IBM InfoSphere Federation Server

バージョン 10.1

フェデレーテッド・システムの管理ガイド

SA88-4827-00
(英文原典：SC19-3694-00)

IBM InfoSphere Federation Server

バージョン 10.1

フェデレーテッド・システムの管理ガイド

SA88-4827-00
(英文原典：SC19-3694-00)

お願い

本書および本書で紹介する製品をご使用になる前に、 495 ページの『特記事項および商標』 に記載されている情報をお読みください。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： IBM InfoSphere Federation Server
Version 10.1
Administration Guide for Federated
Systems

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2012.5

目次

第 1 章 フェデレーテッド・システム

フェデレーテッド・システム は、特殊なタイプの分散データベース管理システム (DBMS) です。フェデレーテッド・システムは、フェデレーテッド・サーバーとして作動する 1 つの DB2® インスタンス、フェデレーテッド・データベースとして機能する 1 つのデータベース、1 つ以上のデータ・ソース、およびこれらのデータベースやデータ・ソースにアクセスする複数のクライアント (ユーザーおよびアプリケーション) で構成されます。

フェデレーテッド・システムでは、単一の SQL ステートメントで、複数のデータ・ソースに対する分散要求を送信できます。例えば、単一の SQL ステートメントで、DB2 表、Oracle 表、および XML タグ付きファイル内にあるデータを結合することができます。次の図は、フェデレーテッド・システムのコンポーネントと、ユーザーがアクセスできるデータ・ソースのサンプルを示しています。

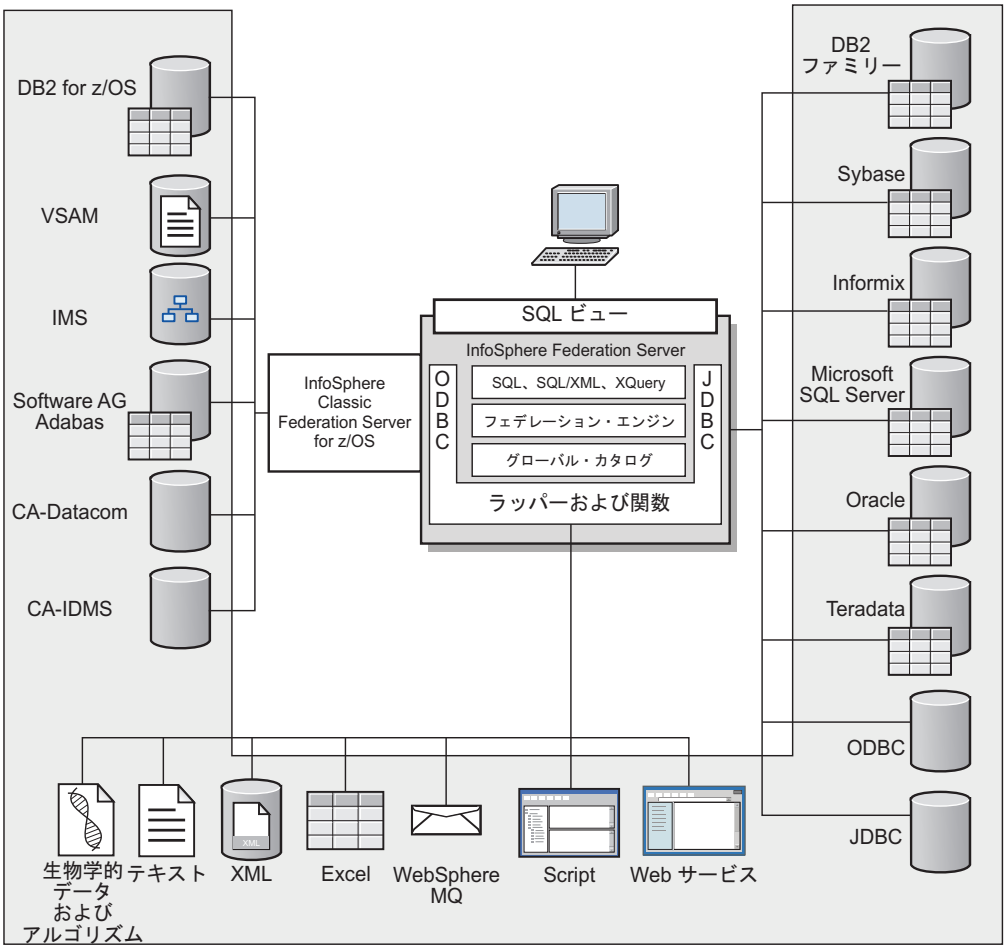


図 1. フェデレーテッド・システムのコンポーネント

フェデレーテッド・システムには、以下を行う能力があります。

- ローカル表のデータとリモート・データ・ソースのデータを、すべてフェデレーテッド・データベースにローカルに保管されているデータであるかのように相関します。
- リレーショナル・データ・ソースのデータを、フェデレーテッド・データベースに保管されているデータであるかのように更新します。
- リレーショナル・データ・ソースとの間でデータを相互に移動します。
- 要求を処理のためデータ・ソースに送信することにより、データ・ソースの処理能力を利用します。
- 分散要求を部分的にフェデレーテッド・サーバーで処理することにより、データ・ソースでの SQL の限界を補います。

サポートされるデータ・ソース

フェデレーテッド・システムを使用してアクセスできるデータ・ソースはたくさんあります。

IBM® InfoSphere® Federation Server では、以下の表に示されたデータ・ソースがサポートされます。

サポートされるデータ・ソースの最新情報については、Web 上のデータ・ソース要件のページを参照してください。

表 1. サポートされるデータ・ソース

データ・ソース	サポートされるバージョン
BioRS	5.2、5.3
CA-Datacom	10, 11
CA-IDMS	16, 17
DB2 for Linux, UNIX, and Windows	8.1.x、8.2.x、9.1、9.5、9.7、9.8、10.1
DB2 for z/OS®	7.x、8.x、9.x、10x
DB2 for System i®	5.2、5.3、5.4、6.1
DB2 Server for VSE and VM	7.2、7.4
フラット・ファイル	
IMS™	10.1, 11.1
Informix®	Informix XPS 8.50、8.51 および Informix IDS IDS 7.31、IDS 9.40、IDS 10.0、11.1、11.5、11.7
JDBC	3.0 以降
Microsoft Excel	2000、2002、2003、2007
Microsoft SQL Server	Microsoft SQL Server 2000/SP4、2005、2008、2008R2
MQ	MQ7
Netezza	4.6, 5.0, 6.0
ODBC*	3.0
OLE DB	2.7、2.8
Oracle	10g, 10gR2, 11g, 11gR1, 11gR2
Software AG Adabas	8.1.3, 8.1.4, 8.2.2
Sybase	Sybase ASE 12.5、15.0、15.5

表 1. サポートされるデータ・ソース (続き)

データ・ソース	サポートされるバージョン
Teradata	2.5, 2.6, 12, 13
VSAM	• CICS® Transaction Server V3.1、V3.2、V4.1 • サポートされる z/OS レベルのネイティブ VSAM アクセス • サポートされる z/OS レベルの DFSMS トランザクション VSAM サービス (DFSMStys) でのトランザクション VSAM アクセス
Web サービス	WSDL 1.0、1.1 SOAP 1.0、1.1
XML	XML1.0、XML1.1

* ODBC は、さまざまなデータ・ソースへのアクセスに使用できますが、特に RedBrick 6.20.UC5 および 6.3、InfoSphere Classic Federation Server for z/OS V8.2 以上、および Netezza にアクセスするために使用できます。

フェデレーテッド・サーバー

フェデレーテッド・システム内の DB2 サーバーのことを、フェデレーテッド・サーバーといいます。 DB2 インスタンスであればいくつでも、フェデレーテッド・サーバーとして機能するように構成することができます。既存の DB2 インスタンスをフェデレーテッド・サーバーとして使用したり、特にフェデレーテッド・システム専用として新しく作成したりできます。

フェデレーテッド・システムを管理する DB2 インスタンスのことをサーバー と呼びますが、それはこのインスタンスがエンド・ユーザーおよびクライアント・アプリケーションからの要求に応答するからです。フェデレーテッド・サーバーは受信した要求の各部を頻繁にデータ・ソースに送信して処理させます。プッシュダウン操作は、リモート側で処理される操作です。フェデレーテッド・システムを管理する DB2 インスタンスは、要求をデータ・ソースにプッシュダウンする場合はクライアントとして働きますが、フェデレーテッド・サーバー と呼ばれます。

その他のアプリケーション・サーバーと同様に、フェデレーテッド・サーバーはデータベース・マネージャー・インスタンスです。アプリケーション・プロセスはフェデレーテッド・サーバーに接続し、フェデレーテッド・サーバー内のデータベースに要求をサブミットします。ただし、次の 2 つの主要な機能により、その他のアプリケーション・サーバーとは区別されます。

- フェデレーテッド・サーバーは、部分的または全面的にデータ・ソース向けの要求を受信するように構成されています。フェデレーテッド・サーバーは、これらの要求をデータ・ソースに配布します。
- その他のアプリケーション・サーバーと同様に、フェデレーテッド・サーバーは DRDA® 通信プロトコル (over TCP/IP) を使用して、 DB2 ファミリーのインスタンスと通信します。ただし、他のアプリケーション・サーバーと異なり、フェデレーテッド・サーバーはデータ・ソースのネイティブ・クライアントを使用して、データ・ソースにアクセスします。例えば、フェデレーテッド・サーバーは Sybase Open Client を使用して Sybase データ・ソースにアクセスし、 Microsoft SQL Server ODBC ドライバーを使用して Microsoft SQL Server データ・ソースにアクセスします。

データ・ソースとは?

フェデレーテッド・システムでは、リレーショナル・データベース (Oracle または Sybase など) または非リレーショナル・データ・ソース (XML タグ付きファイルなど) をデータ・ソース にすることができます。

特定のデータ・ソースを介することで、他のデータ・ソースにアクセスすることも可能です。例えば、ODBC ラッパーを使用して、DB2 for z/OS、IMS、CA-IDMS、CA-Datacom、Software AG Adabas、および VSAM などの、IBM InfoSphere Classic Federation Server for z/OS のデータ・ソースにアクセスできます。

データ・ソースへのアクセスに使用される方式つまりプロトコルは、データ・ソースのタイプによって異なります。例えば、DRDA は DB2 for z/OS のデータ・ソースにアクセスするために使用されます。

データ・ソースはオートノマス (自律的) です。例えば、フェデレーテッド・サーバーが Oracle データ・ソースに照会を送信しているときに、その同じデータ・ソースに Oracle アプリケーションがアクセスしてもかまいません。保全性およびロッキング制約が損なわれない限り、フェデレーテッド・システムが他のデータ・ソースへのアクセスを独占または制限することはありません。

フェデレーテッド・データベース

エンド・ユーザーおよびクライアント・アプリケーションにとって、データ・ソースは、 DB2 データベース・システムの単一の集合データベースに見えます。ユーザーとアプリケーションは、フェデレーテッド・サーバーが管理するフェデレーテッド・データベース とやり取りを行います。

フェデレーテッド・データベースにはデータの関する情報を保管するシステム・カタログが入っています。このフェデレーテッド・データベースのシステム・カタログには、データ・ソースとその特性を示すカタログ項目が入っています。フェデレーテッド・サーバーは、フェデレーテッド・データベース・システム・カタログに保管された情報およびデータ・ソース・ラッパーを検討した上で、SQL ステートメントを処理する最善のプランを決めます。

フェデレーテッド・システムは、データ・ソースからのデータがフェデレーテッド・データベース内の通常のリレーショナルの表またはビューであるかのように、SQL ステートメントを処理します。その結果、次のようになります。

- フェデレーテッド・システムはリレーショナル・データを非リレーショナルのフォーマットのデータと結合することができます。データ・ソースが異なる SQL ダイアレクトを使用していたり、あるいは SQL をまったくサポートしていなくても、あてはまります。
- フェデレーテッド・データベースの特性とデータ・ソースの特性に相違がある場合、フェデレーテッド・データベースの特性が優先されます。照会の結果は DB2 セマンティクスに準拠します。照会の結果の計算に他の DB2 以外のデータ・ソースからのデータが使用される場合でも同様です。

  例:

– フェデレーテッド・サーバーが使用するコード・ページは、データ・ソースが使用するコード・ページと異なります。この場合、データ・ソースの文字データは、フェデレーテッド・ユーザーに戻される際、フェデレーテッド・データベースで使用されているコード・ページに基づいて変換されます。
– フェデレーテッド・サーバーが使用する照合シーケンスは、データ・ソースが使用する照合シーケンスと異なります。この場合、文字データに対するソート操作はすべて、データ・ソースではなくフェデレーテッド・サーバーで行われます。

ラッパーおよびラッパー・モジュール

ラッパー とは、フェデレーテッド・データベースがデータ・ソースと対話するためのメカニズムです。フェデレーテッド・データベースは、ライブラリーに保管されたルーチン (ラッパー・モジュール という) を使用してラッパーをインプリメントします。

これらのルーチンを使用することで、フェデレーテッド・データベースは、データ・ソースへの接続やデータ・ソースからのデータ検索の繰り返しなどの操作を実行できます。通常、フェデレーテッド・インスタンスの所有者は、CREATE WRAPPER ステートメントを使用して、ラッパーをフェデレーテッド・データベースに登録します。 DB2_FENCED オプションを使用すると、ラッパーを fenced またはトラステッドとして登録することができます。

ラッパーは、アクセスするデータ・ソースのタイプごとに 1 つ作成します。例えば、3 つの DB2 for z/OS データベース表、1 つの DB2 for System i 表、2 つの Informix 表、および 1 つの Informix ビューにアクセスするとします。このとき作成する必要があるのは、DB2 データ・ソース・オブジェクト用のラッパーを 1 つと、 Informix データ・ソース・オブジェクト用のラッパーを 1 つです。これらのラッパーをフェデレーテッド・データベースに登録すれば、すぐにそれらのラッパーを使用して対応するデータ・ソースから他のオブジェクトにアクセスすることが可能になります。例えば、DRDA ラッパーを使用すれば、すべての DB2 ファミリーのデータ・ソース・オブジェクト (DB2 Database for Linux, UNIX, and Windows 、DB2 for z/OS 、DB2 for System i、および DB2 Server for VM and VSE) からのデータ・ソースにアクセスできます。

各データ・ソース・オブジェクトを特定して識別 (名前やロケーションなど) するには、サーバー定義とニックネームを使用します。

ラッパーは多くの作業を行います。そのいくつかは次のようなものです。

- データ・ソースに接続します。ラッパーは、データ・ソースの標準の接続 API を使用します。
- データ・ソースに照会をサブミットします。
  – SQL をサポートするデータ・ソースの場合、照会は SQL でサブミットされます。
  – SQL をサポートしないデータ・ソースの場合、照会は、ソースに固有の照会言語に、または一連のソース API 呼び出しに変換されます。
- データ・ソースから結果セットを受け取ります。ラッパーは、データ・ソースの標準 API を使用して、結果セットを受信します。

- データ・ソースのデフォルトのデータ・タイプ・マッピングについてのフェデレーテッド・データベースの照会に応答します。ラッパーには、データ・ソース・オブジェクトにニックネームを作成する時に使用される、デフォルトのタイプ・マッピングが入っています。リレーショナル・ラッパーの場合、ユーザーが作成するデータ・タイプ・マッピングは、デフォルトのデータ・タイプ・マッピングをオーバーライドします。ユーザー定義のデータ・タイプ・マッピングは、グローバル・カタログに保管されます。
- データ・ソースのデフォルトの関数マッピングについてのフェデレーテッド・データベースの照会に応答します。フェデレーテッド・データベースは、照会の計画で使用するためのデータ・タイプのマッピング情報を必要とします。ラッパーには、DB2 関数がデータ・ソースの関数と対応付けられるかどうか、またどのように関数が対応付けられるかを、フェデレーテッド・データベースが判断する際に必要となる情報が含まれています。この情報は、データ・ソースが照会操作を実行できるかどうかを判断するために、SQL コンパイラーにより使用されます。リレーショナル・ラッパーの場合、ユーザーが作成する関数マッピングは、デフォルトの関数タイプ・マッピングをオーバーライドします。ユーザー定義の関数マッピングは、グローバル・カタログに保管されます。

ラッパー・オプション は、ラッパーを構成するため、またはIBM InfoSphere Federation Server がどのようにラッパーを使用するかを定義するために使用されます。

フェデレーテッド・システムおよび DB2 pureScale

DB2 for Linux,UNIX, and Windows では、バージョン 9.8 の DB2 pureScale® フィーチャーを導入しました。バージョン 10 では、Federation Server の機能が DB2 pureScale 環境に統合されました。

DB2 pureScale と共にフェデレーションを使用すると、DB2 pureScale フィーチャーにより実現する可用性とスケーラビリティーを活用することができます。pureScale フィーチャーの詳細については、Introduction to the IBM DB2 pureCluster Feature を参照してください。

フェデレーション機能は、DB2 pureScale インスタンスのどのメンバーに対しても有効です。これらの機能には、挿入、更新、および削除の操作に加えてストアード・プロシージャーなどフェデレーションがサポートする大半の機能が含まれています。

自動クライアント・リルートおよびワークロード・バランシングを使用すると、DB2 pureScale インスタンスのすべてのメンバーがフェデレーテッド・ステートメントを実行できます。クライアント・ライブラリーをすべての DB2 pureScale メンバーにインストールすることが重要です。

フェデレーテッド・システムと対話する方法

フェデレーテッド・データベースは DB2 ファミリーのデータベースであるため、さまざまな方法でこのデータベースと対話することができます。

フェデレーテッド・システムと対話するには、以下のいずれかの方法を使用できます。

- DB2 コマンド行プロセッサー (CLP)
- アプリケーション・プログラム
- DB2 ファミリー・ツール
- Web サービス・プロバイダー

フェデレーテッド・データベースの資料で説明されている手順には、DB2 コマンド行プロセッサーに入力するコマンドおよび SQL ステートメントが示されています。資料が説明しているのは、IBM Data Studio GUI を使用してタスクを実行するタイミングです。

IBM Data Studio

IBM Data Studio は統合的なデータ管理ソリューションであり、 IBM InfoSphere Federation Server の構成と管理を目的として、作成と管理を行うフェデレーテッド・オブジェクトを操作するために使用できます。

IBM Data Studio は、以下の 3 つのコンポーネントで構成されています。

IBM Data Studio 管理クライアント
データベース管理および非 Java プログラミング言語のルーチン開発のための、占有スペースの少ない、スタンドアロン統合開発環境を提供します。

IBM Data Studio フル・クライアント
データベース管理、ルーチン開発、および Java アプリケーション開発のための統合開発環境を提供します。このコンポーネントは、他の IBM ソフトウェア製品と共にインストールして、共通環境を共有することが可能です。

IBM Data Studio Web コンソール
ヘルス・モニターと可用性モニター、ジョブ作成、およびデータベース管理の機能が備わった Web ベースのツールを提供します。

以前にコントロール・センターのツールを使用していた場合は、推奨される IBM Data Studio ツールおよび IBM InfoSphere Optim™ ツールと、コントロール・センター・ツールの間のマッピングを検討してください。推奨されるツール対コントロール・センター・ツールの表を参照してください。

DB2 コマンド行プロセッサー (CLP)

フェデレーテッド・システムのセットアップ、構成、調整、および保守に必要なすべての作業は、DB2 コマンド行プロセッサーから実行することができます。

例えば、DB2 コマンド行プロセッサーを使用して、以下のタスクを実行することができます。

- ユーザー定義データ・タイプ・マッピングを作成、変更、またはドロップする
- ユーザー定義関数マッピングを作成、変更、またはドロップする

アプリケーション・プログラム

アプリケーションに、フェデレーテッド・データを使用するための特別なコーディングは必要ありません。アプリケーションは、他の DB2 クライアント・アプリケ

ーションとまったく同様にシステムにアクセスします。アプリケーションは、フェデレーテッド・サーバー内にあるフェデレーテッド・データベースとインターフェースをとります。

データ・ソースからデータを入手するために、アプリケーションは DB2 SQL で書かれた照会をフェデレーテッド・データベースにサブミットします。するとフェデレーテッド・データベースは、その照会を該当するデータ・ソースに配布し、要求されたデータを収集し、そのデータをアプリケーションに戻します。ただし、フェデレーテッド・データベースはニックネームを使用してデータ・ソースとやりとりするので、ユーザーは次のことに注意する必要があります。

- ニックネームを使用する場合の SQL の制約事項
- ニックネームが付けられたオブジェクトの操作方法

DB2 ファミリー・ツール

フェデレーテッド・データベースと対話するために、ホストおよびミッドレンジ・ツールを使用することができます。

フェデレーテッド・データベースとの対話で使用できるホストおよびミッドレンジ・ツールには、次のものがあります。

- DB2 for z/OS および DB2 for Linux, UNIX, and Windows のために、パフォーマンスおよび管理を機能強化する DB2 ツール。
- 対話式 SQL (STRSQL) (DB2 for System i の場合)

Web サービス・プロバイダー

Web サービス・プロバイダーを経由して、フェデレーテッド・データベースと対話することができます。

フェデレーテッド・システムでは、Web サービス・ラッパーにより、Web サービスを呼び出すニックネームとビューに関する SQL ステートメントを使用して、Web サービスにアクセスできます。 Web サービス・ラッパーを作成し、さらに Web サービスへの入力を指定するニックネームと、SELECT ステートメントを使って Web サービスからの出力にアクセスするニックネームを作成することができます。

フェデレーテッド・データベース・システム・カタログ

フェデレーテッド・データベース・システム・カタログには、フェデレーテッド・データベース内のオブジェクトの情報と、データ・ソース側のオブジェクトの情報が入っています。

フェデレーテッド・データベース内のカタログは、フェデレーテッド・システム全体についての情報が含まれているため、グローバル・カタログ と呼びます。 DB2 照会オプティマイザーは、グローバル・カタログ内の情報およびデータ・ソース・ラッパーを使用して、SQL ステートメントを処理する最善の方法を計画します。グローバル・カタログに保管される情報には、リモートとローカルの情報、例えば列名、列のデータ・タイプ、列のデフォルト値、索引情報、および統計情報などが含まれます。

