pour les colonnes de type numérique, le pilote utilise une simple fonction (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-procédure-stockée>")*, où *<signature-procédure-stockée> = <nom-procédure>(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. 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. Ce paramètre est ignoré lorsque la première méthode de génération de la clé primaire est utilisée.

La valeur par défaut est « avant ».

Méthode 2 : 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 : Avec la valeur «avant», les procédures/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/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 la table ou la vue a bien été mise à jour. Si ce paramètre a la valeur « oui » et qu'une table ou une vue n'est pas mise 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.

Avec le pilote BEA Weblogic jDriver pour Microsoft SQL Server 7/2000, il est sans doute 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 le pilote.

Configuration de l'objet Éditeur

Le tableau suivant liste 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>

Nom du paramètre	Exemple de valeur de configuration	Valeur par défaut	Obligatoire	Balise
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 rechercher des événements de base de données. 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 sur certaines bases de données. 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 peut 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. La valeur par défaut est 10 secondes.

Il est recommandé de ne pas définir de valeur inférieure à dixsecondes.

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 30 secondes.

Il est recommandé de ne pas définir de valeur inférieure à dixsecondes.

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. Cette option est utile si les déclencheurs du canal Éditeur ne sont pas optimisés. L'égalité est déterminée par une opération de comparaison de chaînes avec distinction majuscules/minuscules. Les valeurs vides sont considérées comme égales. La valeur par défaut est « non ».

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. En revanche, il traite le champ STATUS d'une ligne en y indiquant l'état « S » (Réussi). Les lignes dont le traitement échoue restent dans la table de consignation des événements quelle que soit la valeur de ce paramètre. La valeur par défaut est « oui ».

La valeur « oui » 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 être écrites dans une table miroir.

Définition des propriétés de l'ensemble de pilotes

Pour afficher les résultats de débogage de ce pilote, vous devez attribuer une valeur d'attribut DirXML-Driver TraceLevel 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 manuel [DirXML Administration Guide \(Guide d'administration de DirXML\)](http://www.novell.com/documentation) (<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 consignment
- ♦ 4 = verbeux
- ♦ 5 = API JDBC (méthode, arguments, valeurs renvoyées, etc.)
- ♦ 6 = pilote JDBC/ODBC d'un fabricant tiers

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

Configuration de pilotes de fabricants tiers

Les instructions suivantes vous aideront à configurer des pilotes tiers. Pour des instructions de configuration particulières, reportez-vous à la documentation de votre 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 DirXML® pour JDBC (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 \(http://java.sun.com/products/jdk/1.1/docs/guide/intl/encoding.doc.html\)](http://java.sun.com/products/jdk/1.1/docs/guide/intl/encoding.doc.html).

Le tableau suivant liste 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
Réutiliser les instructions ?	non
Utiliser les transactions manuelles ?	non (non recommandé)
Utiliser la connexion simple	oui
Niveau d'isolation de transaction par défaut	non pris en charge (non recommandé)
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.

Cette section contient les informations suivantes :

- ♦ « [Assignation de schéma](#) », page 63
- ♦ « [Table de consignation des événements](#) », page 73
- ♦ « [Utilisation du langage SQL dans des événements XML](#) », page 80

Assignation de schéma

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

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

Classes de base de données logiques

Par classe de base de données logique, on entend l'ensemble des tables et 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
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. Elle est assignée à la classe eDirectory User.

```
CREATE TABLE DIRXML.EMP
(
    EMPNO NUMERIC(8) NOT NULL,
    FNAME VARCHAR(64),
    LNAME VARCHAR(64),
    PWDMINLEN NUMERIC(4),

    CONSTRAINT PK_EMPNO PRIMARY KEY (EMPNO)
);
```



```

CREATE TABLE DIRXML.PHONE
(
    EMPNO NUMERIC(8) NOT NULL,
    PHONE VARCHAR(64),

    CONSTRAINT FK_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>