リモート・カタログ情報は、データ・ソースが使用する情報または名前です。ローカル・カタログ情報は、フェデレーテッド・データベースが使用する情報または名前です。例えば、*EMPNO* という名前の列を持つリモート表があるとします。グローバル・カタログには、このリモートの列名が *EMPNO* として保管されます。別の名前を指定しないかぎり、ローカルの列名は *EMPNO* として保管されます。ローカルの列名を *Employee_Number* に変更することができます。この列を含む照会をサブミットするユーザーは、照会の中で *EMPNO* ではなく *Employee_Number* を使用します。データ・ソース列のローカル名を変更するには、ALTER NICKNAME ステートメントを使用します。

リレーショナル・データ・ソースおよび非リレーショナル・データ・ソースの場合、グローバル・カタログに保管される情報にはリモートとローカルの両方の情報が含まれます。

グローバル・カタログに保管されたデータ・ソース表の情報を見るには、フェデレーテッド・データベース内の
SYSCAT.TABLES、SYSCAT.NICKNAMES、SYSCAT.TABOPTIONS、
SYSCAT.INDEXES、SYSCAT.INDEXOPTIONS、SYSCAT.COLUMNS、および
SYSCAT.COLOPTIONS カタログ・ビューを照会してください。

グローバル・カタログには、データ・ソースについてのその他の情報も入っています。例えば、フェデレーテッド・サーバーがデータ・ソースに接続したり、フェデレーテッド・ユーザー許可をデータ・ソースのユーザー権限にマップするために使用する情報が含まれます。グローバル・カタログには、明示的に設定されたデータ・ソースの属性 (サーバー・オプションなど) が入っています。

SQL コンパイラー

DB2 SQL コンパイラーは、照会の処理に役立つ情報を収集します。

データ・ソースからデータを入手するため、ユーザーおよびアプリケーションは SQL の照会をフェデレーテッド・データベースにサブミットします。照会をサブミットすると、DB2 SQL コンパイラーはグローバル・カタログ内の情報およびデータ・ソース・ラッパーを検討し、照会の処理に役立てます。この情報には、データ・ソースへの接続に関する情報、サーバー情報、マッピング、索引情報、および処理統計が含まれます。

照会オプティマイザー

SQL コンパイラー処理の一部として、照会オプティマイザー は照会を分析します。コンパイラーは、アクセス・プラン と呼ばれる、照会を処理するための代替ストラテジーを作成します。

アクセス・プランは、照会を次のように処理することを要求します。

- 照会をデータ・ソースが処理する
- 照会をフェデレーテッド・サーバーが処理する
- 照会の一部をデータ・ソースが処理し、一部をフェデレーテッド・サーバーが処理する

照会オプティマイザーは、主にデータ・ソースの能力およびデータに関する情報を基にアクセス・プランを評価します。この情報はラッパーとグローバル・カタログにあります。照会オプティマイザーは照会を照会フラグメントと呼ばれるセグメントに分解します。通常、照会フラグメントをデータ・ソースにプッシュダウンした方が、より効率的です (データ・ソースがフラグメントを処理できる場合)。しかし、照会オプティマイザーは次のような他の要素も考慮します。

- 処理する必要のあるデータの量
- データ・ソースの処理速度
- フラグメントが戻すデータの量
- 通信の帯域幅
- 同じ照会結果を表す、使用可能なマテリアライズ照会表がフェデレーテッド・サーバーにあるかどうか

照会オプティマイザーは、照会フラグメントを処理するための代替アクセス・プランを生成します。フェデレーテッド・サーバーのローカルおよびリモート・データ・ソースで代替プランによって行われる作業量は一定ではありません。照会オプティマイザーはコスト・ベースであるため、リソースの消費コストを代替アクセス・プランに割り当てます。それから、照会オプティマイザーは最小のリソース消費コストで照会を処理するプランを選択します。

何らかのフラグメントをデータ・ソースで処理する場合、フェデレーテッド・データベースはそれらのフラグメントをデータ・ソースにサブミットします。データ・ソースがフラグメントを処理した後、結果が取り出されてフェデレーテッド・データベースに戻されます。フェデレーテッド・データベースが処理の一部を実行した場合、その処理結果とデータ・ソースから取り出した結果が組み合わされます。それから、フェデレーテッド・データベースはすべての結果をクライアントに戻します。

適合

データ・ソースではサポートされていない SQL を処理する IBM InfoSphere Federation Server の機能を、*適合* と呼びます。

データ・ソースが照会フラグメントを処理できない場合、またはデータ・ソースで処理するよりもフェデレーテッド・データベースで処理した方が速い場合、フェデレーテッド・サーバーは照会フラグメントをプッシュダウンしません。例えば、データ・ソースの SQL ダイアレクトが、GROUP BY 文節の中では CUBE グループ化をサポートしていないとします。この場合、CUBE グループ化を含み、かつそのデータ・ソース内の表を参照する照会は、フェデレーテッド・サーバーにサブミットされます。フェデレーテッド・データベースは CUBE グループ化をデータ・ソースにプッシュダウンせず、CUBE 自体を処理します。

フェデレーテッド・データベースは、データ・ソース側の機能不足を次の 2 つの方法で補います (適合させます)。

- 照会に記述されている DB2 関数と同等の、1 つまたは複数の操作を使用するようデータ・ソースに求めることができます。例えば、あるデータ・ソースがコタンジェント (COT(x)) 関数をサポートしておらず、タンジェント (TAN(x)) 関数を

サポートしているとします。このときフェデレーテッド・データベースは、コタンジェント (COT(x)) 関数と同等の計算 (1/TAN(x)) を実行するようデータ・ソースに求めることができます。
- データ集合をフェデレーテッド・サーバーに戻し、関数をローカルで実行することができます。

リレーショナル・データ・ソースの場合は、RDBMS の各タイプがそれぞれ国際 SQL 標準のサブセット 1 つをサポートします。また、この標準を超える SQL 構造をサポートする RDBMS のタイプもあります。 *SQL* ダイアレクトとは、1 つの RDBMS タイプがサポートする SQL の全体を指す用語です。ある SQL 構造が DB2 の SQL ダイアレクトにあり、リレーショナル・データ・ソース側のダイアレクトにはない場合、フェデレーテッド・サーバーはそのデータ・ソースに代わってこの構造をインプリメントすることができます。

フェデレーテッド・データベースは SQL ダイアレクト内の相違を補う (適合させる) ことができます。その一例は common-table-expression (共通表式) という文節で、この文節は DB2 SQL に組み込まれています。この文節に指定できるのは、全選択内のどの FROM 文節に指定しても結果セットを参照できる名前です。データ・ソースが使用する SQL ダイアレクトに共通表式が組み込まれていなくても、フェデレーテッド・サーバーはそのデータ・ソースの共通表式を処理します。

「適合」機能のおかげで、フェデレーテッド・データベースはデータ・ソースを照会する際に、 DB2 SQL ダイアレクトを完全にサポートすることができます。 SQL サポートが弱いまたは SQL サポートがないデータ・ソースに対してであっても、適合は適用されます。パススルー・セッションの場合を除き、DB2 SQL ダイアレクトは必ずフェデレーテッド・システムで使用するようにしてください。

パススルー・セッション

パススルー と呼ばれる特殊モードを使用すると、SQL ステートメントをデータ・ソースに直接サブミットすることができます。

この場合 SQL ステートメントは、データ・ソースで使用されている SQL ダイアレクトを使用してサブミットします。しかし、DB2 SQL/API ではできない操作を実行する場合には、パススルー・セッションを使用してください。例えば、プロシージャーの作成、索引の作成、またはデータ・ソースに固有のダイアレクトを使用した照会の実行には、パススルー・セッションを使用します。

現在、パススルーをサポートするデータ・ソースは、SQL を使用するパススルーをサポートします。将来的には、データ・ソースが SQL 以外のデータ・ソース言語を使用するパススルーをサポートすることもありえます。

同様に、パススルー・セッションを使用して、SQL がサポートしていないアクション (例えば、ある種の管理用タスク) を実行することもできます。ただし、パススルー・セッションを使用してすべての管理用タスクを実行することはできません。例えば、データ・ソース側で表を作成したりドロップすることはできますが、リモート・データベースを開始または停止することはできません。

パススルー・セッションでは、静的 SQL と動的 SQL の両方を使用することができます。

フェデレーテッド・サーバーには、パススルー・セッションを管理するための次の SQL ステートメントが提供されています。

SET PASSTHRU
パススルー・セッションを開く。別の SET PASSTHRU ステートメントを発行して新しいパススルー・セッションを開始すると、現行のパススルー・セッションは終了します。

SET PASSTHRU RESET
現行のパススルー・セッションを終了する。

GRANT (Server Privileges)
ユーザー、グループ、許可 ID のリスト、または PUBLIC に対して、特定のデータ・ソースにパススルー・セッションを開始する権限を付与する。

REVOKE (Server Privileges)
パススルー・セッションを開始する権限を取り消す。

サーバー定義およびサーバー・オプション

データ・ソース用のラッパーを作成した後、フェデレーテッド・インスタンスの所有者はデータ・ソースをフェデレーテッド・データベースに定義します。

インスタンス所有者は、データ・ソースを識別するための名前を指定し、またデータ・ソースに関するその他の情報も指定します。この情報には、次のものが含まれます。

- データ・ソースのタイプおよびバージョン
- データ・ソースのデータベース名 (RDBMS のみ)
- データ・ソースに固有のメタデータ

例えば、DB2 ファミリーのデータ・ソースは複数のデータベースを持つことができます。そのため、フェデレーテッド・サーバーがどのデータベースに接続できるかを定義に指定しておく必要があります。それとは対照的に、Oracle データ・ソースが持つデータベースは 1 つなので、フェデレーテッド・サーバーは名前を知らなくてもそのデータベースに接続することができます。そのため、Oracle データ・ソースのフェデレーテッド・サーバー定義にデータベース名は含まれていません。

インスタンス所有者がフェデレーテッド・サーバーに提供する、名前およびその他の情報をまとめてサーバー定義 と呼びます。データ・ソースはデータを求める要求に応答し、それ自体がサーバーとして機能します。

サーバー定義の作成および変更には、CREATE SERVER および ALTER SERVER ステートメントを使用します。

サーバー定義内の情報の一部は、サーバー・オプション として保管されます。サーバー定義を作成するにあたって、サーバーに関して指定可能なオプションを理解しておくことは大切です。

サーバー・オプションは、データ・ソースへの接続が次々に続く間は持続されるように設定するか、または 1 つの接続が継続している間のみ持続されるように設定することができます。

ユーザー・マッピング

ユーザー・マッピングは、フェデレーテッド・サーバー上の 許可 ID とリモート・データ・ソースに接続するために必要な情報との間の関連です。

ユーザー・マッピングを作成するには、以下の情報を CREATE USER MAPPING ステートメントで指定します。

- ローカル許可 ID
- サーバー定義で指定されたリモート・データ・ソース・サーバーのローカル名
- リモート ID およびパスワード

例えば、リモート・サーバー用にサーバー定義を作成して、リモート・サーバーのローカル名に 'argon' を指定したとします。 Mary にリモート・サーバーへのアクセスを付与するには、次のユーザー・マッピングを作成します。

```
CREATE USER MAPPING FOR Mary
SERVER argon
OPTIONS (REMOTE_AUTHID 'remote_ID', REMOTE_PASSWORD 'remote_pw')
```

Mary が SQL ステートメントを発行してリモート・サーバーに接続するとき、フェデレーテッド・サーバーは以下のステップを実行します。

1. Mary のユーザー・マッピングを検索します。
2. リモート・サーバーに関連付けられたリモート・パスワード 'remote_pw' を暗号解読します。
3. リモート・サーバーに接続するためのラッパーを呼び出します。
4. リモート ID 'remote_ID' および暗号解読したリモート・パスワードをラッパーに渡します。
5. Mary のためのリモート・サーバーへの接続を作成します。

デフォルトでは、フェデレーテッド・サーバーはユーザー・マッピングをグローバル・カタログ内の SYSCAT.USEROPTIONS ビューに保管して、リモート・パスワードを暗号化します。代替方法として、ファイルまたは LDAP サーバーなどの外部リポジトリーを使用して、ユーザー・マッピングを保管することもできます。フェデレーテッド・サーバーと外部リポジトリーとの間のインターフェースを提供するには、ユーザー・マッピング・プラグインを作成します。

ユーザー・マッピングをどのように保管する場合でも、それらに対するアクセスを注意深く制限してください。ユーザー・マッピングで暗号漏えいが発生した場合、リモート・データベース内のデータは、無許可の活動に対してぜい弱になることがあります。

InfoSphere Federation Server バージョン 9.7 フィックスパック 2 以降では、すべてのローカル・データベース・ユーザーが 1 つのリモート・ユーザー ID およびパスワードでデータ・ソースにアクセスできるようにする、パブリック・ユーザー・マッピングも作成できます。

ニックネームとデータ・ソース・オブジェクト

ニックネーム とは、アクセス先のデータ・ソース・オブジェクトを識別するために使用する ID です。ニックネームによって識別されるオブジェクトを、データ・ソース・オブジェクト といいます。

別名が代替名であるのとは異なり、ニックネームはデータ・ソース・オブジェクトの代替名ではありません。ニックネームは、フェデレーテッド・サーバーがオブジェクトを参照するために使用するポインターです。ニックネームは通常、CREATE NICKNAME ステートメントに、特定のニックネーム列オプションとニックネーム・オプションを指定して定義されます。

クライアント・アプリケーションまたはユーザーが分散要求をフェデレーテッド・サーバーにサブミットする場合、その要求でデータ・ソースを指定する必要はありません。その代わりに、要求はデータ・ソース・オブジェクトをそのオブジェクトのニックネームで参照します。ニックネームはデータ・ソースの特定のオブジェクトにマップされます。このようにマッピング (対応付け) されることにより、ニックネームをデータ・ソース名で修飾する必要がなくなります。クライアント・アプリケーションまたはユーザーは、データ・ソース・オブジェクトのロケーションを意識する必要がありません。

ここで、ニックネーム *DEPT* が、 *NFX1.PERSON* という Informix データベース表を表すように定義するとします。SELECT * FROM *DEPT* というステートメントをフェデレーテッド・サーバーから使用できます。しかし、フェデレーテッド・サーバーに *NFX1.PERSON* というローカル表がなければ、フェデレーテッド・サーバーから SELECT * FROM *NFX1.PERSON* というステートメントを使用することはできません (パススルー・セッションは除く)。

データ・ソース・オブジェクトにニックネームを作成すると、オブジェクトについてのメタデータがグローバル・カタログに追加されます。照会オプティマイザーは、このメタデータとラッパー内の情報を使用して、データ・ソース・オブジェクトへのアクセスを容易にします。例えば、索引を持つ表にニックネームを作成すると、グローバル・カタログにはその索引についての情報が入り、ラッパーには、DB2 のデータ・タイプとデータ・ソースのデータ・タイプとの間のマッピングが入ります。

ラベル・ベースのアクセス制御 (LBAC) を使用するオブジェクトのニックネームはキャッシュに入れられません。そのため、オブジェクトのデータの安全性は確保されます。例えば、Oracle (Net8) ラッパーを使用して、Oracle Label Security を使用する表に対してニックネームを作成する場合、その表の安全性は自動的に確認されます。その結果として生成されるニックネーム・データはキャッシュに入れることができません。したがって、そのデータに対してマテリアライズ照会表を作成することはできません。 LBAC を使用することにより、情報の表示が適切なセキュリティー特権を持つユーザーのみに制限されます。 LBAC がサポートされる前に作成されたニックネームについては、ALTER NICKNAME ステートメントを使用してキャッシングを使用不可にする必要があります。LBAC は、DRDA (DB2 for Linux, UNIX, and Windows バージョン 9.1 以降を使用するデータ・ソースに対応) および Net8 ラッパーの両方によってサポートされています。

有効なデータ・ソース・オブジェクト

アクセスするデータ・ソースのオブジェクトは、ニックネームで識別します。次の表は、フェデレーテッド・システム内のニックネームを作成できるオブジェクトのタイプを示しています。

表 2. 有効なデータ・ソース・オブジェクト

データ・ソース	有効なオブジェクト
BioRS	BioRS データ・バンク
DB2 Database for Linux, UNIX, and Windows	ニックネーム、マテリアライズ照会表、表、ビュー
DB2 for System i	表、ビュー、P/L (物理/論理) ファイルおよび表タイプ
DB2 for VM and VSE	表、ビュー
DB2 for z/OS	表、ビュー
Informix	表、ビュー、シノニム
JDBC	すべての表タイプ
Microsoft Excel	.xls ファイル (ワークブック内の最初のシートだけがアクセスされる)
Microsoft SQL Server	表、ビュー
ODBC	表、ビュー
Oracle	表、ビュー、シノニム
スクリプト	スクリプト
Sybase	表、ビュー
Teradata	表、ビュー
表構造ファイル	特定のフォーマットに適合するテキスト・ファイル
Web サービス	Web サービス記述言語ファイル内の操作
XML タグ・ファイル	XML 文書内の項目のセット

ニックネーム列オプション

グローバル・カタログには、ニックネームが付けられたオブジェクトに関する追加のメタデータ情報を入れることができます。このメタデータは、データ・ソース・オブジェクトの特定の列の値を記述したものです。このメタデータを、ニックネーム*列オプション* というパラメーターに割り当てます。

ニックネーム列オプションは、列内のデータを通常の列とは異なる方法で処理するようラッパーに指示します。 SQL コンパイラーと照会オプティマイザーは、メタデータを使用して、データにアクセスするためのよりよいプランを作成します。

ニックネーム列オプションは、ラッパーにその他の情報を提供するためにも使用されます。例えば XML データ・ソースの場合、ニックネーム列オプションは、ラッパーが XML 文書から列を解析するときに使用する XPath 式をラッパーに指示するために使用されます。

フェデレーションを使用すると、DB2 サーバーはニックネームが参照するデータ・ソース・オブジェクトを、あたかもローカル DB2 表であるかのように扱います。したがって、ニックネームを作成するどのデータ・ソース・オブジェクトに対しても、ニックネーム列オプションをセットすることができます。ニックネーム列オプションの中には特定のタイプのデータ・ソース用に作られたものもあり、それらは該当するデータ・ソースにのみ適用できます。

フェデレーテッド・データベースの照合シーケンスとは異なる照合シーケンスを持つデータ・ソースがあるとします。フェデレーテッド・サーバーは通常、文字データを含む列をデータ・ソース側でソートすることはありません。データはフェデレーテッド・データベースに戻され、ローカルにソートが行われます。しかしここで、列が文字データ・タイプ (CHAR または VARCHAR) であり、数字 ('0'、'1'、...、'9') だけが含まれているとします。これは、NUMERIC_STRING ニックネーム列オプションに 'Y' を指定することにより明示できます。そうすることにより、DB2 照会オプティマイザーは、オプションでデータ・ソース側でソートを実行できるようになります。ソートをリモート側で実行できれば、データをフェデレーテッド・サーバーに持ってきて、ソートをローカルで実行するというオーバーヘッドが避けられます。

ALTER NICKNAME ステートメントを使用することにより、リレーショナル・ニックネームにニックネーム列オプションを定義することもできます。非リレーショナル・ニックネームには、CREATE NICKNAME および ALTER NICKNAME ステートメントを使用してニックネーム列オプションを定義できます。

データ・タイプ・マッピング

フェデレーテッド・サーバーがデータ・ソースからデータを検索するには、データ・ソース側のデータ・タイプが、対応する DB2 のデータ・タイプに対応付けられて (マッピングされて) いなければなりません。

デフォルトのデータ・タイプ・マッピングの例として、以下のものがあります。

- Oracle タイプ FLOAT は DB2 タイプ DOUBLE にマップされます。
- Oracle タイプ DATE は DB2 タイプ TIMESTAMP にマップされます。
- DB2 for z/OS™ タイプ DATE は、DB2 タイプ DATE にマップされます。

ほとんどのデータ・ソースの場合、ラッパー内にデフォルトのタイプ・マッピングがあります。 DB2 データ・ソース用のデフォルトのタイプ・マッピングは、DRDA ラッパーにあります。 Informix 用のデフォルトのタイプ・マッピングは INFORMIX ラッパーにあります。その他のタイプ・マッピングについても同様です。

非リレーショナルのデータ・ソースの中には、CREATE NICKNAME ステートメントでデータ・タイプ情報を指定しなければならないものがあります。ニックネームの作成時に、データ・ソース・オブジェクトの列ごとに、対応する DB2 データ・タイプを指定する必要があります。それぞれの列は、データ・ソース・オブジェクト内の特定のフィールドまたは列にマップされている必要があります。

リレーショナル・データ・ソースの場合は、デフォルトのデータ・タイプ・マッピングをオーバーライドできます。例えば、Informix INTEGER データ・タイプは、

デフォルトでは DB2 INTEGER データ・タイプにマップします。デフォルト・マッピングをオーバーライドして、Informix の INTEGER データ・タイプを DB2 DECIMAL(10,0) データ・タイプにマップされるようにすることができます。

関数マッピング

フェデレーテッド・サーバーがデータ・ソース関数を認識するには、その関数が DB2 Database for Linux, UNIX and Windows にある対応する既存の関数にマップされている必要があります。

IBM InfoSphere Federation Server は、既存のデータ・ソース関数と DB2 側の対応する関数の間のデフォルトのマッピングを備えています。ほとんどのデータ・ソースの場合、ラッパー内にデフォルトの関数マッピングがあります。 DB2 for z/OS 関数へのデフォルトの関数マッピングは、 DRDA ラッパーにあります。 Sybase 関数へのデフォルトの関数マッピングは CTLIB ラッパーにあります。その他のタイプ・マッピングについても同様です。

リレーショナル・データ・ソースの場合、フェデレーテッド・サーバーが認識しないデータ・ソース関数を使用するには、関数マッピングを作成する必要があります。データ・ソース関数とフェデレーテッド・データベースにある対応する DB2 関数との間に、マッピングを作成します。関数マッピングは通常、新しい組み込み関数や新しいユーザー定義関数がデータ・ソース側で使用可能になったときに使用されます。関数マッピングは、DB2 側に対応する関数が存在しない場合にも使用されます。その場合には、関数テンプレートも作成しなければなりません。

索引仕様

データ・ソース表にニックネームを作成すると、データ・ソース表が持つすべての索引についての情報がグローバル・カタログに追加されます。照会オプティマイザーはこの情報を使用して、分散要求の処理を効率よく行います。データ・ソース索引についてのカタログ情報はメタデータの集まりであり、索引仕様と呼ばれます。

照会オプティマイザーはこの情報を使用して、分散要求の処理を効率よく行います。

フェデレーテッド・サーバーは、次のオブジェクトにニックネームが作成された場合、索引の仕様を作成しません。

- 索引を持たない表
- ビュー。通常、ビューにはリモート・カタログに保管される索引情報はありません。
- フェデレーテッド・サーバーが索引情報を入手できるリモート・カタログを持たない、データ・ソース・オブジェクト