```

Table 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_EMPNO PRIMARY KEY (EMPNO)
);

```

Tableau 1 Exemples de données de table DIRXML.EMP

EMPNO	FNAME	LNAME
1	Jean	Untel

L'association résultante pour cette ligne serait la suivante :

EMPNO=1,table=EMP,schema=DIRXML

Colonne 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 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 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 de ces colonnes comprennent des éléments *<remove-value>* qui sont transformés dans les feuilles de style en éléments *<remove-all-values>* suivis d'une série d'éléments *<add-value>*, à raison d'un élément par valeur ajoutée.

Table 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.

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_EMPNO PRIMARY KEY (EMPNO)
    );

CREATE TABLE DIRXML.PHONE
(
    EMPNO NUMERIC(8) NOT NULL,
    PHONE VARCHAR(64),

    CONSTRAINT FK_EMPNO FOREIGN KEY (EMPNO) REFERENCES
    EMP (EMPNO)
);

```

Les tables enfant peuvent contenir plusieurs valeurs et conviennent donc parfaitement à l'assignation d'attributs eDirectory à valeurs multiples. Chaque attribut à valeurs multiples doit être assigné à une table enfant différente. Par exemple, l'attribut eDirectory Telephone Number (Numéro de téléphone) est assigné à la colonne PHONENO de la table enfant PHONE.

Tableau 2 Exemple de données de table enfant DIRXML.PHONE

EMPNO	PHONENO
1	111-1111
2	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 du nom de la table enfant (par exemple, PHONE.PHONENO).

Attributs référentiels

Il s'agit d'établir des relations référentielles synchronisées entre des classes de base de données logiques à l'aide de plusieurs contraintes de clé étrangère. Dans l'exemple suivant, deux classes de base de données logiques, USERS et GROUPS, sont reliées par l'intermédiaire d'une table enfant unique, MEMBERS.

```

CREATE TABLE USERS
(
    IDU NUMBER(8) NOT NULL,
    LNAME VARCHAR(64) NOT NULL,

    CONSTRAINT PK_USERS_IDU PRIMARY KEY (IDU)
);

```

```

CREATE TABLE GROUPS
(
    IDG NUMBER(8) NOT NULL,

    CONSTRAINT PK_GROUPS_IDG PRIMARY KEY (IDG)
);

CREATE TABLE MEMBERS
(
    IDG NUMBER(8) NOT NULL,
    MEMBER NUMBER(8) NOT NULL,

    CONSTRAINT FK_MEMBERS_IDG FOREIGN KEY (IDG) REFERENCES
    GROUPS (IDG) ,
    CONSTRAINT FK_MEMBERS_MEMBER FOREIGN KEY (MEMBER)
    REFERENCES USERS (IDU)
);

<rule name="Mapping Rule">
    <attr-name-map>
        <class-name>
            <nds-name>User</nds-name>
            <app-name>USERS</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>GROUPS</app-name>
        </class-name>
        <attr-name class-name="Group">
            <nds-name>Member</nds-name>
            <app-name>MEMBERS.MEMBER</app-name>
        </attr-name>
    </attr-name-map>
</rule>

```

Dans l'exemple ci-dessus, l'attribut MEMBERS.MEMBER est considéré comme faisant partie de la classe GROUPS parce que la colonne IDG figure avant la colonne MEMBER dans la table MEMBERS.

Dans l'exemple suivant, l'attribut MEMBER_OF.GROUP est considéré comme faisant partie de la classe USERS parce que la colonne IDU figure avant la colonne GROUP dans la table MEMBER_OF.

```

CREATE TABLE USERS
(
    IDU NUMBER(8) NOT NULL,
    LNAME VARCHAR(64) NOT NULL,

    CONSTRAINT PK_USERS_IDU PRIMARY KEY (IDU)
);

CREATE TABLE GROUPS
(
    IDG NUMBER(8) NOT NULL,

    CONSTRAINT PK_GROUPS_IDG PRIMARY KEY (IDG)
);

CREATE TABLE MEMBER_OF
(
    IDU NUMBER(8) NOT NULL,
    GROUP NUMBER(8) NOT NULL,

    CONSTRAINT FK_MEMBER_OF_IDU FOREIGN KEY (IDU) REFERENCES
    USERS (IDU) ,
    CONSTRAINT FK_MEMBER_OF_GROUP FOREIGN KEY (GROUP)
    REFERENCES GROUPS (IDG)
);

<rule name="Mapping Rule">
    <attr-name-map>
        <class-name>
            <nds-name>User</nds-name>
            <app-name>USERS</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.GROUP</app-name>
        </attr-name>
        <class-name>
            <nds-name>Group</nds-name>
            <app-name>GROUPS</app-name>
        </class-name>
    </attr-name-map>
</rule>

```

En général, il est seulement nécessaire de synchroniser les attributs référentiels 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.