ニックネームの作成時にあった索引に加えて、新しい索引が表に追加されたとします。索引情報はニックネーム作成時にグローバル・カタログに入るため、フェデレーテッド・サーバーは新しい索引については認識していません。同様に、ビューのニックネームを作成しても、フェデレーテッド・サーバーはそのビューの基になる表 (およびその索引) を認識してはいません。このような場合には、ユーザーが必要な索引情報をグローバル・カタログに入れることができます。索引を持たない表の

「索引仕様」を作成することができます。「索引仕様」は、データを速く見付けるには表内のどの列を検索すべきかを照会オプティマイザーに示します。

フェデレーテッド・ストアード・プロシージャー

フェデレーテッド・プロシージャー・アクセスを使用すると、フェデレーテッド・システムのユーザーはリモート・データ・ソースのリモート・ストアード・プロシージャーにアクセスできるようになります。

フェデレーテッド・ストアード・プロシージャーはデータ・ソースのストアード・プロシージャーにマップされるローカルのストアード・プロシージャーです。CREATE PROCEDURE (ソース派生) ステートメントを使用して、フェデレーテッド・ストアード・プロシージャーを登録します。

照合シーケンス

データベースでの文字データのソート順序は、データの構造や、データベースで定義されている照合シーケンスによって異なります。

データベース内のデータがすべて大文字で、数値や特殊文字が含まれないとします。データがデータ・ソースでソートされるか、フェデレーテッド・データベースでソートされるかに関係なく、データのソートは同じ出力になるはずです。各データベースで使用される照合シーケンスは、ソート結果に影響を与えません。データベース内のデータがすべて小文字か、すべて数字の場合も同様に、実際にソートが実行される場所に関係なく、データのソートは同じ結果を生成します。

データが次のいずれかの構造で構成されるとします。

- 文字と数字の組み合わせ
- 大文字と小文字の両方
- @、#、€ などの特殊文字。

フェデレーテッド・データベースが使用する照合シーケンスとデータ・ソースが使用する照合シーケンスが異なる場合、このデータをソートしたときの出力結果は異なる可能性があります。

一般に、照合シーケンスという言葉は、特定の文字を別の文字より高い位置にソートするか、低い位置にソートするか、あるいは同じ位置にソートするかを決定する文字データの定義済みの順序付けを指します。

照合シーケンスによるソート順序の決定方法

照合シーケンスは、コード化文字セットの文字のソート順序を決定します。

文字セットは、コンピューター・システムまたはプログラミング言語で使用される文字の集合です。コード化文字セットの文字はそれぞれ、0 から 255 の範囲内の異なる数値 (またはそれに相当する 16 進数) に割り当てられます。その数値はコード・ポイントと呼ばれ、セット内の文字への数値の割り当ては集合的に、コード・ページと呼ばれます。

文字への割り当てに加え、コード・ポイントはソート順序の文字の位置にマップすることができます。専門的に言うと、照合シーケンスは、文字セットのコード・ポ

イントの、セットの文字のソート順序位置への集合的なマッピングです。文字の位置は数値によって表され、この数値を文字の重みといいます。最も単純な照合シーケンス (ID シーケンスという) での重みは、コード・ポイントと等しくなります。

例: データベース ALPHA は、デフォルト照合シーケンスの EBCDIC コード・ページを使用します。データベース BETA は、デフォルト照合シーケンスの ASCII コード・ページを使用します。以下のように、これら 2 つのデータベースの文字ストリングのソート順序は異なります。

```
SELECT.....
  ORDER BY COL2

EBCDIC-Based Sort           ASCII-Based Sort

COL2                        COL2
----                        ----
V1G                         7AB
Y2W                         V1G
7AB                         Y2W
```

例: 同様に、データベースの文字比較もそのデータベースで定義される照合シーケンスによって異なります。データベース ALPHA は、デフォルト照合シーケンスの EBCDIC コード・ページを使用します。データベース BETA は、デフォルト照合シーケンスの ASCII コード・ページを使用します。以下のように、これら 2 つのデータベースでの文字比較によって生成される結果は異なります。

```
SELECT.....
  WHERE COL2 > 'TT3'

EBCDIC-Based Results     ASCII-Based Results

COL2                     COL2
----                     ----
TW4                      TW4
X82                      X82
39G
```

照会最適化のためのローカル照合シーケンスの設定

管理者は、データ・ソースの照合シーケンスに一致する特定の照合シーケンスを持つフェデレーテッド・データベースを作成することができます。

それから、各データ・ソース・サーバー定義ごとに COLLATING_SEQUENCE サーバー・オプションを「Y」に設定します。この設定は、フェデレーテッド・データベースに、フェデレーテッド・データベースとデータ・ソースの照合シーケンスが一致していることを知らせます。

フェデレーテッド・データベースの照合シーケンスは、CREATE DATABASE コマンドの一部として設定します。このコマンドを通して、次のいずれかのシーケンスを指定できます。

- 一致シーケンス
- システム・シーケンス (データベースをサポートするオペレーティング・システムが使用するシーケンス)
- カスタマイズ・シーケンス (DB2 データベース・システムが提供する事前定義のシーケンス、またはユーザー定義のシーケンス)

データ・ソースが DB2 for z/OS であるとします。 ORDER BY 文節で定義されるソートは、EBCDIC コード・ページに基づく照合シーケンスによってインプリメントされます。 ORDER BY 文節に従ってソートされた DB2 for z/OS データを取得するには、該当する EBCDIC コード・ページに基づく事前定義照合シーケンスを使用するようフェデレーテッド・データベースを構成します。

第 2 章 データ・ソース構成の変更

時間の経過とともに、フェデレーテッド・システムでのデータ・ソース構成を変更する必要が生じることがあります。例えば、データ・ソースが変更された場合、サーバー定義、ニックネーム、およびユーザー・マッピングの更新が必要になる場合があります。また、フェデレーテッド・システムからデータ・ソースへのアクセスを追加または除去することが必要になる場合もあります。

フェデレーテッド・オブジェクトの変更の例では、DB2 コマンド行から SQL ステートメントを発行することによりオブジェクトを変更する方法について説明します。Data Studio で SQL および XQuery エディターを使用して SQL スクリプトを発行することもできます。サポートされる DB2 データ・ソースおよび Oracle データ・ソースの場合、Data Studio の管理エクスプローラーを使用することができます。管理エクスプローラーには、フェデレーテッド・オブジェクトの変更タスクをガイドするダイアログ・ボックスとウィザードが用意されています。

ラッパーの変更

- **ラッパーの変更**

サーバー定義およびサーバー・オプションの変更

- **サーバー定義およびサーバー・オプションの変更**
- **サーバー定義でのサーバー・オプションの使用**

ユーザー・マッピングの変更

- **ユーザー・マッピングの変更**

ニックネームの変更

- **ニックネームの変更**

ラッパー、サーバー定義、ユーザー・マッピング、およびニックネームのドロップ

- **ラッパーのドロップ**
- **サーバー定義のドロップ**
- **ユーザー・マッピングのドロップ**
- **ニックネームのドロップ**

ラッパーの変更

ラッパーの構成が終わったなら、次に ALTER WRAPPER ステートメントを使用して、システム要件に基づいて構成の変更を行うことができます。

始める前に

ステートメントに関連付けられた許可 ID には、SYSADM または DBADM 権限が必要です。

このタスクについて

このタスクは、DB2 コマンド行からラッパーを変更する方法について説明します。ALTER WRAPPER ステートメントを使用して、 1 つ以上のラッパー・オプションの追加、設定、またはドロップを行えます。

制約事項:

- DB2_FENCED ラッパー・オプションはドロップできません。
- 任意の作業単位内に次のいずれかのステートメントが組み込まれている場合には、フェデレーテッド・サーバーはその作業単位内で ALTER WRAPPER ステートメントを処理することができません。
  - ラッパーに組み込まれているデータ・ソースにある表またはビューのニックネームを参照する SELECT ステートメント
  - ラッパーに組み込まれているデータ・ソースにある表またはビューのニックネーム上のオープン・カーソル
  - ラッパーに組み込まれているデータ・ソースにある表またはビューのニックネームに対して発行される挿入、削除、または更新

手順

DB2 コマンド行からラッパーを変更する場合は、ALTER WRAPPER ステートメントを発行します。

ラッパーの変更例

このトピックには、ALTER WRAPPER ステートメントを使用したラッパー・オプションの変更例が記載されています。

drda という名前のラッパーの DB2_FENCED オプションを 'Y' に変更するには、次のステートメントを発行します。

```
ALTER WRAPPER drda OPTIONS (SET DB2_FENCED 'Y');
```

odbc という名前のラッパーの MODULE オプションを '/opt/odbc/lib/libodbc.a (odbc.so)' に変更するには、次のステートメントを発行します。

```
ALTER WRAPPER odbc OPTIONS (SET MODULE '/opt/odbc/lib/libodbc.a(odbc.so)');
```

サーバー定義およびサーバー・オプションの変更

サーバー定義を変更するには、ALTER SERVER ステートメントを使用します。サーバー定義内の情報の一部は、サーバー・オプションとして保管されます。サーバー定義を変更するにあたって、サーバーに関して指定可能なオプションを理解しておくことは大切です。

サーバー定義は、フェデレーテッド・データベースに対するデータ・ソースを識別します。サーバー定義は、データ・ソース・サーバーのローカル名とその他の情報から構成されています。ニックネームを使用する SQL ステートメントが、フェデレーテッド・データベースにサブミットされるときに、サーバーの定義はラッパーによって使用されます。

リレーショナル・データ・ソースの場合は、サーバー・オプションは SET SERVER OPTION ステートメントを使用して、一時的に設定することもできます。このステートメントは、フェデレーテッド・データベースへの 1 回の接続の間、サーバー定義のサーバー・オプション値をオーバーライドします。静的 SQL の場合、SET SERVER OPTION ステートメントの使用はその静的 SQL ステートメントの実行にのみ影響します。SET SERVER OPTION ステートメントを使用しても、オプティマイザーが生成するプランには影響しません。

次の場合には、サーバー定義を変更します。

- 新しいバージョンのデータ・ソースにアップグレードする。
- 特定のデータ・ソース・タイプに関して、すべてのサーバー定義に変更を加える。
- 既存のサーバー定義でサーバー・オプションを追加または変更する。

サーバー定義の変更に関する制限

サーバー定義を変更する際には、複数の制約事項に注意する必要があります。

サーバー定義の変更には、以下の制約事項があります。

- フェデレーテッド・サーバーに登録されていないラッパーは、ALTER SERVER ステートメントに指定できません。
- 次のいずれかの条件のときに、フェデレーテッド・サーバーは指定の作業単位 (UOW) 内で ALTER SERVER ステートメントを処理することができません。
  - ALTER SERVER ステートメントが 1 つのデータ・ソースを参照しており、次のステートメントのいずれかがすでに UOW に組み込まれている。
    - データ・ソース内の表またはビューのニックネームを参照する SELECT ステートメント
    - データ・ソース内の表またはビューのニックネームのオープン・カーソル
    - データ・ソース内の表またはビューのニックネームに対して発行される挿入、削除、または更新
  - ALTER SERVER ステートメントがあるカテゴリーのデータ・ソース (例えば、特定のタイプやバージョンのすべてのデータ・ソース) を参照しており、次のステートメントのいずれかがすでに UOW に組み込まれている。
    - いずれかのデータ・ソース内の表またはビューのニックネームを参照する SELECT ステートメント
    - いずれかのデータ・ソース内の表またはビューのニックネームのオープン・カーソル
    - いずれかのデータ・ソース内の表またはビューのニックネームに対して発行される挿入、削除、または更新
- フェデレーテッド・サーバーは、指定されたサーバー・バージョンが、リモート・サーバー・バージョンと一致しているかどうか検証しません。指定されたサーバー・バージョンが誤っている場合は、DB2 サーバー定義に属するニックネームにアクセスする際に、SQL エラーになる可能性があります。この種の SQL エラーが生じる可能性が最も高くなるのは、リモート・サーバーの実際のバージョンより新しいサーバー・バージョンを指定する場合です。この場合、サーバー定義に属するニックネームにアクセスする際に、フェデレーテッド・サーバーがリ

モート・サーバーに SQL を送信しても、このリモート・サーバーは認識しません。ALTER SERVER ステートメントでは、この状態は、サーバー・バージョンを変更するステートメントの形式 (*server-name* VERSION server-version) のみに適用されます。

- サーバー・オプションのフルネームを指定する必要があります。例えば、省略形の DB2_TWO_PHASE は指定できません。代わりに、サーバー・オプションのフルネーム DB2_TWO_PHASE_COMMIT を指定する必要があります。

サーバー定義内のデータ・ソース・バージョンの変更

サーバー定義を変更して、リモート・サーバーが使用するデータ・ソースのバージョンを変更できます。

始める前に

ALTER SERVER ステートメントを発行する許可 ID は、フェデレーテッド・データベースに対する SYSADM または DBADM 権限を持っている必要があります。

このタスクについて

制約事項

サーバー定義の変更に関する制限を参照してください。

手順

DB2 コマンド行からサーバー定義を変更する場合は、ALTER SERVER ステートメントを発行します。
例: Microsoft SQL Server バージョン 6.5 のデータ・ソースのサーバー定義を処理しているとします。このサーバーに CREATE SERVER ステートメントで割り当てた名前は SQLSVR_ASIA です。 Microsoft SQL Server がバージョン 7.0 にアップグレードされた場合、次のステートメントはサーバー定義を変更します。

```
ALTER SERVER SQLSVR_ASIA VERSION 7
```

特定のデータ・ソース・タイプのすべてのサーバー定義の変更

単一の ALTER SERVER ステートメントで、特定のデータ・ソースのタイプについての既存のサーバー定義をすべて変更できます。同一タイプのすべてのサーバー定義に同じ変更を適用するには、単一のステートメントを使用します。

始める前に

ALTER SERVER ステートメントを発行する許可 ID は、フェデレーテッド・データベースに対する SYSADM または DBADM 権限を持っている必要があります。

このタスクについて

制約事項

以前に行った ALTER SERVER ステートメント操作でサーバー・オプションが追加されている場合には、特定の種類のデータ・ソース全体に対して ALTER SERVER

ステートメントを使用して、サーバー・オプションを設定またはドロップすることができます。

手順

特定のデータ・ソースのタイプについての既存のサーバー定義をすべて変更するには、単一の ALTER SERVER ステートメントを発行します。
例: Sybase データ・ソース用のグローバル・カタログに Sybase サーバーが 5 つ登録されているとします。ユーザー ID が認証のためにこれらの Sybase サーバーに送信されるたびに、フェデレーテッド・サーバーがそのユーザー ID を大文字に変換するようにします。さらに、フェデレーテッド・サーバーが Sybase サーバーからの SQL ステートメントに対する応答を待機する時間も設定します。時間は秒単位で指定します。次の ALTER SERVER ステートメントで 5 つのサーバー定義をすべて同時に変更できます。

```
ALTER SERVER TYPE sybase
   OPTIONS (ADD FOLD_ID 'U', ADD TIMEOUT '600')
```

サーバー定義でのサーバー・オプションの使用

サーバー・オプションには、汎用のサーバー・オプションと特定のデータ・ソース・タイプのサーバー・オプションがあります。

始める前に

ALTER SERVER ステートメントを発行する許可 ID は、フェデレーテッド・データベースに対する SYSADM または DBADM 権限を持っている必要があります。

このタスクについて

制約事項

23 ページの『サーバー定義の変更に関する制限』を参照してください。

サーバー・オプションにセットされた値は、データ・ソースに対して行われる複数回の接続にまたがって持続します。これらの値は、フェデレーテッド・システム・カタログに保管されます。

手順

コマンド行プロンプトからこのタスクを実行するには、ALTER SERVER ステートメントを発行します。
例:

- INFMX01 というサーバー名を使用して、Informix サーバーのサーバー定義を作成したとします。ここで、DB2_MAXIMAL_PUSHDOWN オプションを Y に変更します。サーバー定義を変更するステートメントは、次のようになります。

  ```
  ALTER SERVER INFMX01 OPTIONS (SET DB2_MAXIMAL_PUSHDOWN 'Y')
  ```

- ORCL99 というサーバー名を使用して、Oracle サーバーのサーバー定義を作成したとします。ここで、その定義に FOLD_ID オプションおよび FOLD_PW オプションを追加することにします。その場合、サーバー定義を変更するステートメントは次のようになります。

```
ALTER SERVER ORCL99 OPTIONS (ADD FOLD_ID 'U', FOLD_PW 'U')
```

- CTLIB ラッパーが Sybase サーバーからの応答を待機する秒数をタイムアウト値に設定します。この値を設定するには、TIMEOUT サーバー・オプションを使用します。その場合、サーバー定義を変更するステートメントは次のようになります。

```
ALTER SERVER SYBSERVER OPTIONS (ADD TIMEOUT '60')
```

リレーショナル・データ・ソース用のサーバー・オプションの一時的な変更

SET SERVER OPTION ステートメントは、フェデレーテッド・データベースへの 1 回の接続の間、サーバー定義のサーバー・オプション値をオーバーライドします。オーバーライドする値は、グローバル・カタログには保管されません。

このタスクについて

静的 SQL の場合、SET SERVER OPTION ステートメントの使用はその静的 SQL ステートメントの実行にのみ影響します。SET SERVER OPTION ステートメントを使用しても、オプティマイザーが生成するプランには影響しません。

手順

リレーショナル・データ・ソース用にサーバー・オプション値を一時的に指定する場合は、 SET SERVER OPTION ステートメントを使用します。
例:

```
SET SERVER OPTION PLAN_HINTS TO 'Y' FOR SERVER SYB_SERVER
```

サーバー・オプション設定値の階層

同一のサーバー・オプションの設定値に対して、データ・ソース・タイプの値と特定のデータ・ソース・サーバーの値が異なる場合には、それらの設定値間には階層が存在します。

例えば、データ・ソース・タイプ SYBASE について、PLAN_HINTS サーバー・オプションが 'Y' に設定されています。しかし、特定の Sybase データ・ソース・サーバー PURNELL については、PLAN_HINTS サーバー・オプションはサーバー定義で 'N' に設定されています。このとき、特定のデータ・ソース・サーバーの設定値は、データ・ソース・タイプの設定値をオーバーライドします。この構成により、PLAN_HINTS は PURNELL 以外のすべての Sybase データ・ソース・サーバーで使用可能になります。

ユーザー・マッピングの変更

ユーザー・マッピングは、フェデレーテッド・サーバーでの許可 ID とデータ・ソースでの許可 ID との間の関連付けです。ユーザー・マッピングは、分散要求をデータ・ソースに送信できるようにするために必要です。

始める前に

このステートメントを発行する許可 ID がデータ・ソースにマップされる許可 ID と異なる場合には、ステートメントを発行する許可 ID がフェデレーテッド・デー

タベースに対する SYSADM または DBADM 権限を持っている必要があります。

このタスクについて

ALTER USER MAPPING ステートメントは、特定のフェデレーテッド・サーバーの許可 ID 用のデータ・ソースで使用される許可 ID またはパスワードを変更するために用いられます。

制約事項

任意の作業単位 (UOW) 内に次のいずれかのステートメントが組み込まれている場合には、フェデレーテッド・サーバーはその UOW 内で ALTER USER MAPPING ステートメントを処理することができません。

- マッピングに組み込まれているデータ・ソースにある表またはビューのニックネームを参照する SELECT ステートメント
- マッピングに組み込まれているデータ・ソースにある表またはビューのニックネーム上のオープン・カーソル
- マッピングに組み込まれているデータ・ソースにある表またはビューのニックネームに対して発行された挿入、削除、または更新

手順

DB2 コマンド行からユーザー・マッピングを変更するには、ALTER USER MAPPING ステートメントを発行します。
例: Jenny はフェデレーテッド・サーバーを使用して、SYBSERVER と呼ばれる Sybase サーバーに接続します。 Jenny は *jennifer* という許可 ID でフェデレーテッド・サーバーにアクセスします。許可 ID *jennifer* は、Sybase サーバーで許可 ID *jenn* にマップされます。 Sybase サーバーの Jenny の許可 ID は *jen123* に変更されます。このように *jennifer* を *jen123* にマップする ALTER USER MAPPING ステートメントは、次のようになります。

```
ALTER USER MAPPING FOR jennifer SERVER SYBSERVER
   OPTIONS (SET REMOTE_AUTHID 'jen123')
```

Tomas は、フェデレーテッド・サーバーを使用して ORASERVER と呼ばれる Oracle サーバーに接続します。 Tomas は *tomas* という許可 ID でフェデレーテッド・サーバーにアクセスします。許可 ID *tomas* は、Oracle サーバーで許可 ID *tom* にマップされます。 Oracle サーバーでの Tomas のパスワードは変更されます。新しいパスワードは day2night です。このように *tomas* を新しいパスワードにマップする ALTER USER MAPPING ステートメントは、次のようになります。

```
ALTER USER MAPPING FOR tomas SERVER ORASERVER
   OPTIONS (SET REMOTE_PASSWORD 'day2night')
```

上記の REMOTE_AUTHID および REMOTE_PASSWORD ユーザー・オプションには、 CREATE SERVER ステートメントで FOLD_ID および FOLD_PW サーバー・オプションを 'U' または 'L' に設定している場合を除き、大文字小文字の区別があります。

ニックネームの変更

ニックネームとは、データ・ソースにあるアクセス対象のオブジェクトを参照するために使用する ID です。ニックネームを変更すると、グローバル・カタログに保管されているデータ・ソース列名を変更し、列オプションを設定することができます。

始める前に

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- SYSADM または DBADM 権限
- ステートメントで指定されたニックネームに対する ALTER 特権
- ステートメントで指定されたニックネームに対する CONTROL 特権
- ニックネームにスキーマ名がある場合は、スキーマに対する ALTERIN 特権
- ニックネームのカタログ・ビューの DEFINER 列に記録された、ニックネームの定義者

このタスクについて

以下の作業を行うときに、ニックネームを変更できます。

- データ・ソース・オブジェクトの列のローカル列名を変更する
- データ・ソース・オブジェクトの列のローカル・データ・タイプを変更する
- ニックネームと列のオプションを追加、設定、またはドロップする
- 主キーを追加またはドロップする
- 1 つ以上のユニーク制約、参照制約、またはチェック制約を追加またはドロップする
- 1 つ以上の参照制約、チェック制約、または関数依存関係制約を変更する

制約事項

『ニックネームの変更に関する制限』を参照してください。

手順

ニックネームを DB2 コマンド行から変更するには、 ALTER NICKNAME ステートメントに適切なパラメーター・セットを指定して発行します。
データ・ソースのオブジェクト構造や内容が大きく変更された場合は、ニックネーム統計情報を更新してください。多数の行が追加または削除されることも、大きな変更とみなします。

ニックネームの変更に関する制限

ニックネームを変更する際には、複数の制約事項に注意する必要があります。

列オプション

以下のオプションのいずれかが列に設定されている場合、その列に他のオプションを追加することはできません。

- SOAPACTIONCOLUMN

- URLCOLUMN
- PRIMARY_KEY
- FOREIGN_KEY

BioRS の場合

- ELEMENT_NAME オプションを使用して、列のエレメント名を変更する場合、新しい名前が正しいかどうかを確認するための検査は行われません。オプションが誤っていると、照会で列が参照された時にエラーになる可能性があります。
- IS_INDEXED 列オプションへの変更を行う場合、BioRS サーバーで変更が検査されることはありません。オプションが誤っていると、照会で列が参照された時にエラーになる可能性があります。

データ・タイプ

- 列のデータ・タイプを変更する場合、新しいデータ・タイプは、対応するデータ・ソース列またはエレメントのデータ・タイプと互換性がなければなりません。ローカル・データ・タイプを、リモート・データ・タイプと非互換のデータ・タイプに変更すると、予測不能なエラーが生じる場合があります。
- *local_data_type* を LONG VARCHAR、LONG VARGRAPHIC、 XML、またはユーザー定義タイプにすることはできません。
- *data_source_data_type* は、ユーザー定義のタイプにすることはできません。
- 非リレーショナル・データ・ソースの中には、既存のローカル・タイプのオーバーライドや新しいローカル・タイプの作成を行えないものがあります。この制限については、特定のデータ・ソース・ラッパーの資料を調べてください。
- 列のデータ・タイプのローカル指定が変更された場合、フェデレーテッド・データベース・マネージャーは、その列について収集された統計情報 (HIGH2KEY や LOW2KEY など) をすべて無効にします。
- 特定のデータ・ソース・オブジェクトのニックネームを使用してアクセスされた場合、ローカル・タイプがそのオブジェクトに設定されます。同じデータ・ソース・オブジェクトに、デフォルトのデータ・タイプ・マッピングを使用する別のニックネームを付けることができます。

索引 ALTER NICKNAME ステートメントは、フェデレーテッド・データベース内に新しいデータ・ソース索引を登録するためには使用できません。「索引の指定」を作成するには、CREATE INDEX ステートメントに SPECIFICATION ONLY 文節を指定します。

LOCAL NAME および LOCAL TYPE パラメーター

- 以下の場合、ALTER NICKNAME ステートメントを使用して、ニックネーム内の列のローカル名またはデータ・タイプを変更することはできません。
  - ニックネームがビュー、SQL メソッド、または SQL 関数で使用されている。
  - ニックネームに対してインフォメーショナル制約を定義している。

- ALTER NICKNAME ステートメントで、 LOCAL NAME パラメーター、LOCAL TYPE パラメーター、またはこの両方を指定する必要もある場合は、最後に federated_column_options 文節を指定する必要があります。

ニックネーム

ALTER NICKNAME ステートメントを使用して、BioRS ニックネームによって参照される、または BioRS ニックネームで使用される BioRS データ・バンクの名前を変更することはできません。 BioRS データ・バンクの名前が変更される場合、ニックネームをドロップして、再びそのニックネームを作成しなければなりません。

ALTER NICKNAME ステートメントを使用して、キャッシュ表またはマテリアライズ照会表のあるニックネームのキャッシングを使用不可にすることはできません。ニックネームのキャッシングを使用不可にする場合は、その前にキャッシュ表およびマテリアライズ照会表をドロップしなければなりません。

作業単位

フェデレーテッド・サーバーは、以下の条件のいずれかでは、指定の作業単位内で ALTER NICKNAME ステートメントを処理できません。

- ALTER NICKNAME ステートメントで参照されているニックネームが、同じ作業単位内にオープン・カーソルを持っている場合
- ALTER NICKNAME ステートメントで参照されているニックネームに関して、同じ作業単位で挿入、削除、または更新が発行される場合

ニックネームの列名の変更

ニックネームを変更すると、列名を変更できます。

始める前に

ステートメントを発行する許可 ID には、以下の特権が少なくとも 1 つ含まれていなければなりません。

- SYSADM または DBADM 権限
- ステートメントで指定されたニックネームに対する ALTER 特権
- ステートメントで指定されたニックネームに対する CONTROL 特権
- ニックネームにスキーマ名がある場合は、スキーマに対する ALTERIN 特権
- ニックネームのカタログ・ビューの DEFINER 列に記録された、ニックネームの定義者

このタスクについて

ニックネームを作成すると、データ・ソース・オブジェクトに関連付けられている列名は、フェデレーテッド・データベースに保管されます。データ・ソースには、ラッパーが列名を指定するものと、ニックネーム作成時にユーザーが列名を指定しなければならないものがあります。

制約事項

28 ページの『ニックネームの変更に関する制限』を参照してください。

手順

DB2 コマンド行からニックネームの列名を変更する場合は、ALTER NICKNAME ステートメントを発行します。

```
ALTER NICKNAME nickname
   ALTER COLUMN current_name
   LOCAL NAME new_name
```

ニックネーム・オプションの変更

ニックネーム・オプションとは、 CREATE NICKNAME および ALTER NICKNAME ステートメントを発行する時にニックネームに指定する、パラメーターのことです。 ALTER NICKNAME ステートメントを使用することにより、ニックネーム・オプションを追加、設定、またはドロップできます。

始める前に

ステートメントを発行する許可 ID には、以下の特権が少なくとも 1 つ含まれていなければなりません。

- SYSADM または DBADM 権限
- ステートメントで指定されたニックネームに対する ALTER 特権
- ステートメントで指定されたニックネームに対する CONTROL 特権
- ニックネームにスキーマ名がある場合は、スキーマに対する ALTERIN 特権
- ニックネームのカタログ・ビューの DEFINER 列に記録された、ニックネームの定義者

このタスクについて

制約事項

28 ページの『ニックネームの変更に関する制限』を参照してください。

手順

コマンド行プロンプトからニックネーム・オプションを変更するには、ALTER NICKNAME ステートメントを発行します。

```
ALTER NICKNAME nickname
    OPTIONS (SET option_name 'option_string_value')
```

例: 表構造ファイル drugdata1.txt にニックネーム DRUGDATA1 が作成されるとします。CREATE NICKNAME ステートメントで元々定義されていた完全修飾パスは /user/pat/drugdata1.txt でした。 FILE_PATH ニックネーム・オプションを変更するには、以下のステートメントを発行します。

```
ALTER NICKNAME DRUGDATA1 OPTIONS (SET FILE_PATH '/usr/kelly/data/drugdata1.txt')
```

ニックネーム列オプションの変更

ニックネーム列オプションは、ALTER NICKNAME ステートメントを使用して、追加、設定、またはドロップすることができます。

始める前に

ステートメントを発行する許可 ID には、以下の特権が少なくとも 1 つ含まれていなければなりません。

- SYSADM または DBADM 権限
- ステートメントで指定されたニックネームに対する ALTER 特権
- ステートメントで指定されたニックネームに対する CONTROL 特権
- ニックネームにスキーマ名がある場合は、スキーマに対する ALTERIN 特権
- ニックネームのカタログ・ビューの DEFINER 列に記録された、ニックネームの定義者

このタスクについて

列情報は、CREATE NICKNAME および ALTER NICKNAME ステートメントにニックネーム列オプション と呼ばれるパラメーターを使用して指定します。その値は大文字でも小文字でも指定できます。

制約事項

28 ページの『ニックネームの変更に関する制限』を参照してください。

手順

コマンド行プロンプトからニックネーム列オプションを変更するには、ALTER NICKNAME ステートメントを使用します。
初めてオプションを追加する場合は、ADD キーワードを使用してオプションを追加します。このオプションが以前に追加された場合は、SET キーワードを使用してそのオプションを変更します。
例 1: リレーショナル・データ・ソースでの NUMERIC_STRING 列オプションの指定
NUMERIC_STRING 列オプションは、文字タイプの列 (CHAR および VARCHAR) に適用されます。フェデレーテッド・データベースの照合シーケンスとは異なる照合シーケンスを持つデータ・ソースがあるとします。フェデレーテッド・サーバーは通常、文字データを含む列をデータ・ソース側でソートすることはありません。データはフェデレーテッド・データベースに戻され、ローカルにソートが行われます。しかしここで、列が文字データ・タイプであり、数字 ('0'、'1'、...、'9') だけが入っているとします。これは、NUMERIC_STRING 列オプションを 'Y' にすれば示すことができます。こうすることで、DB2 UDB 照会オプティマイザーには、データ・ソース側でソートを実行するオプションが追加されます。ソートをリモート側で実行できれば、データをフェデレーテッド・サーバーでソートするというオーバーヘッドが避けられます。
INDONESIA_SALES という名前の Oracle 表のニックネームが ORA_INDSALES であるとします。表にはデータ・タイプが VARCHAR の POSTAL_CODE という列があります。当初、その列には数字だけが含まれ、NUMERIC_STRING 列オプションは 'Y' に設定されていました。しかし今は、数字と数字以外の文字も入っています。このとき、NUMERIC_STRING 列オプションを 'N' に変更するには、次のステートメントを使用します。

```
ALTER NICKNAME ORA_INDSALES ALTER COLUMN POSTAL_CODE
   OPTIONS (SET NUMERIC_STRING 'N')
```

例 2: リレーショナル・データ・ソースでの VARCHAR_NO_TRAILING_BLANKS 列オプションの指定
VARCHAR_NO_TRAILING_BLANKS 列オプションは、末尾ブランクを含まない特定の列を識別するために使用できます。列に対して実行されるすべての操作 (比較演算など) を調べる際に、 SQL コンパイラーがこの設定を解析します。
INDONESIA_SALES という名前の Oracle 表のニックネームが ORA_INDSALES であるとします。表にはデータ・タイプが VARCHAR の NAME という列があります。 NAME 列には末尾ブランクは含まれていません。このニックネームに VARCHAR_NO_TRAILING_BLANKS オプションを追加するには、次のステートメントを使用します。

```
ALTER NICKNAME ORA_INDSALES ALTER COLUMN NAME
   OPTIONS (ADD VARCHAR_NO_TRAILING_BLANKS 'Y')
```

例 3: 非リレーショナル・データ・ソースでの XPATH 列オプションの指定
EMPLOYEE は XML データ・ソースのニックネームです。 *fnmae* 列には XPATH が指定されています。この XPATH 列オプションを別のパスに設定するには、次のステートメントを使用します。

```
ALTER NICKNAME EMPLOYEE ALTER COLUMN fname
   OPTIONS (SET XPATH './@first')
```

ラッパーのドロップ

ラッパーをドロップする場合、それにはいくつかの理由があります。

始める前に

DROP WRAPPER ステートメントを発行するには、 SYSADM または DBADM 権限を持っている必要があります。

このタスクについて

1 つのデータ・ソースにアクセスするときに使用できるラッパーが複数ある場合があります。どのラッパーを選択するかは、使用しているデータ・ソース・クライアント・ソフトウェアのバージョンによって異なります。または、フェデレーテッド・サーバーで使用しているオペレーティング・システムによって決まる場合もあります。例えば、2 つの Oracle 表と 1 つの Oracle ビューにアクセスするとします。 Oracle バージョン 10 を使用しており、フェデレーテッド・サーバーのオペレーティング・システムは Windows です。当初は、SQLNET ラッパーを作成して使用していました。しかし、IBM InfoSphere Federation Server は SQLNET ラッパーをサポートしないため、SQLNET ラッパーをドロップして NET8 ラッパーを作成することにします。

ラッパーをドロップする別の理由としては、そのラッパーが関連付けられているデータ・ソースにアクセスする必要がなくなったという場合があります。例えば、ある企業で Informix データベースと Microsoft SQL server データベースの両方のクライアント情報にアクセスする必要があるとします。そこで、Informix データ・ソース用と Microsoft SQL Server データ・ソース用にそれぞれ 1 つずつラッパーを作成しました。ところがその後、すべての情報を Microsoft SQL Server から Informix に移行することが決まりました。こうして、Microsoft SQL Server ラッパーはもう必要ないので、ドロップできます。

重要: ラッパーをドロップすると、重大な影響が生じます。フェデレーテッド・サーバーに登録した他のオブジェクトは次のような影響を受けます。

- ドロップされたラッパーに依存するサーバー定義もすべてドロップされます。
- ドロップされたサーバー定義に依存するオブジェクトもすべてドロップされます。
- ドロップされたサーバー定義に依存するニックネームもすべてドロップされます。サーバー定義に依存するニックネームをドロップすると、それらのニックネームに依存するオブジェクトが影響を受けます。
  - ドロップされたニックネームに従属する索引の指定はすべてドロップされます。
  - ドロップされたニックネームに従属するビューはすべて「作動不能」とマークされます。
  - ドロップされたニックネームに依存するマテリアライズ照会表はすべてドロップされます。
- ドロップされたニックネームに依存するすべてのパッケージおよびキャッシュ付き動的 SQL ステートメントは無効とマークされ、従属オブジェクトが再作成されるまで無効のままになります。

手順

ラッパーをドロップするには、DROP ステートメントを使用します。
例: Microsoft SQL Server *MSSQLODBC3* ラッパーのドロップ:

```
DROP WRAPPER MSSQLODBC3
```

サーバー定義のドロップ

サーバー定義をドロップすると、グローバル・カタログから定義が削除されます。サーバー定義が参照するデータ・ソース・オブジェクトは影響を受けません。サーバー定義は、DB2 コントロール・センターを使用して、または DB2 コマンド行プロセッサーから DROP ステートメントを使用して、ドロップすることができます。

始める前に

サーバー定義をドロップするには、SYSADM または DBADM 権限を持っている必要があります。

このタスクについて

あるデータ・ソース・サーバーにアクセスする必要がなくなった場合、フェデレーテッド・データベースからサーバー定義をドロップします。サーバー定義をドロップすると、フェデレーテッド・サーバーに登録した他のオブジェクトに次のような影響が及びます。

- ドロップされたサーバー定義に依存する、すべてのユーザー定義関数マッピング、ユーザー定義データ・タイプ・マッピング、およびユーザー・マッピングもドロップされます。
- ドロップされたサーバー定義に依存するニックネームもすべてドロップされます。サーバー定義に依存するニックネームをドロップすると、それらのニックネームに依存するオブジェクトが影響を受けます。

- ドロップされたニックネームに従属する索引の指定はすべてドロップされます。
- ドロップされたニックネームに従属するビューはすべて「作動不能」とマークされます。
- ドロップされたニックネームに依存するマテリアライズ照会表はすべてドロップされます。

• ドロップされたニックネームに依存するすべてのパッケージおよびキャッシュ付き動的 SQL ステートメントは無効とマークされ、従属オブジェクトが再作成されるまで無効のままになります。

制約事項

次のいずれかの条件のときに、フェデレーテッド・サーバーは指定の作業単位 (UOW) 内で DROP SERVER ステートメントを処理することができません。

• ALTER SERVER ステートメントが 1 つのデータ・ソースを参照しており、次のステートメントのいずれかがすでに UOW に組み込まれている。
  - データ・ソース内の表またはビューのニックネームを参照する SELECT ステートメント
  - データ・ソース内の表またはビューのニックネームのオープン・カーソル
  - データ・ソース内の表またはビューのニックネームに対して発行される挿入、削除、または更新
• ALTER SERVER ステートメントがあるカテゴリーのデータ・ソース (例えば、特定のタイプやバージョンのすべてのデータ・ソース) を参照しており、次のステートメントのいずれかが UOW に組み込まれている。
  - いずれかのデータ・ソース内の表またはビューのニックネームを参照する SELECT ステートメント
  - いずれかのデータ・ソース内の表またはビューのニックネームのオープン・カーソル
  - いずれかのデータ・ソース内の表またはビューのニックネームに対して発行される挿入、削除、または更新

手順

サーバー定義を削除するには、DROP ステートメントを発行します。
サーバー定義を削除するには、DROP ステートメントを発行します。

```
DROP SERVER server_name
```

ここで *server_name* はドロップするサーバー定義を識別します。
例: Informix サーバーがサーバー名 INFMX01 を使用します。以下の DROP ステートメントはサーバー定義をドロップします。

```
DROP SERVER INFMX01
```

ユーザー・マッピングのドロップ

リモート・データ・ソースへのアクセスがユーザーには必要でなくなったなら、フェデレーテッド・サーバーとリモート・データ・ソース・サーバーの間のユーザー・マッピングをドロップします。ユーザーが複数のデータ・ソース・サーバーにマップしている場合には、それぞれのマッピングを別々にドロップする必要があります。

始める前に

DROP USER MAPPING ステートメントを発行するには、 DROP ステートメントの許可 ID がユーザー・マッピングで指定されたフェデレーテッド・データベースのユーザー ID と異なる場合に、この許可 ID が SYSADM または DBADM 権限を持っている必要があります。許可 ID とユーザー・マッピング内のユーザー ID が一致する場合には、特権または権限は必要ありません。

手順

ユーザー・マッピングをドロップするには、DROP ステートメントを発行します。

```
DROP USER MAPPING FOR user_ID SERVER local_server_name
```

ここで、

- *user_ID* は、フェデレーテッド・サーバー上でのユーザーの許可 ID です。
- *local_server_name* は、サーバー定義でリモート・データ・ソース・サーバーを識別するために使用されるローカル名です。

ニックネームのドロップ

ニックネームをドロップすると、フェデレーテッド・サーバーのグローバル・カタログからニックネームが削除されます。ニックネームが参照するデータ・ソース・オブジェクトは影響を受けません。

始める前に

ニックネームはカタログにリストされていなければなりません。

ニックネームをドロップする場合、DROP ステートメントの許可 ID が持つ特権は、次のいずれかでなければなりません。

- SYSADM または DBADM 権限
- ニックネームのスキーマに対する DROPIN 特権
- ニックネームのカタログ・ビューの DEFINER 列に記録された、ニックネームの定義者
- ニックネームに対する CONTROL 特権

このタスクについて

ニックネームをドロップすると、フェデレーテッド・サーバーに登録した他のオブジェクトに次のような影響が及びます。

- ニックネームをドロップすると、そのニックネームに従属するオブジェクトが影響を受けます。

- ドロップされたニックネームに従属する索引の指定はすべてドロップされます。
- ドロップされたニックネームに従属するビューはすべて「作動不能」とマークされます。
- ドロップされたニックネームに依存するマテリアライズ照会表はすべてドロップされます。

- ドロップされたニックネームに依存するすべてのパッケージおよびキャッシュ付き動的 SQL ステートメントは無効とマークされ、従属オブジェクトが再作成されるまで無効のままになります。

制約事項

リレーショナル・データ・ソースを参照するニックネームの場合は、次のいずれかの条件のときに、フェデレーテッド・サーバーは任意の作業単位 (UOW) 内で DROP NICKNAME ステートメントを処理することができません。

- 同じ UOW 内にステートメントで参照されているニックネームのオープン・カーソルがある。
- このステートメントで参照されているニックネームのカーソルが、同じ UOW 内の SELECT ステートメントによって参照されている。
- ステートメントで参照されているニックネームに対して、同じ UOW 内で挿入、削除、または更新が発行されている。

非リレーショナル・データ・ソースを参照するニックネームの場合は、次のいずれかの条件のときに、フェデレーテッド・サーバーは任意の作業単位 (UOW) 内で DROP NICKNAME ステートメントを処理することができません。

- このステートメントで参照されているニックネームのカーソルが同じ UOW 内でオープンしている。
- このステートメントで参照されているニックネームのカーソルが、同じ UOW 内の SELECT ステートメントによって参照されている。

手順

ニックネームを削除するには、DROP ステートメントを発行します。

```
DROP NICKNAME nickname
```

ここで *nickname* はドロップするニックネームを示します。

第 3 章 フェデレーテッド・システムでのデータ・タイプ・マッピング

データ・ソース側のデータ・タイプは、対応する DB2 データ・タイプにマップされなければなりません。このようにマッピングされることによって、フェデレーテッド・サーバーはデータ・ソースからデータを検索できるようになります。

フェデレーテッド・データベースによってデフォルトのデータ・タイプ・マッピングが提供されているデータ・ソースもありますが、使用するデータ・タイプ・マッピングをユーザーが提供しなければならないデータ・ソースもあります。非リレーショナル・データ・ソースの場合、既存のデータ・タイプ・マッピングをオーバーライドしたり、新しいマッピングを作成したりすることはできません。

デフォルトのデータ・タイプ・マッピングの例として、以下のものがあります。

- Oracle タイプの FLOAT は、デフォルトで DB2 タイプの DOUBLE にマップされます。
- Oracle タイプの DATE は、デフォルトで DB2 タイプの TIMESTAMP にマップされます。
- DB2 for z/OS タイプ DATE は、デフォルトで DB2 タイプ DATE にマップされます。

マッピング変更後に作成されたニックネームは、新しいタイプ・マッピングを使用します。マッピング変更前に作成されたニックネームは、デフォルトのデータ・タイプ・マッピングを使用します。

既に作成したニックネームがある場合には、次の 2 通りの方法で既存のニックネームを更新できます。

- 各ニックネームを変更する。
- 各ニックネームをドロップしてから再作成する。

DB2 フェデレーテッド・サーバーは、以下のデータ・タイプのマッピングをサポートしていません。

- ローカル・データ・タイプを LONG VARCHAR、LONG VARGRAPHIC、またはユーザー定義タイプにすることはできません。
- リモート・データ・タイプをユーザー定義タイプにすることはできません。

  ただし、リモート・データ・ソース側のビューで cast 関数を使用して、ユーザー定義データ・タイプを組み込みデータ・タイプまたはシステム・データ・タイプに変換することは可能です。そのようにしてから、ビューのニックネームを作成できます。ほとんどのデータ・ソースの場合、このようにして作成されたビューは統計情報や索引情報を持ちません。また、これらのビューを更新することはできません。