En pratique, pour la synchronisation des objets Utilisateur et Groupe, Novell recommande de synchroniser l'attribut Group Membership (Adhésion au groupe) de la classe Utilisateur au lieu de l'attribut Member (Membre) de la classe Groupe. En effet, en cas de synchronisation de l'attribut Member (Membre), il est possible que des événements relatifs aux utilisateurs non associés parviennent au canal Abonné. En cas de synchronisation de l'attribut Group Membership (Adhésion au groupe), seuls les groupes qui correspondent à des utilisateurs associés parviennent au canal Abonné.

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/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, 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, LAME, 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>
```

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.

En conséquence, les assignations d'attributs à valeur unique et à valeurs multiples sont traitées à l'identique.

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 DirXML 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 » « FAXNO », respectivement.

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 identificateurs de base de données réservés. L'ordre, le nombre et les types de données de ses colonnes doivent cependant rester constants.

Colonnes de la table de consignation des événements

- ♦ RECORD_ID

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. *Les valeurs d'état respectent la casse et doivent donc être en majuscules.*

- ♦ 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 peuvent également être regroupés en deux autres catégories : avec retour d'interrogation (5 à 8) et avec retour d'interrogation (1 à 4).

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**

L'ancienne valeur du champ. 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. 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. 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
insertion	<add>
mise à jour	<modify>
suppression	<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 EMP
(
    EMPNO NUMERIC(8) NOT NULL,
    FNAME VARCHAR2(64),
    PHOTO LONGRAW,

    CONSTRAINT PK_EMPNO PRIMARY KEY (EMPNO)
);
```

Tableau 3 Contenu initial d'EMP

EMPNO	FNAME	LNAME	PHOTO
1	Jacques	Untel	0xAAAA

Tableau 4 Contenu actuel d'EMP

EMPNO	FNAME	LNAME	PHOTO
1	Jean	Untel	0xAAAA

1. Insertion de champ

Tableau 5 Contenu de la table EVENTLOG

EVENT_TYPE	TABLE	TABLE_KEY	COLUMN_NAME	OLD_VALUE	NEW_VALUE
1	EMP	EMPNO=1	FNAME	NULL	Jacques

```

<add class-name="EMP">
  <association>EMPNO=1,table=EMP</association>
  <add-attr attr-name="FNAME">
    <value type="string">Jack</value>
  </add-attr>
</add>

```

2. Mise à jour de champ

Tableau 6 **Contenu de la table EVENTLOG**

EVENT_TYPE	TABLE	TABLE_KEY	COLUMN_NAME	OLD_VALUE	NEW_VALUE
2	EMP	EMPNO=1	FNAME	Jacques	Jean

```

<modify class-name="EMP">
  <association>EMPNO=1,table=EMP</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>

```

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

Tableau 7 **Contenu de la table EVENTLOG**

EVENT_TYPE	TABLE	TABLE_KEY	COLUMN_NAME	OLD_VALUE	NEW_VALUE
3	EMP	EMPNO=1	FNAME	Jacques	Jean

```

<modify class-name="EMP">
  <association>EMPNO=1,table=EMP</association>
  <modify-attr attr-name="FNAME">
    <remove-all-values/>
    <add-value>
      <value type="string">John</value>
    </add-value>
  </modify-attr>
</modify>

```

4. Suppression de ligne

Tableau 8 Contenu de la table EVENTLOG

EVENT_TYPE	TABLE	TABLE_KEY	COLUMN_NAME	OLD_VALUE	NEW_VALUE
4	EMP	EMPNO=1	NULL	NULL	NULL

```
<delete class-name="EMP">
  <association>EMPNO=1,table=EMP</association>
</delete>
```