データ・タイプ・マッピングとフェデレーテッド・データベース・グローバル・カタログ

ローカル・データ・タイプ定義は、フェデレーテッド・データベース・グローバル・カタログの SYSCAT.COLUMNS カタログ・ビューに保管されます。

CREATE NICKNAME ステートメントを書くときには、ニックネームが表すデータ・ソース・オブジェクトを指定します。そのデータ・ソース・オブジェクト内の各列またはフィールドについては、ほとんどの場合、フェデレーテッド・サーバーが DB2 のサポートするデータ・タイプを定義します。非リレーショナルのデータ・ソースの中には、ユーザーが DB2 データ・タイプを指定しなければならないものがあります。

リレーショナル・データ・ソースの場合、どのローカル・データ・タイプを SYSCAT.COLUMNS カタログ・ビューに保管したらよいかを判別するために、フェデレーテッド・サーバーは順方向データ・タイプ・マッピング情報を、ラッパーと SYSCAT.TYPEMAPPINGS カタログ・ビューから探します。
SYSCAT.TYPEMAPPINGS カタログ・ビュー内のマッピングは、ラッパー内のデフォルトのマッピングより優先されます。代替マッピングを作成してデフォルトのデータ・タイプ・マッピングをオーバーライドすると、フェデレーテッド・サーバーはその代替マッピングを使用します。 1 つの列に複数のマッピングが適用されている場合、フェデレーテッド・サーバーは作成された中で最新のマッピングを使用します。

非リレーショナル・データ・ソースの場合、どのローカル・データ・タイプを SYSCAT.COLUMNS カタログ・ビューに保管したらよいかを判別するために、フェデレーテッド・サーバーはデータ・タイプ・マッピング情報をラッパーから探します。非リレーショナル・データ・ソースにより、ラッパーによって定義されたデータ・タイプをユーザーが変更できる度合いは異なります。ユーザーは列を全く指定できず、ラッパーがデータ・タイプを定義する非リレーショナル・データ・ソースもあります。また、ユーザーがデータ・タイプをオーバーライドできるデータ・ソースもあります。さらに別のデータ・ソースでは、ユーザーが CREATE NICKNAME ステートメントに列データ・タイプを指定しなければなりません。

リレーショナル・データ・ソース用に CREATE TABLE 透過 DDL を作成する場合は、ステートメントに DB2 データ・タイプを指定します。フェデレーテッド・サーバーは、フェデレーテッド・データベースとデータ・ソースの間の逆方向データ・タイプ・マッピングの情報を調べます。フェデレーテッド・サーバーはこの情報をラッパーと SYSCAT.TYPEMAPPINGS カタログ・ビューから探します。

データ・ソース列からの値がフェデレーテッド・データベースに戻されるとき、その値はデータ・ソース列にマッピングが適用された後のデータ・タイプである DB2 データ・タイプに完全に一致します。このマッピングがデフォルトのマッピングである場合には、その値はマッピングのデータ・ソース・タイプにも完全に適合します。例えば、FLOAT 列を持つ Oracle 表がフェデレーテッド・データベースに定義されているとすると、この列には Oracle FLOAT から DB2 DOUBLE へのデフォルト・マッピングが自動的に適用されます。そして、その列から戻される値は、FLOAT と DOUBLE の両方のデータ・タイプに完全に適合します。

代替データ・タイプ・マッピングを作成する場合

リレーショナル・データ・ソースには、代替データ・タイプ・マッピングを作成できます。

次のような状況の場合には、代替データ・タイプ・マッピングを作成するとよいでしょう。

- デフォルトのデータ・タイプ・マッピングをオーバーライドする場合。

  いくつかのラッパーについては、戻される値のフォーマットや長さを変更することができます。フォーマットや長さの変更は、値を適合させるべき DB2 データ・タイプを変更して行います。例えば、Oracle DATE データ・タイプはタイム・スタンプとして使用され、それには世紀、年、月、日、時、分、秒が含まれます。デフォルトでは、Oracle DATE データ・タイプは DB2 TIMESTAMP データ・タイプにマップします。時、分、秒の情報だけを戻すには、デフォルトのデータ・タイプ・マッピングをオーバーライドして、 Oracle DATE データ・タイプが DB2 TIME データ・タイプにマップされるようにします。 Oracle DATE 列が照会されると、 Oracle タイム・スタンプ値の時刻の部分だけがフェデレーテッド・サーバーに戻されます。

- デフォルトのマッピングが存在しない場合。

  デフォルトのデータ・タイプ・マッピングを新規データ・ソース・データ・タイプに使用できない場合は、そのデータ・タイプ用にマッピングを作成しなければなりません。

新しいデータ・タイプ・マッピングを定義するには、CREATE TYPE MAPPING ステートメントを使用します。作成したマッピングは、フェデレーテッド・データベースの SYSCAT.TYPEMAPPINGS カタログ・ビューに保管されます。

非リレーショナル・データ・ソースのデータ・タイプ・マッピング

非リレーショナル・データ・ソースの中には、データ・タイプ・マッピングがラッパーにないものもあります。場合によっては、CREATE NICKNAME ステートメントにローカル・タイプ情報を指定することが必要になります。

次の例は、一部の非リレーショナル・データ・ソースについて、 CREATE NICKNAME ステートメントに列データ・タイプを指定する方法を示したものです。

```
CREATE NICKNAME DRUGDATA1
   (Dcode Integer NOT NULL, Drug CHAR(20), Manufacturer CHAR(20))
   FOR SERVER biochem_lab
   OPTIONS (FILE_PATH '/usr/pat/DRUGDATA1.TXT', COLUMN_DELIMITER ',',
   SORTED 'Y', KEY_COLUMN 'DCODE', VALIDATE_DATA_FILE 'Y')
```

順方向および逆方向のデータ・タイプ・マッピング

データ・ソースのデータ・タイプとフェデレーテッド・データベースのデータ・タイプの間のマッピングには、順方向タイプのマッピングと逆方向タイプのマッピングの 2 種類があります。

順方向タイプ・マッピング とは、リモート・データ・タイプから対応するローカル・データ・タイプへのマッピングのことです。データ・ソース・オブジェクトのニックネームを作成するときに、順方向データ・タイプ・マッピングが使用されます。データ・ソース・オブジェクト内の各列の対応するローカル・タイプは、グローバル・カタログに保管されます。

逆方向タイプ・マッピング とは、ローカル・データ・タイプから対応するリモート・データ・タイプへのマッピングのことです。逆方向タイプ・マッピングは透過 DDL とともに使用します。

図 2 は、順方向および逆方向データ・タイプ・マッピングを示したものです。

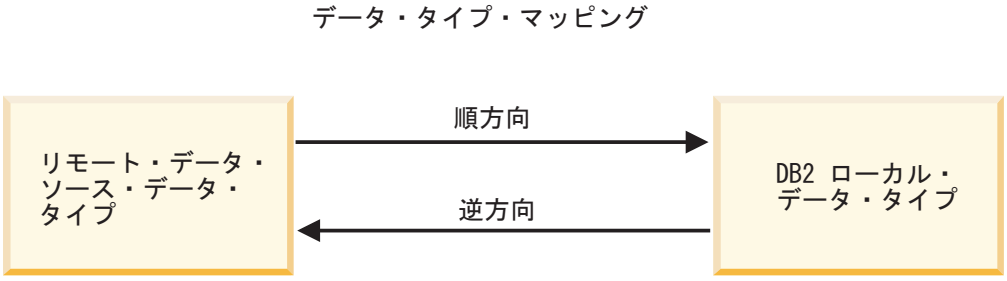


図 2. 順方向および逆方向のデータ・タイプ・マッピング

データ・タイプ・マッピングの作成

CREATE TYPE MAPPING ステートメントを使用して、データ・タイプ・マッピングを作成します。このステートメントは、コマンド行プロセッサーから実行することも、アプリケーション・プログラムに組み込むこともできます。

始める前に

ステートメントに関連付けられた許可 ID が持つ特権として、SYSADM または DBADM 権限が必要です。

このタスクについて

特定の使用に関する情報については、 CREATE TYPE MAPPING ステートメントを参照してください。

制約事項

- *local_data_type* 値を LONG VARCHAR、LONG VARGRAPHIC、またはユーザー定義タイプにすることはできません。
- *data_source_data_type* 値をユーザー定義のタイプにすることはできません。
- 非リレーショナル・データ・ソースの場合、既存のデータ・タイプ・マッピングをオーバーライドしたりマッピングを作成したりする度合いは限定されています。

手順

CREATE TYPE MAPPING ステートメントを実行して、データ・タイプ・マッピングを作成します。
CREATE TYPE MAPPING ステートメントにサーバー・タイプを指定するには、以下のいずれかを *server-type* 値として使用する必要があります。

表 3. 有効なサーバー・タイプ

データ・ソース	サーバー・タイプ
DB2 Database for Linux, UNIX, and Windows	DB2/CS DB2/UDB DB2/NT DB2/SUN DB2/HP DB2/HPUX DB2/AIX DB2/6000 DB2/PE DB2/PTX DB2/SCO DB2/LINUX DB2/EEE DB2/2
DB2 for z/OS	DB2/MVS DB2/ZOS DB2/390
DB2 Server for VSE and VM	DB2/VM DB2/VSE SQL/DS
DB2 for System i	DB2/400 DB2/ISERIES
Oracle	ORACLE
Informix	INFORMIX
ODBC	ODBC
Microsoft SQL Server	MSSQLSERVER
JDBC	JDBC
Teradata	TERADATA
Sybase	SYBASE

データ・ソース・データ・タイプのタイプ・マッピングの作成例

この例では、Oracle NUMBER データ・タイプを使用するすべての Oracle 表およびビューが DB2 DECIMAL(8,2) データ・タイプにマップされるようにします。
Oracle NUMBER データ・タイプは、デフォルトでは、 DB2 DOUBLE データ・タイプという浮動小数点データ・タイプにマップされます。

既存のニックネームのローカル・タイプを変更する場合は ALTER NICKNAME ステートメントを使用してください。ローカル・データ・タイプを DECIMAL(8,2) に

変更するには、それぞれのニックネームに対して個別に変更する必要があります。ニックネームが存在しない場合、そのデータ・ソース・タイプを指定するデータ・タイプ・マッピングを作成します。 CREATE TYPE MAPPING ステートメントを実行する前に、データ・ソース・ラッパーが作成されていることを確認します。

Oracle NUMBER データ・タイプから DB2 DECIMAL(8,2) データ・タイプへのタイプ・マッピングを作成するには、CREATE TYPE MAPPING ステートメントを実行します。以下に例を示します。

```
CREATE TYPE MAPPING MY_ORACLE_DEC FROM SYSIBM.DECIMAL(8,2)
    TO SERVER TYPE ORACLE TYPE NUMBER
```

MY_ORACLE_DEC
: タイプ・マッピングに与える名前です。すでにカタログにあるデータ・タイプ・マッピング名と重複する名前は指定できません。

FROM *SYSIBM.DECIMAL(8,2)*
: ローカル DB2 スキーマおよびローカル・データ・タイプです。長さまたは精度、およびスケールが指定されていない場合、これらの値はソース・データ・タイプから決定されます。

TO SERVER TYPE *ORACLE*
: データ・ソースのタイプ。

TYPE *NUMBER*
: ローカル・データ・タイプにマッピングする元のデータ・ソースのデータ・タイプです。ユーザー定義データ・タイプは使用できません。

Oracle 列に対して DB2 DECIMAL(8,2) がローカル・データ・タイプとして定義されます。 NUMBER 列を含む Oracle 表およびビューのニックネームを作成すると、Oracle NUMBER データ・タイプが DB2 DECIMAL(8,2) データ・タイプにマップされます。

データ・ソース・データ・タイプおよびバージョン別のタイプ・マッピングの作成例

この例では、Oracle 表およびビューがバージョンの異なる Oracle サーバー上に存在しているとします。 Oracle バージョン 8.0.3 サーバー上に存在するすべての表およびビューについて、 Oracle NUMBER(23,2) データ・タイプを使用する列が DB2 DECIMAL(8,2) データ・タイプにマップされるようにします。

Oracle NUMBER(23,3) データ・タイプは、デフォルトでは DB2 DECIMAL(23,3) データ・タイプにマップされるように設定されています。既存のニックネームのローカル・タイプを変更する場合は ALTER NICKNAME ステートメントを使用してください。ローカル・データ・タイプを DECIMAL(8,2) に変更するには、それぞれのニックネームに対して個別に変更する必要があります。ニックネームが存在しない場合、そのデータ・ソース・タイプを指定するデータ・タイプ・マッピングを作成します。 CREATE TYPE MAPPING ステートメントを実行する前に、データ・ソース・ラッパーが作成されていることを確認します。 Oracle バージョン 8.0.3 サーバーの Oracle NUMBER(23,3) データ・タイプを DB2 DECIMAL(8,2) データ・タイプにマップするには、CREATE TYPE MAPPING ステートメントを実行します。以下に例を示します。

```
CREATE TYPE MAPPING ORA_DEC FROM SYSIBM.DECIMAL(8,2)
    TO SERVER TYPE ORACLE VERSION 8.0.3 TYPE NUMBER(23,3)
```

ORA_DEC
タイプ・マッピングに与える名前です。すでにカタログにあるデータ・タイプ・マッピング名と重複する名前は指定できません。

FROM *SYSIBM.DECIMAL(8,2)*
ローカル DB2 スキーマおよびローカル・データ・タイプです。長さまたは精度、およびスケールが指定されていない場合、これらの値はソース・データ・タイプから決定されます。

TO SERVER TYPE *ORACLE*
データ・ソースのタイプ。

VERSION *8.0.3*
データ・ソース・サーバーのバージョン。バージョンを指定する必要があります。この例に示されているように、リリースおよびリリースのモディフィケーションも指定できます。

TYPE *NUMBER(23,3)*
ローカル・データ・タイプにマッピングする元のデータ・ソースのデータ・タイプです。ユーザー定義データ・タイプは使用できません。

フェデレーテッド・データベースは、Oracle バージョン 8.0.3 サーバー上の Oracle 列に対して DB2 DECIMAL(8,2) をローカル・データ・タイプとして定義します。しかし、バージョン 8.0.3 ではないサーバー上の Oracle 表およびビューは、代わりにデフォルトのデータ・タイプ・マッピングを使用します。 NUMBER 列を含む Oracle 表およびビューのニックネームを作成すると、Oracle NUMBER データ・タイプが DB2 DECIMAL(8,2) データ・タイプにマップされます。

サーバーにあるすべてのデータ・ソース・オブジェクト用のタイプ・マッピングの作成例

この例では、サーバーはフェデレーテッド・データベースに対して ORA2SERVER として定義されます。各表には、Oracle DATE データ・タイプの列が含まれます。

Oracle DATE データ・タイプに含まれるのは、世紀、年、月、日、時、分、秒です。 Oracle DATE データ・タイプは、デフォルトではローカル DB2 TIMESTAMP データ・タイプにマップされます。ただし、このサーバー上のオブジェクトを照会するときには、結果セットに時間情報 (時、分、秒) だけが戻されるようにします。

既存のニックネームのローカル・タイプを変更する場合は ALTER NICKNAME ステートメントを使用してください。ローカル・データ・タイプを TIME に変更するには、それぞれのニックネームに対して個別に変更する必要があります。

ニックネームが存在しない場合、そのデータ・ソース・タイプを指定するデータ・タイプ・マッピングを作成します。

例えば、ORA2SERVER について、Oracle DATE データ・タイプを DB2 TIME データ・タイプにマップするには、次のステートメントを発行します。

```
CREATE TYPE MAPPING ORA2_DATE FROM SYSIBM.TIME
   TO SERVER ORA2SERVER TYPE DATE
```

ORA2_DATE
タイプ・マッピングに与える名前です。すでにカタログにあるデータ・タイプ・マッピング名と重複する名前は指定できません。

FROM *SYSIBM.TIME*
ローカル DB2 スキーマおよびローカル・データ・タイプです。長さまたは精度、およびスケールが指定されていない場合、これらの値はソース・データ・タイプから決定されます。

TO SERVER *ORA2SERVER*
データ・ソース・サーバーのローカル名。

TYPE *DATE*
ローカル・データ・タイプにマッピングする元のデータ・ソースのデータ・タイプです。ユーザー定義データ・タイプは使用できません。

フェデレーテッド・データベースは、データ・タイプ DATE の Oracle 列に対して DB2 の TIME をローカル・データ・タイプとして定義します。

DATE 列を含む Oracle 表およびビューのニックネームを作成すると、Oracle DATE データ・タイプが DB2 DECIMAL(8,2) データ・タイプにマップされます。

その他の Oracle サーバーのデータ・ソース・オブジェクトには、このデータ・タイプ・マッピングは適用されません。

データ・タイプ間のキャスト

ソース・データ・タイプからターゲット・データ・タイプにデータ・タイプをキャストできます。

データ・タイプ間のキャストは、暗黙的に行われることも明示的に行われることもあります。

- 暗黙的なキャストでは、暗黙指定された変換規則セットに基づき、特定のデータ・タイプのデータを別のデータ・タイプのデータに自動変換します。この自動変換は、弱いタイプ定義をサポートするために行われます。
- 明示的なキャストは、強いタイプ定義をサポートします。強いタイプ定義では、データ・タイプを一致させることが必要です。比較や割り当てを実行する前に、一方または両方のデータ・タイプを共通データ・タイプに明示的に変換する必要があります。

データ・タイプ間のキャストのフェデレーション・サポートにより、ニックネームを参照するフェデレーテッド照会が、弱いタイプ定義と強いタイプ定義の両方をサポートするサーバーにアクセスできるようになります。

キャスト関数は、リモート・サーバー側に対応する関数がない場合には、リモート・サーバーにプッシュダウンできません。キャストを多用すると、パフォーマンス上の問題につながる可能性があります。

例: 明示的キャスト

```
UPDATE nickname SET varcharcol = CAST(intcol AS varchar(10))

SELECT REAL(varchar_col) FROM nickname1;

SELECT VARCHAR(double_col) FROM nickname1;
```

例: 暗黙的キャスト

例 1:

```
UPDATE nickname SET varcharcol = intcol;
```

この割り当て操作では、リモート・サーバーにプッシュダウンされるステートメントは以下のようになります。

```
UPDATE nickname SET varcharcol = varchar(intcol);
```

例 2:

```
INSERT INTO nickname (varcharcol) SELECT intcol FROM nickname1;
```

この割り当て操作では、リモート・サーバーにプッシュダウンされるステートメントは以下のようになります。

```
INSERT INTO nickname (varcharcol) SELECT varchar(intcol) FROM nickname1;
```

例 3:

```
SELECT * SELECT nickname SELECT intcol = varcharcol;
```

この比較操作では、リモート・サーバーにプッシュダウンされるステートメントは以下のようになります。

```
SELECT * SELECT nickname SELECT intcol = CAST(varcharcol AS decfloat)
```

TIMESTAMP データ・タイプのサポート

TIMESTAMP データ・タイプは小数秒の精度まで制御するためにパラメーター化されました。

DB2 for Linux, UNIX, and Windows データ・ソースでは、リモート・タイム・スタンプは TIMESTAMP(*p*) にマップされます。ここで、*p* は精度を表し、秒の小数部分の桁数を指定します。 *p* の範囲は、0 以上 12 以下です。

それ以外のデータ・ソースでは、リモート・タイム・スタンプはデフォルトの精度 6 の TIMESTAMP にマップされます。これらのデータ・ソースでは、デフォルトの順方向タイプ・マッピング用に提供されているサンプルに類似したマッピングを使用して、TIMESTAMP(*p*) サポートを利用することができます。

ニックネーム TIMESTAMP 列の精度が、それに対応するリモート表の列の精度よりも小さく、リモート表の列の余分な小数桁にゼロ以外の数字が含まれている場合、そのニックネーム列を使用する述部の結果は予測不能になることがあります。そのニックネーム列を使用する述部の結果は、述部をプッシュダウンしない場合には正しく機能します。

データ・ソース・オブジェクトのローカル・タイプの変更

ローカル・データ・タイプを変更するには、CREATE TYPE MAPPING ステートメントではなく、ALTER NICKNAME ステートメントを使用します。

始める前に

ステートメントを発行する許可 ID には、以下の特権が少なくとも 1 つ含まれていなければなりません。

- SYSADM または DBADM 権限
- ステートメントで指定されたニックネームに対する ALTER 特権
- ステートメントで指定されたニックネームに対する CONTROL 特権
- ニックネームにスキーマ名がある場合は、スキーマに対する ALTERIN 特権
- ニックネームのカタログ・ビューの DEFINER 列に記録された、ニックネームの定義者

このタスクについて

ニックネームを作成すると、データ・ソース・オブジェクトに関連付けられているデータ・タイプは、フェデレーテッド・データベースに保管されます。データ・ソースには、ラッパーが自動でデータ・タイプを指定するものと、ニックネーム作成時にユーザーがデータ・タイプを指定しなければならないものがあります。

特定のデータ・ソース・オブジェクトの列に、ローカル・タイプを指定することができます。

重要: 列のローカル・データ・タイプを、対応するリモート・タイプとは大きく異なるタイプに変更すると、結果としてエラーになったり、情報が失われてしまう場合があります。

制約事項

28 ページの『ニックネームの変更に関する制限』を参照してください。

手順

コマンド行プロンプトからローカル・データ・タイプを変更するには、コマンド行から ALTER NICKNAME ステートメントを発行します。
例えば、以下のように指定します。

```
ALTER NICKNAME nickname ALTER COLUMN column_name
   LOCAL TYPE data_type
```

文字データを含むローカル列の内容をビット (バイナリー) データとして扱う場合は、 ALTER NICKNAME ステートメントに FOR BIT DATA 文節を使用します。この文節を使用して列のローカル・データ・タイプを変更した場合、他のシステムとデータ交換を行ったときには、コード・ページ変換が行われません。リモート・データベースの照合シーケンスに関係なく、比較はバイナリーで行われます。

データ・ソース・オブジェクトのローカル・タイプの変更例

このトピックには、データ・ソース・オブジェクトのデータ・タイプを変更する方法を示す例が記載されています。

例: 数値データ・タイプ・マッピング

従業員情報の Oracle 表で、BONUS 列がデータ・タイプ NUMBER(32,3) で定義されているとします。 Oracle データ・タイプ NUMBER(32,3) は、デフォルトでは DB2 データ・タイプ DOUBLE (倍精度浮動小数点数データ・タイプ) にマップされます。 BONUS 列を含む照会を実行すると、次のような値が戻されます。

```
5.0000000000000E+002
1.0000000000000E+003
```

浮動小数は、小数点を移動する桁数と方向を指示します。この例で、+002 は小数点の位置を右に 2 桁移動することを意味し、 +003 は右に 3 桁移動することを意味します。

BONUS 列を含む照会が戻す値を、金額らしくすることもできます。その場合、表の BONUS 列のローカル定義を DOUBLE データ・タイプから DECIMAL データ・タイプに変更します。実際の賞与の形式を反映する精度と位取りを使用します。例えば、賞与のドル金額 (整数部分) が 6 桁を超えないのであれば、NUMBER(32,3) を DECIMAL(8,2) にマップします。この新しいローカル・タイプの制約によって、 BONUS 列を含む照会からは値が次のような形式で戻されます。

```
 500.00
1000.00
```

Oracle 表のニックネームは ORASALES です。 ORASALES 表の BONUS 列を DB2 DECIMAL (8,2) データ・タイプにマップするには、次の ALTER NICKNAME ステートメントを発行します。

```
ALTER NICKNAME ORASALES ALTER COLUMN BONUS
   LOCAL TYPE DECIMAL(8,2)
```

ORASALES
: Oracle 表に定義したニックネーム。

ALTER COLUMN *BONUS*
: フェデレーテッド・データベース SYSCAT.COLUMNS カタログ・ビューにローカルに定義した列の名前。

LOCAL TYPE *DECIMAL(8,2)*
: 列の新しいローカル・タイプであることを表します。

このマッピングは、ニックネーム ORASALES で識別される Oracle 表の BONUS 列にのみ適用されます。 BONUS 列が含まれるその他のすべての Oracle データ・ソース・オブジェクトは、 Oracle NUMBER データ・タイプのデフォルトのデータ・タイプ・マッピングを使用します。

例: 日付データ・タイプ・マッピング

Oracle 表 SALES のニックネームは ORASALES です。 SALES 表には Oracle DATE データ・タイプの列が 1 つ含まれています。デフォルトのタイプ・マッピングでは、 Oracle DATE データ・タイプは DB2 TIMESTAMP データ・タイプにマ

ップされます。しかし、この列からデータを検索して表示したいのは、日付値だけであるとします。この場合、SALES 表のニックネームを変更し、ローカル・タイプを DB2 DATE データ・タイプに変更することができます。

```
ALTER NICKNAME ORASALES ALTER COLUMN ORDER_DATE
   LOCAL TYPE DATE
```

例: 非リレーショナル・データ・ソースのデータ・タイプ・マッピング

drugdata1.txt という名前の表構造ファイルのニックネームは DRUGDATA1 です。drugdata1.txt ファイルには医薬品名をリストした列が含まれており、その列の名前は DRUG です。 DRUG 列は当初 CHAR(20) で定義されましたが、列の長さを CHAR(30) に変更することが必要になりました。この場合、drugdata1.txt ファイルのニックネームを次のようにして変更し、マッピングを正しい長さに変更することができます。

```
ALTER NICKNAME DRUGDATA1 ALTER COLUMN DRUG
   LOCAL TYPE CHAR(30)
```

LONG データ・タイプの VARCHAR データ・タイプへの変更

LONG データ・タイプへの挿入および更新操作を使用可能にするために、 LONG データ・タイプを VARCHAR データ・タイプに変更することができます。

表 4 には、変更可能な LONG データ・タイプがデータ・ソース別にリストされています。

表 4. VARCHAR データ・タイプに変更可能な LONG データ・タイプ (データ・ソース別)

データ・ソース	リモート・データ・タイプ	長さ	デフォルトのローカル・データ・タイプ	VARCHAR への変更
DRDA	LONG VARCHAR	1–32672	CLOB	VARCHAR
	LONG VARCHAR FOR BIT DATA	1–32672	BLOB	VARCHAR FOR BIT DATA
Informix	BYTE	1–32672	BLOB	VARCHAR FOR BIT DATA
	TEXT	1–32672	CLOB	VARCHAR
Microsoft SQL Server	IMAGE	1–32672 のホスト変数、1–8000 のリテラル	BLOB	VARCHAR FOR BIT DATA
	TEXT	1–32672 のホスト変数、1–8000 のリテラル	CLOB	VARCHAR
Oracle NET8	LONG	1–32672 のホスト変数、1–4000 のリテラル	CLOB	VARCHAR

表 4. VARCHAR データ・タイプに変更可能な LONG データ・タイプ (データ・ソース別) (続き)

データ・ソース	リモート・データ・タイプ	長さ	デフォルトのローカル・データ・タイプ	VARCHAR への変更
	LONG RAW	1–32672 のホスト変数、1–4000 のリテラル	BLOB	VARCHAR FOR BIT DATA
Sybase CTLIB	IMAGE	1–32672	BLOB	VARCHAR FOR BIT DATA
	TEXT	1–32672	CLOB	VARCHAR
Teradata	BYTE	32673–64000	BLOB	VARCHAR FOR BIT DATA(32672)
	CHAR	32673–64000	CLOB	VARCHAR(32672)
	VARBYTE	32673–64000	BLOB	VARCHAR FOR BIT DATA(32672)
	VARCHAR	32673–64000	CLOB	VARCHAR(32672)

第 4 章 関数およびユーザー定義関数のマッピング

関数マッピングは、フェデレーテッド・サーバー関数とユーザー定義関数をデータ・ソースの既存の関数と関連付けます。

フェデレーテッド・システムでの関数マッピング

IBM InfoSphere Federation Server は、既存のデータ・ソース関数と DB2 側の対応する関数の間のデフォルトのマッピングを備えています。

フェデレーテッド・サーバーがデータ・ソース関数を使用するには、DB2 関数または関数テンプレートからデータ・ソース関数へのマッピングが必要です。

ラッパー・モジュール内にデフォルトの関数マッピングがあります。

非リレーショナル・データ・ソースの場合、既存の関数マッピングをオーバーライドしたり新規のマッピングを作成したりすることはできません。

フェデレーテッド・システムでの関数マッピングの作用

1 つ以上の関数を含むフェデレーテッド・サーバーに照会をサブミットすると、フェデレーテッド・サーバーは DB2 関数とデータ・ソース関数の間のマッピングについての情報を調べます。

フェデレーテッド・サーバーはマッピングの情報を求めて 2 つの場所を調べます。

- ラッパー。データ・ソースのラッパーには、デフォルトの関数マッピングが含まれます。
- SYSCAT.FUNCMAPPINGS カタログ・ビュー。このビューには、ラッパーにあるデフォルトの関数マッピングをオーバーライドまたは増強する、ユーザーが作成する項目が含まれます。これには、デフォルトの関数マッピングがないときにユーザーが作成する新規のマッピングも含まれます。関数に複数のマッピングを適用できるときは、最近作成したものが適用されます。

関数マッピングのオプションは関数についての情報、およびデータ・ソースでの関数の処理にかかる可能性のあるコストについての情報を指定します。関数マッピング・オプションは次のような情報を提供します。

- リモート・データ・ソース関数の名前
- データ・ソース関数が呼び出された最初の時および最後の時に処理される命令数の推定値
- データ・ソース関数が呼び出された最初の時および最後の時に実行される入出力数の推定値
- データ・ソース関数の呼び出しごとに処理される命令数の推定値

関数マッピングを作成するときは、DB2 関数または関数テンプレートをデータ・ソースの側の対応する関数にマッピングします。 DB2 側の対応する関数が存在しな

いか、あるいはフェデレーテッド・サーバーにデータ・ソース関数の使用を強制する場合には、その役割を果たす関数テンプレートを作成することができます。

関数マッピングが重要である理由

関数マッピングは、照会オプティマイザーによって行われるプッシュダウン分析に対するいくつかの重要な入力のうちの 1 つです。

照会オプティマイザーは、最適な照会アクセス・プランの決定にあたって、特定タイプの SQL 関数または操作に関するデータ・ソースの実行能力を因子に分解します。関数がマッピングを持たない場合、その関数がデータ・ソースに送られて処理されることはありません。関数や他の操作をデータ・ソースにプッシュダウンできれば、パフォーマンスが向上します。

データ・ソースが DB2 関数と類似した関数を持っていて戻される結果の差がわずかである場合は、関数マッピングを作成することによりパフォーマンスが向上する場合があります。例えば、Informix STDEV (標準偏差) 関数が生成する結果は、入力データ・セットによっては DB2 STDDEV 関数と異なります。そのため、Informix ラッパーは、この 2 つの関数のデフォルトのマッピングを持ちません。結果セットに差があってもかまわなければ、Informix データ・ソースにアクセスし、DB2 STDDEV 関数を使用する照会のパフォーマンスが向上する可能性があります。 Informix STDEV と DB2 STDDEV 関数の関数マッピングを作成することにより、照会オプティマイザーは、この関数の処理をデータ・ソースに投げることも選択肢として持てるようになります。

関数マッピングを作成する場合

デフォルトの関数マッピングをデータ・ソース関数に使用できないときは、関数マッピングを作成できます。

関数マッピングを使用できない 1 つの理由として、フェデレーテッド・データベースがデータ・ソース関数に対応する関数を持っていないということが挙げられます。

マッピングを使用できないもう一つの理由として、データ・ソースが DB2 関数と類似した関数を持っていても、その関数が同じ結果を戻さないということがあります。ある入力データ・セットに対してデータ・ソースが少しでも異なる結果を戻す場合、ラッパーは通常、そうした関数にマップしません。ただし、結果セットに差があってもかまわない場合は、それらの関数間のマッピングを作成できます。マッピングを作成することによって、パフォーマンスが向上する場合があります。

以下の場合に関数マッピングを使用します。

- データ・ソースで新しい組み込み関数が使用できるようになった。
- データ・ソースで新しいユーザー定義関数が使用できるようになった。
- 対応する DB2 関数がない。
- 対応する関数があり、戻される結果は少し異なるが、それでもかまわない。

関数マッピングの設定は、SYSCAT.FUNCMAPPINGS カタログ・ビューに保管されます。

関数マッピングを作成した場合、データ・ソースで評価された関数からの戻り値と、フェデレーテッド・データベースで評価された互換関数からの戻り値が、異なる可能性があります。 IBM InfoSphere Federation Server は関数マッピングを使用しますが、SQL 構文エラーまたは予期しない結果になる可能性があります。

アプリケーションでのユーザー定義関数

アプリケーション開発者は、それぞれのアプリケーションまたはドメインに固有の、独自の一連の関数を作成する必要があることがしばしばあります。この目的には、ユーザー定義のスカラー関数を使用できます。

例えば、小売店は、販売している品目のコストを追跡するために PRICE データ・タイプを定義できます。この店は、SALES_TAX 関数も定義する必要があるかもしれません。この関数は、与えられた価格値を入力として該当する消費税を計算し、そのデータを要求ユーザーまたはアプリケーションに戻すというものです。

こうした関数は、ラージ・オブジェクト・タイプや特殊タイプを含むすべてのデータベース・タイプで機能することができます。ユーザー定義関数を使用すれば、照会の中に計算と検索の強力な述部を含めることができるので、データのソースに近い無関係のデータをフィルターに掛けることによって、応答時間を短縮できるようになります。 SQL オプティマイザーは、ユーザー定義関数を SUBSTR や LENGTH などの組み込み関数とまったく同様に扱います。アプリケーションの開発は、C、C++、および COBOL といったさまざまなアプリケーション言語環境を使用して行えます。異なるアプリケーション言語環境を使用して開発されるアプリケーション間でも、SQL ユーザー定義関数のセットを共用できます。

ユーザー定義関数は、データを操作し、アクションを実行できます。例えば、ユーザー定義関数で、電子メッセージを送信したり、フラット・ファイルを更新したりできます。

DB2 では、ユーザー定義関数として以下の関数があります。

- ユーザーが最初から定義する関数。
- SYSFUN スキーマに含まれる関数。例えば、SIN、COS、TAN のような数学関数、RADIANS、LOG10、POWER のような科学関数、
  LEFT、DIFFERENCE、UCASE のような汎用関数があります。

ユーザー定義関数のマッピングの要件

フェデレーテッド・システムでデータ・ソース・ユーザー定義関数を呼び出す前に、フェデレーテッド・データベースはフェデレーテッド・サーバーのグローバル・カタログに保管されている関数の仕様にデータ・ソース関数を関連付ける必要があります。

フェデレーテッド・データベースが関数の仕様をデータ・ソース関数に関連付ける際には、次の 2 つの条件があります。

- フェデレーテッド・データベースが、そのシグニチャーがデータ・ソース関数のものと一致する関数を持つ。シグニチャーは、関数名と関数入力パラメーターで構成されます。シグニチャーは、次の両方の条件が真の場合に一致します。
  - シグニチャーの名前、およびパラメーターの数が同じである。

– 一方のシグニチャーの各パラメーターのデータ・タイプが、他方のシグニチャーの対応するパラメーターのデータ・タイプと同じ (または変換可能) である。

- フェデレーテッド・データベースが必要なシグニチャーがある関数を持たない場合、このシグニチャーを含む関数テンプレートを定義できる。その後、呼び出すデータ・ソース関数にその関数テンプレートをマップします。

関数マッピングの設定は、SYSCAT.FUNCMAPPINGS カタログ・ビューに保管されます。

関数マッピングの作成

CREATE FUNCTION MAPPING ステートメントを使用して、デフォルトの関数マッピングをオーバーライドする代替関数マッピングを指定します。

代替関数マッピングを作成すると、項目は SYSCAT.FUNCMAPPINGS カタログ・ビューに現れます。

CREATE FUNCTION MAPPING ステートメントを使用して、関数マッピング・オプションを指定することもできます。関数マッピング・オプションを指定すると、その情報は SYSCAT.FUNCMAPOPTIONS カタログ・ビューに現れます。

CREATE FUNCTION MAPPING ステートメントを使用して、以下の処理を行えます。

- 特定のタイプのすべてのデータ・ソース用の関数マッピングを作成する。例えば、すべての Informix データ・ソースを対象としたものです。
- 特定のタイプとバージョンのすべてのデータ・ソース用の関数マッピングを作成する。例えば、すべての Informix 9 データ・ソースを対象としたものです。
- 特定のサーバー用の関数マッピングを作成する。
- 関数マッピングの統計情報をオプティマイザーに提供する。
- デフォルトの関数マッピングまたはユーザーが定義した関数マッピングを使用不可にする。

CREATE FUNCTION MAPPING ステートメントは、コマンド行プロセッサーで発行できます。 CREATE FUNCTION MAPPING ステートメントをアプリケーション・プログラムに組み込むこともできます。DB2 コントロール・センターは、関数マッピングの作成または変更をサポートしていません。

CREATE FUNCTION MAPPING ステートメントでの関数名の指定

CREATE FUNCTION MAPPING ステートメントに入力する値は、一緒にマップされる関数の名前が同じか異なるかに依存します。

ステートメントの許可 ID が保持する特権は、SYSADM または DBADM 権限でなければなりません。

名前が同じ関数のマッピング

名前が同じ 2 つの関数 (または、DB2 関数テンプレートとデータ・ソース関数) 間のマッピングを作成できます。

手順

名前が同じ 2 つの関数をマップするには、CREATE FUNCTION MAPPING ステートメントを発行します。
例: Informix データ・ソースの MYFUN という名前のユーザー定義関数を、TINA.MYFUN という名前の DB2 ユーザー定義関数にマップするとします。Informix データ・ソース・サーバーの名前は INFORMIX2 です。次のステートメントで関数がマップされます。

```
CREATE FUNCTION MAPPING FOR TINA.MYFUN(SYSTEM.INTEGER) SERVER INFORMIX2
```

名前が異なる関数のマッピング

名前が異なる 2 つの関数 (または、DB2 関数テンプレートとデータ・ソース関数) 間のマッピングを作成できます。

このタスクについて

名前が異なる 2 つの関数のマッピングを作成するには、CREATE FUNCTION MAPPING ステートメントを以下のようにして発行します。

手順

1. DB2 関数または関数テンプレートの名前を function_name パラメーターに割り当てます。
2. REMOTE_NAME という関数マッピング・オプションを指定し、このオプションにデータ・ソース関数の名前を割り当てます。 REMOTE_NAME は、255 文字より少なくなくてはなりません。

例

例: Oracle データ・ソースの UPPERCASE という名前のユーザー定義関数を、DB2 関数 UCASE(CHAR) にマップするとします。Oracle データ・ソース・サーバーの名前は ORACLE2 です。この関数マッピングの名前を ORACLE_UPPER にします。構文は、次のようになります。

```
CREATE FUNCTION MAPPING ORACLE_UPPER FOR SYSFUN.UCASE(CHAR)
    SERVER ORACLE2 OPTIONS
    (REMOTE_NAME 'UPPERCASE')
```

特定のデータ・ソース・タイプ用の関数マッピングの作成

特定のタイプのすべてのデータ・ソース用の関数へのマッピングを作成できます。

始める前に

ステートメントの許可 ID が保持する特権は、SYSADM または DBADM 権限でなければなりません。

このタスクについて

制約事項

既存の関数マッピングをオーバーライドしたり、非リレーショナル・データ・ソース用の新しいマッピングを作成したりすることはできません。

手順

DB2 関数テンプレートをデータ・ソース関数にマップするには、CREATE FUNCTION MAPPING ステートメントを使用します。

例: DB2 関数テンプレートを、すべての Oracle データ・ソース用の Oracle ユーザー定義関数にマップする

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN1
     FOR NOVA.STATS ( DOUBLE, DOUBLE )
     SERVER TYPE ORACLE
     OPTIONS (REMOTE_NAME 'STAR.STATISTICS')
```

テンプレートの名前は STATS で、NOVA というスキーマに属します。Oracle ユーザー定義関数の名前は STATISTICS で、STAR というスキーマに属します。

特定のデータ・ソース・タイプおよびバージョン用の関数マッピングの作成

特定のバージョンのデータ・ソース・タイプを使用するすべてのデータ・ソース用の関数へのマッピングを作成できます。

始める前に

ステートメントの許可 ID が保持する特権は、SYSADM または DBADM 権限でなければなりません。

このタスクについて

制約事項

既存の関数マッピングをオーバーライドしたり、非リレーショナル・データ・ソース用の新しいマッピングを作成したりすることはできません。

手順

特定のデータ・ソース・タイプおよびバージョン用のマッピングを作成するには、CREATE FUNCTION MAPPING ステートメントを使用します。

例: DB2 関数テンプレートを、バージョン 15 を使用するすべての Sybase データ・ソース用の Sybase ユーザー定義関数にマップする

テンプレートの名前は SYB_STATS で、EARTH というスキーマに属します。Sybase ユーザー定義関数の名前は STATISTICS で、MOON というスキーマに属します。CREATE FUNCTION MAPPING は、次のようになります。

```
CREATE FUNCTION MAPPING SYBASE_STATS
   FOR EARTH.SYB_STATS ( DOUBLE, DOUBLE )
   SERVER TYPE SYBASE VERSION 15
   OPTIONS (REMOTE_NAME 'MOON.STATISTICS')
```

特定のサーバーのすべてのデータ・ソース・オブジェクト用の関数マッピングの作成

特定のリモート・サーバーのすべてのデータ・ソース・オブジェクトで使用される関数へのマッピングを作成できます。

始める前に

ステートメントの許可 ID が保持する特権は、SYSADM または DBADM 権限でなければなりません。

このタスクについて

制約事項

既存の関数マッピングをオーバーライドしたり、非リレーショナル・データ・ソース用の新しいマッピングを作成したりすることはできません。

手順

特定のサーバーのすべてのデータ・ソース・オブジェクト用の関数マッピングを作成するには、CREATE FUNCTION MAPPING ステートメントを使用します。

例: BONUS という関数テンプレートを、BONUS というユーザー定義関数にマップする

ORA_SALES という Oracle データ・ソース・サーバーのみに適用するマッピングが必要であるとします。関数名が同じであるため、REMOTE_NAME 関数マッピング・オプションを指定する必要はありません。

```
CREATE FUNCTION MAPPING BONUS_CALC FOR BONUS()
 SERVER ORA_SALES
```

関数テンプレート

フェデレーテッド・サーバーは、データ・ソース関数とフェデレーテッド・データベースにある DB2 の側の対応する関数の間にマッピングが存在する場合にデータ・ソース関数を認識します。

DB2 側の対応する関数が存在しない場合には、その役割を果たす関数テンプレートを作成することができます。

関数テンプレート は、フェデレーテッド・サーバーにデータ・ソース関数の呼び出しを強制する目的で作成される DB2 関数です。ただし、通常の関数とは異なり、関数テンプレートには実行可能コードがありません。フェデレーテッド・サーバーが関数テンプレートを指定する照会を受け取ると、フェデレーテッド・サーバーはデータ・ソース関数を呼び出します。

関数テンプレートは、AS TEMPLATE パラメーターを使った CREATE FUNCTION ステートメントによって作成されます。

関数テンプレートを作成したら、次にテンプレートとデータ・ソース関数の間の関数マッピングを作成する必要があります。関数マッピングは CREATE FUNCTION MAPPING ステートメントを使用して作成されます。

関数テンプレートの作成

フェデレーテッド・サーバーがデータ・ソース関数を認識するのは、データ・ソース関数とフェデレーテッド・データベースの対応する関数のマッピングがある場合です。関数テンプレートを作成することによって、対応するものがないときにその代わりにすることができます。

始める前に

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- SYSADM または DBADM 権限
- データベースに対する IMPICIT_SCHEMA 権限 (関数の暗黙的または明示的スキーマ名が存在しない場合)
- 関数のスキーマ名が存在する場合は、スキーマに対する CREATEIN 特権

このタスクについて

制約事項

データ・ソース関数が入力パラメーターを持つ場合は、以下の制約事項があります。

- 対応する DB2 関数が持つ入力パラメーターの数は、データ・ソース関数と同じでなければなりません。
- 対応する DB2 関数の入力パラメーターのデータ・タイプは、データ・ソース関数の入力パラメーターの該当データ・タイプと互換性がなければなりません。LONG VARCHAR、LONG VARGRAPHIC、またはユーザー定義タイプは、データ・タイプとして使用できません。

データ・ソース関数が入力パラメーターを持たない場合、対応する DB2 関数は入力パラメーターを持つことができません。

手順

1. CREATE FUNCTION ステートメントを AS TEMPLATE パラメーターを指定して使用します。 例:

```
CREATE FUNCTION BONUS ()
  RETURNS DECIMAL(8,2)
  AS TEMPLATE
  DETERMINISTIC
  NO EXTERNAL ACTION
```

BONUS ()
関数テンプレートに付ける名前。

RETURNS *DECIMAL(8,2)*
出力のデータ・タイプ。

AS TEMPLATE
これが関数ではなく関数テンプレートであることを指示します。

DETERMINISTIC
関数は、ある特定の引数値セットに対して常に同じ結果を戻すことを指定します。

NO EXTERNAL ACTION
関数は、データベース・マネージャーが管理しないオブジェクトに外的影響を与えないことを指定します。

DETERMINISTIC および NO EXTERNAL ACTION 文節は、この関数自体が deterministic 関数であるかどうか、および外部アクションにつながるかどうかによって指定する必要があります。そうしないと、この関数テンプレートでサポートされる SQL 操作に制限が出てきます。

2. 関数テンプレートを作成したら、次にテンプレートとデータ・ソース関数の間の関数マッピングを作成する必要があります。関数マッピングは CREATE FUNCTION MAPPING ステートメントを使用して作成されます。 例:

```
CREATE FUNCTION MAPPING MY_INFORMIX_FUN FOR BONUS()
  SERVER TYPE INFORMIX OPTIONS (REMOTE_NAME 'BONUS()')
```

MY_INFORMIX_FUN
関数マッピングに付ける名前。フェデレーテッド・データベースのグローバル・カタログにすでに記述されている関数マッピング名と重複する名前は使用できません。これは固有でなければなりません。

FOR *BONUS()*
ローカル DB2 関数テンプレート名。データ・タイプ入力パラメーターを括弧で囲みます。

SERVER TYPE *INFORMIX*
マップ先の関数が含まれるデータ・ソースのタイプを特定します。

OPTIONS *(REMOTE_NAME 'BONUS()')*
ローカル DB2 関数テンプレートにマップするリモート・データ・ソース関数の名前を特定するためのオプション。

デフォルトの関数マッピングの使用不可化

デフォルトの関数マッピングをドロップすることはできません。ただし、使用不可にすることによって作動不能にすることができます。

始める前に

ステートメントの許可 ID が保持する特権は、SYSADM または DBADM 権限でなければなりません。

手順

デフォルトの関数マッピングを使用不可にするには、CREATE FUNCTION MAPPING ステートメントで DB2 関数の名前を指定し、DISABLE オプションを

'Y' に設定します。

例: DB2 SIN 関数と Oracle データ・ソースの類似関数のデフォルトの関数マッピングを使用不可にする

Oracle データを要求し SIN を参照する照会が処理されるとき、どちらの関数も呼び出される可能性があります。呼び出される関数は、必要とするオーバーヘッドがどちらの関数が小さいと照会オプティマイザーが見積もったかによります。

DB2 SIN 関数が呼び出されるようにして Oracle SIN 関数が呼び出されないようにするには、デフォルトの関数マッピングを使用不可にする必要があります。次の構文を使用します。

```
CREATE FUNCTION MAPPING FOR SYSFUN.SIN(INT)
TYPE ORACLE OPTIONS (DISABLE 'Y')
```

関数マッピングのドロップ

作成した関数マッピングが必要なくなったら、**DROP** ステートメントを実行することにより、その関数マッピングを削除できます。

始める前に

ステートメントの許可 ID が保持する特権は、SYSADM または DBADM 権限でなければなりません。

このタスクについて

作成してデフォルトの関数マッピングをオーバーライドした関数マッピングをドロップすると、デフォルトの関数マッピングが使用されるようになります。

関数マッピングは SYSCAT.FUNCMAPPINGS カタログ・ビューにリストされます。

手順

作成した関数マッピングをドロップするには、DROP ステートメントを使用します。

例: BONUS_CALC という関数マッピングをドロップする

```
DROP FUNCTION MAPPING BONUS_CALC
```

第 5 章 索引仕様の作成

リレーショナル・データ・ソースの場合、索引仕様を作成して、グローバル・カタログのリモート索引の列についての情報を保管し、検索照会のパフォーマンスが最適になるようにします。

フェデレーテッド・システムでの索引仕様

フェデレーテッド・システムで、ニックネームを指定した CREATE INDEX ステートメントを使用して、リモート・オブジェクトの索引の可用性に関する情報をグローバル・カタログに保管します。照会オプティマイザーはこの情報を使用して、照会を最適化します。

CREATE INDEX ステートメントを発行すると、以下の処理が行われます。

- 表のニックネームが作成されている場合、CREATE INDEX ステートメントは、リモート表の作成済み索引に関する索引情報を収集します。
- ビューのニックネームが作成されている場合、CREATE INDEX ステートメントは、ビューのニックネームを参照し、ビューの基礎表の索引に関する情報を収納します。

フェデレーテッド・サーバーは、リモート索引を構成する列とそのユニーク性プロパティーを、「索引仕様」から知ることができます。索引キーのユニーク値の数など、索引の統計的プロパティーを、フェデレーテッド・サーバーがここから知ることはできません。

ニックネームが作成されたときにリモート索引が配備されていた場合は、索引仕様を指定する必要はありません。

フェデレーテッド・サーバーは、次のものにニックネームが作成された場合、「索引仕様」を作成しません。

- 索引を持たない表
- ビュー。通常、ビューにはリモート・カタログに保管される索引情報はありません
- フェデレーテッド・サーバーが索引情報を入手できるリモート・カタログを持たない、データ・ソース・オブジェクト

ニックネームの作成時にあった索引に加えて、新しい索引が表に追加されたとします。索引情報はニックネーム作成時にグローバル・カタログに入るため、フェデレーテッド・サーバーは新しい索引については認識していません。同様に、ビューのニックネームを作成しても、フェデレーテッド・サーバーはそのビューの基になる表 (およびその索引) を認識してはいません。このような場合には、ユーザーが必要な索引情報をグローバル・カタログに入れることができます。索引を持たない表の「索引仕様」を作成することができます。「索引仕様」は、データを速く見付けるには表内のどの列を検索すべきかを照会オプティマイザーに示します。

「索引仕様」は、リレーショナル・データ・ソースで使用してください。非リレーショナル・データ・ソースの「索引仕様」を作成しても、パフォーマンスは向上しません。

データ・ソース・オブジェクトの索引仕様の作成

データ・ソース表のニックネームを作成すると、そのデータ・ソース表が持つ索引に関する情報が、フェデレーテッド・サーバーによってグローバル・カタログに入れられます。オプティマイザーはこの情報を使用して、分散要求の処理を効率よく行います。メタデータの集まりであるこの情報は、*索引仕様* と呼ばれます。

始める前に

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- SYSADM または DBADM 権限
- オブジェクトに対する CONTROL 特権またはオブジェクトに対する INDEX 特権のいずれか。および、データベースに対する IMPLICIT_SCHEMA 権限 (索引の暗黙的または明示的スキーマ名が存在しない場合)、またはスキーマに対する CREATEIN 特権 (索引のスキーマ名が既存スキーマを指す場合) のいずれか。

このタスクについて

フェデレーテッド・サーバーは、以下の場合は索引仕様を作成しません。

- 索引を持たない表のニックネームが作成された。
- 索引を含まないデータ・ソース・オブジェクト (ビューや Informix シノニムなど) のニックネームが作成された。
- 非リレーショナル・オブジェクト (例えば、表構造ファイル、Excel スプレッドシート、XML タグ付きファイルなど) のニックネームが作成された。
- リモート索引が LOB 列または XML 列にある。
- リモート索引に含まれるキーの長さの合計が 1024 バイトを超えている。
- キー・パーツの最大数が 16 を超えている。

これらの場合、フェデレーテッド・サーバーはデータ・ソース・オブジェクトの索引仕様を保管しません。ただし、このリストの最初の 2 つの項目については、ユーザーが必要な索引情報をグローバル・カタログに入れることができます。索引情報を指定するには、CREATE INDEX ステートメントを使用します。

制約事項

ニックネームの索引仕様を作成する際に、いくつかの制約事項があります。

- BIND オプション DYNAMICRULES BIND を適用すると、ステートメントを動的に準備できなくなります。また、CREATE INDEX ステートメントに INCLUDE、CLUSTER、PCTFREE、 MINPCTUSED、DISALLOW REVERSE SCANS、ALLOW REVERSE SCANS パラメーターを使用できません。
- UNIQUE の指定は、索引キーのデータにデータ・ソース表の各行のユニーク値が含まれる場合のみ行ってください。ユニーク性はチェックされません。
- 指定した列の保管長さの合計が 1024 を超えることはできません。

- LOB 列、あるいは LOB を基にした特殊タイプ列を索引の一部として使用することはできません。列の長さ属性が小さく 1024 バイトの制限範囲に入る場合でも、この制約が適用されます。

手順

索引を作成するには、アプリケーション・プログラムに CREATE INDEX ステートメントを組み込むか、あるいはこのステートメントをコマンド行から動的 SQL ステートメントとして発行します。

CREATE INDEX ステートメントをニックネームを指定して使用すると、フェデレーテッド・グローバル・カタログに索引仕様が作成され、データ・ソース表の索引は作成されません。

索引仕様を作成するには、次の構文を使用します。

```
CREATE INDEX index_name ON nickname
(column_name) SPECIFICATION ONLY

CREATE UNIQUE INDEX index_name ON nickname
(column_name DESC) SPECIFICATION ONLY
```

索引仕様の場合、*column_name* は、フェデレーテッド・サーバーがデータ・ソース表の列を参照するときに使用する名前です。

新しい索引を取得する表の索引仕様の作成

表が新しい索引を取得する場合は、その表に対応するニックネームの索引仕様を作成する必要があります。

始める前に

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- SYSADM または DBADM 権限
- オブジェクトに対する CONTROL 特権またはオブジェクトに対する INDEX 特権のいずれか。および、データベースに対する IMPLICIT_SCHEMA 権限 (索引の暗黙的または明示的スキーマ名が存在しない場合)、またはスキーマに対する CREATEIN 特権 (索引のスキーマ名が既存スキーマを指す場合) のいずれか。

このタスクについて

表が新しい索引を取得する状況として、以下のような場合があります。

- 索引を持たないが後で索引を取得する表のニックネームを作成する。
- 索引を持つが後で別の索引を取得する表のニックネームを作成する。

こうした状況では、表の索引仕様を作成する必要があります。こうすることによって、SQL コンパイラーは、その表を参照する照会を処理するときに索引仕様の情報を使用することができます。

制約事項

ニックネームの索引を作成する際に、いくつかの制約事項があります。

- BIND オプション DYNAMICRULES BIND を適用すると、ステートメントを動的に準備できなくなります。また、CREATE INDEX ステートメントに INCLUDE、CLUSTER、PCTFREE、 MINPCTUSED、DISALLOW REVERSE SCANS、ALLOW REVERSE SCANS パラメーターを使用できません。
- UNIQUE の指定は、索引キーのデータにデータ・ソース表の各行のユニーク値が含まれる場合のみ行ってください。ユニーク性はチェックされません。
- 指定した列の保管長さの合計が 1024 を超えることはできません。
- LOB 列、あるいは LOB を基にした特殊タイプ列を索引の一部として使用することはできません。列の長さ属性が小さく 1024 バイトの制限範囲に入る場合でも、この制約が適用されます。

手順

以下の例は、索引を取得する表に対応したニックネームの索引仕様を作成する方法を示したものです。

例: 索引を持たないが後で索引を取得する表

索引を持たない CURRENT_EMP というデータ・ソース表のニックネーム *EMPLOYEE* を作成したとします。このニックネームが作成された後で CURRENT_EMP の索引が定義されており、そこでは索引キーとして WORKDEPT 列と JOB 列が使用されています。

この索引を表す索引仕様を作成するには、次の構文を使用します。

```
CREATE UNIQUE INDEX JOB_BY_DEPT ON EMPLOYEE
(WORKDEPT, JOB) SPECIFICATION ONLY
```

ここで、*JOB_BY_DEPT* は索引名です。

例: 新しい索引を取得する表

JAPAN_SALES という表のニックネーム *JP_SALES* を作成したとします。表には、ニックネームが作成されたときに持っていた索引に加えて、新しい索引が後で追加されています。新しい索引は、索引キーとして MARKUP 列を使用しています。

この索引を表す索引仕様を作成するには、次の構文を使用します。

```
CREATE UNIQUE INDEX JP_MARKUP ON JP_SALES (MARKUP) SPECIFICATION ONLY
```

ここで、*JP_MARKUP* は索引名です。

ビューの索引仕様の作成

ビューのニックネームを作成しても、フェデレーテッド・サーバーはそのビューの基になる表 (およびその索引) を認識していません。ビューの索引仕様を作成することによって、SQL コンパイラーが、そのビューを参照する照会を処理するときに、索引仕様の情報を使用できるようにする必要があります。

始める前に

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- SYSADM または DBADM 権限
- オブジェクトに対する CONTROL 特権またはオブジェクトに対する INDEX 特権のいずれか。および、データベースに対する IMPLICIT_SCHEMA 権限 (索引の暗黙的または明示的スキーマ名が存在しない場合)、またはスキーマに対する CREATEIN 特権 (索引のスキーマ名が既存スキーマを指す場合) のいずれか。

このタスクについて

制約事項

ニックネームの索引を作成する際に、いくつかの制約事項があります。

- BIND オプション DYNAMICRULES BIND を適用すると、ステートメントを動的に準備できなくなります。また、CREATE INDEX ステートメントに INCLUDE、CLUSTER、PCTFREE、 MINPCTUSED、DISALLOW REVERSE SCANS、ALLOW REVERSE SCANS パラメーターを使用できません。
- UNIQUE の指定は、索引キーのデータにデータ・ソース表の各行のユニーク値が含まれる場合のみ行ってください。ユニーク性はチェックされません。
- 指定した列の保管長さの合計が 1024 を超えることはできません。
- LOB 列、あるいは LOB を基にした特殊タイプ列を索引の一部として使用することはできません。列の長さ属性が小さく 1024 バイトの制限範囲に入る場合でも、この制約が適用されます。

手順

- 表索引の基になっている列 (複数の場合あり) をビューに含めておいてください。
- 基礎表のすべての索引の索引仕様を作成する場合は、それぞれの索引仕様を別々に作成する必要があります。

例

例: REGION 索引を表す索引仕様を作成する

JAPAN_SALES2007 というビューにニックネーム *JP_SALES2007* を作成したとします。このビューの基礎表は JAPAN_SALES 表で、この表には索引として REGION、AMOUNT、SALES_REP が含まれます。作成する CREATE INDEX ステートメントは、ビューのニックネームを参照し、ビューの基礎表の索引に関する情報を収納します。

```
CREATE UNIQUE INDEX JP_2007_REGION ON JP_SALES2007
(REGION) SPECIFICATION ONLY
```

ここで、*JP_2007_REGION* は索引名、*JP_SALES2007* はビュー JAPAN_SALES2007 のニックネームです。

Informix シノニムの索引仕様の作成

このトピックでは、表またはビューを基にした Informix シノニムに対してフェデレーテッド・サーバーがとるアクションについて説明します。

始める前に

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- SYSADM または DBADM 権限
- オブジェクトに対する CONTROL 特権またはオブジェクトに対する INDEX 特権のいずれか。および、データベースに対する IMPLICIT_SCHEMA 権限 (索引の暗黙的または明示的スキーマ名が存在しない場合)、またはスキーマに対する CREATEIN 特権 (索引のスキーマ名が既存スキーマを指す場合) のいずれか。

このタスクについて

Informix では、表またはビューのシノニムを作成することができます。フェデレーテッド・サーバーで Informix シノニムのニックネームを作成することが可能ですが、フェデレーテッド・サーバーがとるアクションは、シノニムが表とビューのどちらを基にしたものかによって異なります。

- シノニムのニックネームが作成されていて、このシノニムは Informix 表を基にしているとします。シノニムが表す表が索引を持っているとフェデレーテッド・サーバーが判別した場合は、シノニムの索引仕様が作成されます。シノニムが表す表が索引を持っていない場合は、シノニムの索引仕様は作成されません。ただし、CREATE INDEX ステートメントを使用して、手動で索引仕様を作成できます。
- シノニムのニックネームが作成されていて、このシノニムは Informix ビューを基にしているとします。フェデレーテッド・サーバーは、ビューがどの基礎表 (複数の場合あり) に基づいているか判別できません。したがって、シノニムの索引仕様は作成されません。ただし、CREATE INDEX ステートメントを使用して、手動で索引仕様を作成できます。

制約事項

ニックネームの索引を作成する際に、いくつかの制約事項があります。

- BIND オプション DYNAMICRULES BIND を適用すると、ステートメントを動的に準備できなくなります。また、CREATE INDEX ステートメントに INCLUDE、CLUSTER、PCTFREE、 MINPCTUSED、DISALLOW REVERSE SCANS、ALLOW REVERSE SCANS パラメーターを使用できません。
- UNIQUE の指定は、索引キーのデータにデータ・ソース表の各行のユニーク値が含まれる場合のみ行ってください。ユニーク性はチェックされません。
- 指定した列の保管長さの合計が 1024 を超えることはできません。
- LOB 列、あるいは LOB を基にした特殊タイプ列を索引の一部として使用することはできません。列の長さ属性が小さく 1024 バイトの制限範囲に入る場合でも、この制約が適用されます。

手順

以下の例は、Informix シノニムに対応したニックネームの索引仕様を作成する方法を示したものです。

例: 表を基にした Informix シノニムのニックネームが作成されている

シノニムの基になっているのが、索引を含まない Informix 表だった場合でも、ユーザーがシノニムの索引仕様を作成できるので、オプティマイザーは、データを素早く検出するにはどの列を検索すればよいかが分かります。作成するステートメントでシノニムのニックネームを指定することにより、シノニムの基になっている表の列に関する情報を提供できます。

この例では、SALES_CONTRACTS というシノニムのニックネーム *CONTRACTS* を作成したとします。このシノニムの基になっている表の名前は SALES2006_TABLE で、この表には索引として REGION、AMOUNT、SALES_REP が含まれています。作成する CREATE INDEX ステートメントは、シノニムのニックネームを参照し、シノニムの基礎表の索引に関する情報を収納します。

REGION 索引を表す索引仕様を作成するには、次の構文を使用します。

```
CREATE UNIQUE INDEX NORTHWEST_2006_REGION ON CONTRACTS (REGION) SPECIFICATION ONLY
```

ここで、*NORTHWEST_2006_REGION* は索引名、*CONTRACTS* はシノニム SALES_CONTRACTS のニックネームです。

例: ビューを基にした Informix シノニムのニックネームが作成されている

JAPAN_SALES2007 というビューを基にしたシノニムのニックネーム *JP_SALES2007* を作成したとします。このビューの基礎表は JAPAN_SALES 表で、この表には索引として REGION、AMOUNT、SALES_REP が含まれます。作成する CREATE INDEX ステートメントは、シノニムのニックネームを参照し、ビューの基礎表の索引に関する情報を収納します。

ビューを基にしたシノニムの索引仕様を作成するときは、表索引の基になっている列 (複数の場合あり) がビューに含まれていることを確認してください。基礎表のすべての索引の索引仕様を作成する場合は、それぞれの索引仕様を別々に作成する必要があります。

REGION 索引を表す索引仕様を作成するには、次の構文を使用します。