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

Tableau 9 Contenu de la table EVENTLOG

EVENT_TYPE	TABLE	TABLE_KEY	COLUMN_NAME	OLD_VALUE	NEW_VALUE
5	EMP	EMPNO=1	NULL	NULL	NULL

```
<add class-name="EMP">
  <association>EMPNO=1,table=EMP</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">qqq=</value>
  </add-attr>
</add>
```

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

Tableau 10 Contenu de la table EVENTLOG

EVENT_TYPE	TABLE	TABLE_KEY	COLUMN_NAME	OLD_VALUE	NEW_VALUE
6	EMP	EMPNO=1	NULL	NULL	NULL

```
<modify class-name="EMP">
  <association>EMPNO=1,table=EMP</association>
  <modify-attr attr-name="FNAME">
    <remove-all-values/>
    <add-value>
      <value type="string">John</value>
    </add-value>
  </modify-attr>
</modify>
```

```

</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">qgo=</value>
  </add-value>
</modify-attr>
</modify>

```

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

Tableau 11 Contenu de la table EVENTLOG

EVENT_TYPE	TABLE	TABLE_KEY	COLUMN_NAME	OLD_VALUE	NEW_VALUE
7	EMP	EMPNO=1	PHOTO	NULL	NULL

```

<add class-name="EMP">
  <association>EMPNO=1,table=EMP</association>
  <add-attr attr-name="PHOTO">
    <value type="octet">qgo=</value>
  </add-attr>
</add>

```

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

Tableau 12 Contenu de la table EVENTLOG

EVENT_TYPE	TABLE	TABLE_KEY	COLUMN_NAME	OLD_VALUE	NEW_VALUE
8	EMP	EMPNO=1	PHOTO	NULL	NULL

```

<modify class-name="EMP">
  <association>EMPNO=1,table=EMP</association>
  <modify-attr attr-name="PHOTO">
    <remove-all-values/>
    <add-value>
      <value type="octet">qgo=</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 une hypothétique table vide appelée EMP. 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),

    CONSTRAINT PK_EMPNO PRIMARY KEY (EMPNO)
);
```

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 un événement 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*. *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 EMP SET FNAME = 'John' </jdbc:sql>
  </jdbc:statement>
</input>
```


Comme l'objet Abonné résout les événements <add> pour insérer des instructions, le code XML ci-dessous 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 DirXML, 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 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: { \$<nomchamp> } 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 <nomchamp> 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 `<add>` 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 `<add>`. 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 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>NOVELL</password>
  </add>
  <jdbc:statement>
    <jdbc:sql>CREATE USER DOE 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 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 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 FNAME = 'John' WHERE EMPNO = 1;
UPDATE DIRXML.EMP SET LNAME = 'Doe' 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 EMP SET FNAME = 'John' WHERE
      EMPNO = {$EMPNO}</jdbc:sql>
  </jdbc:statement>
</input>
```

Les données XML ci-dessus sont converties comme suit :

```
INSERT INTO EMP(LNAME) VALUES('Doe');
/* COMMIT; //implicit */

UPDATE 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 EMP SET FNAME = 'John' WHERE
      EMPNO = {$EMPNO}</jdbc:sql>
</jdbc:statement>
</input>

```

Le code XML ci-dessus est converti comme suit :

```

INSERT INTO EMP(LNAME) VALUES('Doe');
UPDATE 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 EMP SET FNAME = 'John'
            WHERE EMPNO = {$EMPNO}</jdbc:sql>
    </jdbc:statement>
</input>

```

Les données XML ci-dessus sont converties comme suit :

```

INSERT INTO EMP(LNAME) VALUES ('Doe');
UPDATE 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 oeuvre 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.

Novell recommande 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 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 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 à 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 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 pour 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>NOVELL</password>
  </add>
  <jdbc:statement jdbc:op-id="0">
    <jdbc:sql>CREATE USER DIRXML 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')
    </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 EMP SET FNAME = 'John'
      WHERE EMPNO = {$EMPNO}</jdbc:sql>
  </jdbc:statement>
  <jdbc:statement>
    <jdbc:sql>UPDATE EMP SET FNAME = 'John'
      WHERE EMPNO = {$EMPNO}</jdbc:sql>
  </jdbc:statement>
  <jdbc:statement>
    <jdbc:sql>UPDATE EMP SET FNAME = 'John'
      WHERE EMPNO = {$EMPNO}</jdbc:sql>
  </jdbc:statement>
  <jdbc:statement>
    <jdbc:sql>UPDATE EMP SET FNAME = 'John'
      WHERE EMPNO = {$EMPNO}</jdbc:sql>
  </jdbc:statement>
</input>
```


6

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 ? », page 94
- ♦ « Comment synchroniser des tables situées dans plusieurs schémas ? », page 94
- ♦ « Pourquoi le pilote ne traite-t-il pas les enregistrements dans le journal des événements ? », page 94
- ♦ « Le pilote peut-il gérer les comptes utilisateur de la base de données ? », page 95
- ♦ « Le pilote peut-il synchroniser les données de type binaire et chaîne en grande quantité ? », page 95
- ♦ « Pourquoi l'acheminement des données via le canal Éditeur est-il si lent ? », page 95
- ♦ « Le pilote peut-il synchroniser plusieurs classes ? », page 95
- ♦ « Pourquoi les colonnes de clé étrangère et de clé primaire doivent-elles porter le même nom ? », page 96
- ♦ « Le pilote prend-il en charge le codage SSL ? », page 96
- ♦ « Comment assigner des attributs à valeurs multiples à des champs de base de données à valeur unique ? », page 96
- ♦ « Pourquoi le pilote synchronise-t-il des chaînes incorrectes ? », page 96

Réponses aux questions fréquentes sur le pilote

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. À ce titre, il ignore les tables sans contrainte. 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 « Autoriser le retour en boucle ? » de l'objet Éditeur a la valeur « 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 les champs STATUS de l'enregistrement ont la valeur 'N' (nouveau). Les enregistrements dont les champs STATUS contiennent autre chose que 'N' ne seront pas traités. En outre, veuillez 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 80.

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 le journal des événements contient un grand nombre de lignes, il doit être indexé. 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, la valeur « no » 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 générale à ce problème est de placer le pilote et le pilote tiers à distance. Cette installation distante permet au pilote et au pilote tiers de s'exécuter localement sur le serveur de base de données. Toutes les données qui transitent par le réseau entre le moteur et le pilote seront alors chiffrées au format SSL.

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 72.

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 à la page [Character Encoding Values](http://java.sun.com/products/jdk/1.1/docs/guide/intl/encoding.doc.html) (<http://java.sun.com/products/jdk/1.1/docs/guide/intl/encoding.doc.html>) du site Web de Sun.

7

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.

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 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 endommagées, DirXML™ cesse de fonctionner. En conséquence, vous ne devez utiliser des opérations d'écriture qu'en cas de nécessité. Pour éviter toute altération involontaire d'associations, cet utilitaire crée un fichier ldif d'annulation pour toutes les opérations d'écriture.

Lisez les recommandations suivantes avant d'utiliser cet outil :

- ♦ 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™ 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, Novell® recommande 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 localement, il utilise une connexion non sécurisée. À ce titre, le serveur LDAP de eDirectory doit être temporairement configuré pour accepter des 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 : Novell recommande 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'outil

Cet utilitaire doit être exécuté une fois pour chaque instance du pilote installée sur le serveur cible. Procédez comme suit pour exécuter l'utilitaire.

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.

Exécution de l'utilitaire

- 1 Arrêtez le(s) pilote(s).
- 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,

- ♦ Remplacement du schéma « other » par le schéma « DIRXML »

oldRDN : schema=other

newRDN : schema=DIRXML

2. Remplacement du nom de table

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

- ♦ Remplacement de la table « EMP » par la table « emp » :

oldRDN : table=EMP

newRDN : table=emp

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,

- ♦ Remplacement de la colonne « other » par la colonne « EMPNO »

oldRDN : other=*

newRDN : EMPNO=*

8

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™, 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.

Les instructions suivantes permettent de supprimer un objet de eDirectory.

- 1** Dans 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 qui vous permettront d'effectuer les opérations suivantes :

- ♦ « Désinstallation des objets Oracle* », page 104
- ♦ « Désinstallation des objets Microsoft* SQL Server », page 104
- ♦ « Désinstallation des objets IBM* DB2 UDB », page 105
- ♦ « Désinstallation des objets Sybase* », page 105
- ♦ « Désinstallation des objets MySQL », page 106

Important : Novell recommande 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.

Désinstallation des objets Oracle*

Suivez les instructions ci-après pour désinstaller vos objets de base de données Oracle.

Par défaut, le compte « SYSTEM » est protégé par le mot de passe « MANAGER ».

- 1** À partir d'un client Oracle, comme SQL Plus, loguez-vous en tant qu'utilisateur « SYSTEM ».
- 2** Exécutez le script de désinstallation en vue d'une synchronisation directe ou indirecte. Exemple :

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

ou

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

Désinstallation des objets Microsoft* SQL Server

Suivez les instructions ci-après pour désinstaller vos objets de base de données Microsoft SQL Server.

Par défaut, le compte « sa » n'est protégé par aucun mot de passe.

- 1** Démarrez l'Analyseur de requêtes.
- 2** Loguez-vous à votre serveur sous le nom d'utilisateur « sa ».
- 3** Ouvrez et exécutez le script de désinstallation en vue d'une synchronisation directe ou indirecte. Exemple :

`tools\sql\mssql\direct\UNINSTALL_DIRECT.sql`

ou

`tools\sql\mssql\indirect\UNINSTALL_INDIRECT.sql`

Désinstallation des objets IBM* DB2 UDB

Suivez les instructions ci-après pour désinstaller vos objets IBM DB2 Universal Database (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`

ou

`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.

Le script de désinstallation ne détruit ni la base de données « dirxml », ni le compte utilisateur « dirxml ».

Désinstallation des objets Sybase*

Suivez les instructions ci-après pour désinstaller vos objets de base de données Sybase Adaptive Server Enterprise (ASE).

Par défaut, le compte « sa » n'est protégé par aucun mot de passe.

- 1 À partir d'un client Sybase, comme isql, exécutez le script de désinstallation en vue d'une synchronisation directe ou indirecte.

Exemple :

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

ou

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

Désinstallation des objets MySQL

Suivez les instructions ci-après pour désinstaller vos objets de base de données MySQL.

Par défaut, le compte « root » n'est protégé par aucun mot de passe.

- 1 À partir d'un client MySQL, comme mysql, exécutez le script de désinstallation pour une synchronisation indirecte. Exemple :

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

Désinstallation des objets Informix

Suivez les instructions ci-après pour désinstaller vos objets de base de données Informix Dynamic Server (IDS).

Par défaut, le compte « informix » est protégé par le mot de passe « informix ».

- 1 Démarrez l'Éditeur SQL.
- 2 Connectez-vous à votre serveur sous le nom d'utilisateur « informix ».
- 3 Exécutez le script de désinstallation en vue d'une synchronisation directe ou indirecte. Exemple :

```
tools\sql\informix\direct\UNINSTALL_DIRECT.sql
```

ou

```
tools\sql\informix\indirect\UNINSTALL_INDIRECT.sql
```

A

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 natifs d'une base de données dépend de cette dernière. 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`

B

Méthodes `java.sql.DatabaseMetaData`

Cette annexe liste 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 \(http://java.sun.com/products/jdk/1.2/docs/api/java/sql/DatabaseMetaData.html\)](http://java.sun.com/products/jdk/1.2/docs/api/java/sql/DatabaseMetaData.html).

Méthodes obligatoires :

- ♦ `java.sql.ResultSet getColumnns(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)`

C

Méthodes JDBC

Cette annexe liste les méthodes JDBC 1.0 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 annexe, 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)`