```
CREATE UNIQUE INDEX JP_2007_REGION ON JP_SALES2007 (REGION) SPECIFICATION ONLY
```

ここで、*JP_2007_REGION* は索引名、*JP_SALES2007* はビュー JAPAN_SALES2007 のニックネームです。

第 6 章 フェデレーテッド・プロシージャーの開発

フェデレーテッド・プロシージャーを使用すれば、データ・ソースにあるプロシージャー (つまり、リモート・プロシージャー) をローカル・プロシージャーのように呼び出すことができます。

フェデレーテッド・プロシージャー

フェデレーテッド・プロシージャー とは、データ・ソース上のプロシージャーを参照するフェデレーテッド・データベース・オブジェクトです。

別名が代替名であるのとは異なり、フェデレーテッド・プロシージャーはデータ・ソース・プロシージャーの代替名ではありません。フェデレーテッド・プロシージャーは、フェデレーテッド・データベースで定義するものですが、フェデレーテッド・プロシージャーの呼び出し時には、データ・ソース・プロシージャーが呼び出されます。フェデレーテッド・プロシージャーはフェデレーテッド・データベース・オブジェクトなので、ユーザーおよびクライアント・アプリケーションは、フェデレーテッド・プロシージャーを呼び出すことによって、データ・ソース・プロシージャーのロジックを呼び出すことができます。データ・ソース・プロシージャーの結果 (出力パラメーターなど) は、フェデレーテッド・プロシージャーによって戻されます。フェデレーテッド・プロシージャーを使用すれば、ユーザーおよびクライアント・アプリケーションは、データ・ソース・プロシージャーのロケーションを意識しなくてすみます。データ・ソース・プロシージャーを呼び出すには、フェデレーテッド・プロシージャーの名前を使用します。

フェデレーテッド・プロシージャーとリモート・プロシージャーの関係は、ニックネームとリモート表の関係と同じです。ニックネームとフェデレーテッド・プロシージャーは、フェデレーテッド・データベース上のオブジェクトです。ニックネームは、データ・ソース上のオブジェクト (表やビューなど) を参照するオブジェクトです。ニックネームを使用して、データ・ソース・オブジェクトを照会します。フェデレーテッド・プロシージャーを使用して、データ・ソース・プロシージャーを呼び出します。

フェデレーテッド・プロシージャーを登録するには、CREATE PROCEDURE (ソース派生) ステートメントを使用し、プロシージャーを呼び出すには、CALL ステートメントを使用します。CREATE PROCEDURE (ソース派生) ステートメントは、アプリケーション・プログラムに組み込むこともできれば、動的 SQL ステートメントによって実行することもできます。

パラメーターおよびデータ・タイプ

フェデレーションは、 3 つのタイプのパラメーター (IN、OUT、INOUT) を使用します。フェデレーテッド・プロシージャーを作成すると、デフォルトの順方向データ・タイプ・マッピングが使用されて、データ・ソース・プロシージャーのパラメーターのデータ・タイプがフェデレーテッド・データ・タイプにマッピングされます。このマッピングには、入力パラメーターや出力パラメーターのデータ・タイプと、結果セットの各列のデータ・タイプが含まれます。データ・ソース・パラメー

ターのデフォルトのタイプ・マッピングをオーバーライドする場合は、CREATE TYPE MAPPING ステートメントを使用します。ただし、結果セットのタイプ・マッピングは、ユーザー定義タイプ・マッピングの影響を受けません。ニックネーム列でサポートされているデータ・タイプのうち、LOB データ・タイプ以外のすべてのデータ・タイプを使用できます。フェデレーテッド・プロシージャーは、複合データ・タイプをサポートしていません。

ユーザー・マッピングおよび許可

フェデレーテッド・プロシージャーを呼び出すには、フェデレーテッド・プロシージャーとデータ・ソース・プロシージャーの両方に関して正しい許可が必要です。フェデレーテッド・プロシージャーを呼び出すと、そのフェデレーテッド・プロシージャーを作成したユーザー・マッピングと許可 ID の特権を使用して、データ・ソース表がアクセスされます。

例えば、ユーザー ZELLER が FP1 というフェデレーテッド・プロシージャーを作成します。FP1 プロシージャーは、Sybase 表にアクセスする Sybase プロシージャーを参照します。ZELLER のユーザー・マッピングにあるリモート・ユーザー ID には、Sybase 表を更新するための特権があります。ユーザー ZELLER は、FP1 プロシージャーの EXECUTE 特権をユーザー BHATIA に付与します。ユーザー BHATIA には、FP1 プロシージャーの参照先になっている Sybase プロシージャーの EXECUTE 特権を持つリモート・ユーザー ID に対する有効なユーザー・マッピングが必要です。ユーザー BHATIA がマッピングされているそのリモート・ユーザー ID は、その Sybase プロシージャーに関する SELECT 特権を必要としません。ユーザー BHATIA は、FP1 プロシージャーを呼び出すことによって、Sybase 表を更新できます。

プロシージャー呼び出しとアクセス・レベル

プロシージャー呼び出しとアクセス・レベルには、以下の問題があります。

- フェデレーテッド・プロシージャーでは、PRECOMPILE コマンドの実行時にデフォルトで CALL RESOLUTION IMMEDIATE 文節が使用されます。CALL RESOLUTION DEFERRED 文節は、フェデレーテッド・プロシージャーではサポートされていません。
- フェデレーテッド・プロシージャーがバッファーまたは標準出力に結果を出力するデータ・ソース・プロシージャーを呼び出す場合は、出力を表示できません。
- 外部のユーザー定義関数からフェデレーテッド・プロシージャーを呼び出す場合は、フェデレーテッド・プロシージャーのアクセス・レベルを READS SQL DATA または MODIFIES SQL DATA に設定しないでください。フェデレーテッド・アクセスは、外部関数の内部でブロックされます。
- パススルー・モードでのローカル・プロシージャーの呼び出しに当てはまる制限がフェデレーテッド・プロシージャーにも当てはまります。
- CALL ステートメントの各引数と、プロシージャー内の対応するパラメーターとの間には互換性がなければなりません。フェデレーテッド・プロシージャーには、ローカル・プロシージャーと同じパラメーター割り当て規則が適用されます。

トランザクション

トランザクションでフェデレーテッド・プロシージャーを使用する場合は、以下の問題について検討してください。

- フェデレーテッド・プロシージャーの参照先のデータ・ソース・プロシージャーでは、COMMIT ステートメントまたは ROLLBACK ステートメントを実行しないでください。フェデレーションはこの制限を強制しませんが、データ・ソース・プロシージャーで COMMIT ステートメントまたは ROLLBACK ステートメントを実行すると、データの不整合が発生する可能性があります。
- MODIFIES SQL DATA アクセス・レベルのフェデレーテッド・プロシージャーは、トリガー、動的コンパウンド・ステートメント、SQL スカラー、表、行関数、メソッドの内部から呼び出せません。SAVEPOINT ステートメントの実行後に、MODIFIES SQL DATA アクセス・レベルのフェデレーテッド・プロシージャーを呼び出すことはできません。
- フェデレーテッド・プロシージャーは、WITH HOLD カーソル、両方向スクロール・カーソル、更新可能カーソルをサポートしていません。データ・ソース・プロシージャーがこれらのタイプのカーソルを使用している場合でも、警告メッセージやエラー・メッセージを受け取ることはありません。これらのタイプのカーソルを使用するデータ・ソース・プロシージャーで結果セットにアクセスするアプリケーションは、それとは異なる動作をする場合があります。通常、カーソルは保持機能なしで戻され、順方向読み取り専用カーソルになります。

カタログ・ビュー

以下のシステム・カタログ・ビューは、フェデレーテッド・プロシージャーについての情報を保管します。

- SYSCAT.ROUTINES
- SYSCAT.ROUTINESFEDERATED
- SYSCAT.ROUTINEOPTIONS
- SYSCAT.ROUTINEPARMS
- SYSCAT.ROUTINEPARMOPTIONS

フェデレーテッド・プロシージャーの作成

フェデレーテッド・プロシージャーは、フェデレーテッド・サーバーから呼び出す各データ・ソース・プロシージャーごとに作成する必要があります。

始める前に

- データ・ソース・プロシージャーが既に存在している必要があります。
- データ・ソースのラッパーが登録され、サーバー定義が作成されています。
- 必要なユーザー・マッピングが作成されています。
- ステートメントの許可 ID によって保持される特権には、少なくとも以下のいずれかが含まれます。
  - データベースに対する IMPLICIT_SCHEMA 特権 (プロシージャーのスキーマ名が既存のスキーマを参照していない場合)

– スキーマに対する CREATEIN 特権 (プロシージャーのスキーマ名が既存のスキーマを参照している場合)
– SYSADM または DBADM 権限

- 許可 ID によってデータ・ソースで保持される特権には、リモート・カタログ表からプロシージャーの記述を選択する特権も含まれている必要があります。

このタスクについて

フェデレーテッド・プロシージャーを作成するには、CREATE PROCEDURE (ソース派生) SQL ステートメントを使用します。

手順

特定のデータ・ソースに必要なオプションに応じて CREATE PROCEDURE ステートメントを指定します。

データ・ソースおよびフェデレーテッド・プロシージャー

フェデレーテッド・プロシージャーを作成する前に、フェデレーションがデータ・ソース・プロシージャーをどのようにサポートしているかを検討します。

DB2 データ・ソースのフェデレーテッド・プロシージャー

フェデレーテッド・プロシージャーを作成する前に、CREATE PROCEDURE ステートメントでオプションを指定する方法について検討します。

フェデレーテッド・プロシージャーを作成するときには、以下の問題を念頭に置いてください。

- DB2 プロシージャーは、IN、OUT、INOUT パラメーターをサポートします。
- CREATE PROCEDURE ステートメントの SQL アクセス、決定論、および外部アクション文節について DB2 プロシージャーが指定するのと同じオプションを指定する必要があります。
- 2 つのリモート DB2 プロシージャーの名前が同じである場合、NUMBER OF PARAMETERS オプションを使用して、使用するプロシージャーを識別します。
- フェデレーテッド・データベースは、DB2 Universal Database™ for z/OS 内のプロシージャーのために作成され、COMMIT ON RETURN YES 文節を含むフェデレーテッド・プロシージャーに対して、COMMIT ステートメントを発行しません。

例

この例では、CREATE PROCEDURE ステートメントを使用して、DB2 のデータ・ソース・プロシージャーのためのフェデレーテッド・プロシージャーを作成する方法を示しています。DB2 コマンド行から CREATE PROCEDURE ステートメントを発行するか、Data Studio でフェデレーテッド・プロシージャーを作成することができます。

```
CREATE PROCEDURE PROC1 SOURCE KELLER.PROC1_DB2
     NUMBER OF PARAMETERS 3 FOR SERVER DB2_SERVER
     SPECIFIC MYPROC1 WITH RETURN TO CLIENT ALL
     MODIFIES SQL DATA DETERMINISTIC EXTERNAL ACTION;
```

PROC1
必須です。フェデレーテッド・プロシージャーの名前を指定します。

SOURCE KELLER.PROC1_DB2
必須です。DB2 プロシージャーのスキーマと名前を指定します。DB2 プロシージャーの場合は、CREATE PROCEDURE ステートメントで 2 部構成の名前を指定します。その 2 部構成の名前の形式は、*source_schema_name.source_procedure_name*です。

NUMBER OF PARAMETERS 3
DB2 プロシージャーで使用している IN、OUT、INOUT パラメーターの総数を指定します。このパラメーターは、同じスキーマ名とプロシージャー名を使用しているプロシージャーが複数存在する場合に使用します。例えば、KELLER というスキーマに、3 つのパラメーターを使用する PROC1 プロシージャーと 1 つのパラメーターを使用する PROC1 プロシージャーがあるとすれば、その両方のプロシージャーの名前はいずれも KELLER.PROC1 になります。そのような場合に、データ・ソース・プロシージャーの NUMBER OF PARAMETERS の値によって、CREATE PROCEDURE ステートメントでどちらのプロシージャーを参照しているのかを明示できます。

FOR SERVER DB2_SERVER
必須です。フェデレーテッド・プロシージャーを作成するサーバー定義を指定します。

SPECIFIC MYPROC1
作成するフェデレーテッド・プロシージャーの固有の名前を指定します。このパラメーターはフェデレーテッド・プロシージャー専用であり、データ・ソース・プロシージャーには関連付けられません。固有の名前を指定しない場合は、フェデレーテッド・データベース・マネージャーによって名前が生成されます。

WITH RETURN TO CLIENT ALL
結果セットをクライアント・アプリケーションに戻すことを指定します。フェデレーションは、最大で 1 つの結果セットを戻します。このパラメーターを指定しない場合は、デフォルトで WITH RETURN TO CALLER ALL が使用されます。

MODIFIES SQL DATA
フェデレーテッド・プロシージャーに組み込まれている SQL ステートメントのデータ・アクセスのレベルを指定します。指定した文節が DB2 プロシージャーと合致していない場合は、エラー・メッセージが戻されます。この文節を指定しない場合は、DB2 プロシージャーの文節が使用されます。

DETERMINISTIC
フェデレーテッド・プロシージャーが特定の引数値セットについて常に同じ結果を戻すかどうかを指定します。このパラメーターによって、フェデレーテッド・サーバーとデータ・ソースの間の対話のパフォーマンスを改善できます。指定した文節が DB2 プロシージャーと合致していない場合は、エラー・メッセージが戻されます。この文節を指定しない場合は、DB2 プロシージャーの文節が使用されます。

EXTERNAL ACTION
データベース・マネージャーによって管理されていないオブジェクトの状態

を変更する操作をフェデレーテッド・プロシージャーによって実行するかどうかを指定します。指定した文節が DB2 プロシージャーと合致していない場合は、エラー・メッセージが戻されます。この文節を指定しない場合は、DB2 プロシージャーの文節が使用されます。

フェデレーテッド・プロシージャーと Microsoft SQL Server

フェデレーテッド・プロシージャーを作成する前に、どのパラメーターがサポートされているのか、どのように結果セットが戻されるか、およびどのような制限が存在するのかを検討します。

パラメーター

Microsoft SQL Server プロシージャーは、オプションの INPUT および OUTPUT パラメーターの使用をサポートしています。 SQL Server ラッパーは、各 INPUT パラメーターをフェデレーテッドの IN パラメーターに、各 OUTPUT パラメーターをフェデレーテッドの INOUT パラメーターにそれぞれマッピングします。SQL Server プロシージャーでオプション・パラメーターを使用する場合は、CREATE PROCEDURE ステートメントの NUMBER OF PARAMETERS 文節を指定するときに、それらのパラメーターをカウントする必要があります。

結果セット

SQL Server ラッパーは、結果セットと OUTPUT パラメーターを戻すことができます。SQL Server プロシージャーが OUTPUT パラメーターと結果セットの両方を戻す場合は、パラメーターだけが戻されます。結果セットは廃棄され、SQL0464W エラー・メッセージを受け取ることになります。結果セットを戻すプロシージャーの DB2_RETURN_STATUS 値を取り出すことはできますが、プロシージャーからの実際の戻り値が何であっても、値 0 (ゼロ) が常に戻されます。

制限

以下の場合、SQL Server ラッパーはプロシージャーを呼び出せません。

- 別のプロシージャーがすでに呼び出されている場合
- 1 つの接続で別のステートメントが実行されている場合

これらの制限を回避するには、別のサーバーでフェデレーテッド・プロシージャーを定義できます。この例では、ニックネーム *sql_nick* とプロシージャー *sql_proc* が別々のサーバーで定義されている場合、以下のステートメントは正常に実行されます。ニックネームとプロシージャーが同じサーバーで定義されていれば、ステートメントは失敗します。

```
DECLARE clientcur CURSOR FOR SELECT colsml,coldec,colvch,coltsp
FROM sql_nick OPEN clientcur; CALL sql_proc();
```

例

この例では、CREATE PROCEDURE ステートメントを使用して、Microsoft SQL Server のデータ・ソース・プロシージャーのためのフェデレーテッド・プロシージャーを作成する方法を示しています。Microsoft SQL Server は UNIQUE ID 文節をサポートしていませんし、SOURCE 文節での source_package_name の値もサポートしていないことに注意してください。

```
CREATE PROCEDURE PROC1 SOURCE BHATIA.PROC1_MSSQL
     NUMBER OF PARAMETERS 5 FOR SERVER MSSQL_SERVER
     SPECIFIC MYPROC1 WITH RETURN TO CLIENT ALL
     MODIFIES SQL DATA DETERMINISTIC EXTERNAL ACTION;
```

PROC1
必須です。フェデレーテッド・プロシージャーの名前を指定します。

SOURCE BHATIA.PROC1_MSSQL
必須です。データ・ソース上のスキーマとプロシージャーの名前を指定します。

FOR SERVER MSSQL_SERVER
必須です。フェデレーテッド・プロシージャーを作成するサーバー定義を指定します。

NUMBER OF PARAMETERS 5
データ・ソース・プロシージャーで使用している IN および OUTPUT パラメーターの総数を指定します。

SOURCE BHATIA.PROC1_SQL
データ・ソース・プロシージャーのスキーマと名前を指定します。その 2 部構成の名前の形式は、*source_schema_name.source_procedure_name*です。

SPECIFIC MYPROC1
作成するフェデレーテッド・プロシージャーの固有の名前を指定します。このパラメーターはフェデレーテッド・プロシージャー専用であり、データ・ソース・プロシージャーには関連付けられません。固有の名前を指定しない場合は、フェデレーテッド・データベース・マネージャーによって名前が生成されます。

WITH RETURN TO CLIENT ALL
結果セットをクライアント・アプリケーションに戻すことを指定します。フェデレーションは、最大で 1 つの結果セットを戻します。このパラメーターを指定しない場合は、デフォルトで WITH RETURN TO CALLER ALL が使用されます。

MODIFIES SQL DATA
フェデレーテッド・プロシージャーに組み込まれている SQL ステートメントのデータ・アクセスのレベルを指定します。指定した文節がデータ・ソース・プロシージャーと合致していない場合は、エラー・メッセージが戻されます。この文節を指定しない場合は、データ・ソース・プロシージャーの文節が使用されます。

DETERMINISTIC
フェデレーテッド・プロシージャーが特定の引数値セットについて常に同じ結果を戻すかどうかを指定します。このパラメーターによって、フェデレーテッド・サーバーとデータ・ソースの間の対話のパフォーマンスを改善できます。

EXTERNAL ACTION
データベース・マネージャーによって管理されていないオブジェクトの状態を変更する操作をフェデレーテッド・プロシージャーによって実行するかどうかを指定します。

フェデレーテッド・プロシージャーと Oracle

同じ Oracle 関数に関して、フェデレーテッド・プロシージャーと関数マッピングを作成できます。

SQL の中で Oracle 関数をスカラー関数として使用する必要がある場合は、関数マッピングを使用してください。CALL ステートメントの中で Oracle 関数を使用する必要がある場合は、フェデレーテッド・プロシージャーを使用してください。

Oracle では、関数に対して 1 つの値だけが戻されます。フェデレーテッド・プロシージャーの戻り値は、パラメーター・リストの先頭に追加の OUT パラメーターとして表示されます。パラメーターの名前は常に DEFAULT になります。CREATE PROCEDURE (ソース派生) ステートメントで NUMBER OF PARAMETERS 文節を指定するときに、戻り値はカウントしないでください。

Oracle のデータ・タイプによっては、プロシージャーのパラメーターを宣言するときに、精度、長さ、位取りに関する情報が Oracle カタログに格納されないことがあります。フェデレーテッド・プロシージャーの作成時には、Oracle プロシージャーに関する情報が Oracle カタログから収集されます。精度、長さ、位取りに関する情報が Oracle カタログに格納されていない場合、フェデレーテッド・プロシージャーは以下のように動作します。

- パラメーターのデータ・タイプの最大長を使用します。
- Oracle の NUMBER データ・タイプをフェデレーテッドの DOUBLE データ・タイプにマッピングします。このマッピングを変更するには、Oracle の NUMBER に関するデフォルトの順方向データ・タイプ・マッピングをオーバーライドします。

ヒント: デフォルトの順方向データ・タイプ・マッピングをオーバーライドすると、他のフェデレーテッド DDL 操作 (CREATE NICKNAME など) にも影響があります。そのため、プロシージャーを作成する前にタイプ・マッピングを変更してください。新しいタイプ・マッピングで Oracle プロシージャーのためのプロシージャーを作成してから、新しいタイプ・マッピングをドロップします。その後に作成するニックネームやプロシージャーは、デフォルトのタイプ・マッピングを使用することになります。

例

この例では、CREATE PROCEDURE ステートメントを使用して、Oracle のデータ・ソース・プロシージャーのためのフェデレーテッド・プロシージャーを作成する方法を示しています。DB2 コマンド行から CREATE PROCEDURE ステートメントを発行するか、Data Studio でフェデレーテッド・プロシージャーを作成することができます。

```
CREATE PROCEDURE PROC2 SOURCE ZELLER_SCHEMA.ORACLE_PKG9.PROC2
     NUMBER OF PARAMETERS 5 UNIQUE_ID '2' FOR SERVER ORA_SERVER
     SPECIFIC MYPROC1 WITH RETURN TO CLIENT ALL
     MODIFIES SQL DATA DETERMINISTIC NO EXTERNAL ACTION;
```

PROC2
必須です。フェデレーテッド・プロシージャーの名前を指定します。

SOURCE ZELLER_SCHEMA.ORACLE_PKG9.PROC2
必須です。Oracle プロシージャー、または関数のスキーマ、パッケージ、名

前を指定します。 Oracle プロシージャー、または関数がパッケージに含まれている場合は、CREATE PROCEDURE ステートメントで 3 部構成の名前を指定する必要があります。その 3 部構成の名前の形式は、*source_schema_name.source_package_name.source_procedure_name*です。Oracle プロシージャー、または関数がパッケージに含まれていない場合は、CREATE PROCEDURE ステートメントで 2 部構成の名前を指定する必要があります。その 2 部構成の名前の形式は、*source_schema_name.source_procedure_name*です。

NUMBER OF PARAMETERS 5
Oracle プロシージャーで使用している IN、OUT、INOUT パラメーターの総数を指定します。このパラメーターは、同じスキーマ名とプロシージャー名を使用しているプロシージャーが複数存在する場合に使用します。例えば、ZELLER というスキーマに、2 つのパラメーターを使用する PROC1 プロシージャーと 3 つのパラメーターを使用する PROC1 プロシージャーがあるとすれば、その両方のプロシージャーの名前はいずれも ZELLER.PROC1 になります。そのような場合に、データ・ソース・プロシージャーの NUMBER OF PARAMETERS の値によって、CREATE PROCEDURE ステートメントでどちらのプロシージャーを参照しているのかを明示できます。Oracle の REFCURSOR パラメーターも NUMBER OF PARAMETERS のカウントに含める必要があります。

UNIQUE_ID '2'
Oracle プロシージャーの固有 ID を指定します。 UNIQUE_ID パラメーターは、スキーマ名とプロシージャー名とパラメーター数によって Oracle プロシージャーを一意的に識別できない場合にのみ使用します。UNIQUE ID は、Oracle システム・カタログの ALL_ARGUMENTS.OVERLOAD 列の値です。 UNIQUE ID パラメーターを指定しない場合は、フェデレーテッド・サーバーが多重定義プロシージャーを検出して、エラーを戻します。このオプションは、Oracle プロシージャーでのみ使用してください。

FOR SERVER ORA_SERVER
必須です。フェデレーテッド・プロシージャーを作成するサーバー定義を指定します。

SPECIFIC MYPROC1
作成するフェデレーテッド・プロシージャーの固有の名前を指定します。このパラメーターはフェデレーテッド・プロシージャー専用であり、データ・ソース・プロシージャーには関連付けられません。固有の名前を指定しない場合は、フェデレーテッド・データベース・マネージャーによって名前が生成されます。このパラメーターはオプションです。

WITH RETURN TO CLIENT ALL
結果セットをクライアント・アプリケーションに戻すことを指定します。フェデレーションは、最大で 1 つの結果セットを戻します。このパラメーターを指定しない場合は、デフォルトで WITH RETURN TO CALLER ALL が使用されます。

MODIFIES SQL DATA
フェデレーテッド・プロシージャーに組み込まれている SQL ステートメントのデータ・アクセスのレベルを指定します。指定した文節が Oracle プロ

シージャーと合致していない場合は、エラー・メッセージが戻されます。この文節を指定しない場合は、Oracle プロシージャーの文節が使用されます。

DETERMINISTIC
フェデレーテッド・プロシージャーが特定の引数値セットについて常に同じ結果を戻すかどうかを指定します。このパラメーターによって、フェデレーテッド・サーバーとデータ・ソースの間の対話のパフォーマンスを改善できます。

NO EXTERNAL ACTION
データベース・マネージャーによって管理されていないオブジェクトの状態を変更する操作をフェデレーテッド・プロシージャーによって実行するかどうかを指定します。

フェデレーテッド・システムの多重定義プロシージャー

同じ名前と同じスキーマを持ったプロシージャーのことを*多重定義* プロシージャーといいます。多重定義プロシージャー同士で、パラメーターの数を変えたり、パラメーター・シグニチャーを変えたりすることができます。 プロシージャーを多重定義する目的は、1 つのプロシージャーに関して似たようなバージョンをいくつか作成することです。

フェデレーテッド・サーバーで多重定義プロシージャーを作成できるのは、各プロシージャーのパラメーターの数が異なっている場合に限られます。

Oracle では、各プロシージャーのパラメーターの数が異なっている場合、またはパラメーターのタイプが異なっている場合に、多重定義プロシージャーを作成できます。Oracle の多重定義プロシージャーについても、フェデレーテッド・プロシージャーを作成できます。

名前、スキーマ名、およびパラメーター数が同じであるプロシージャーを区別するには、フェデレーテッド・プロシージャーを作成するときに UNIQUE ID を指定する必要があります。

UNIQUE ID を確認するには、2 とおりの方法があります。コントロール・センターのディスカバリー機能を使用する方法と、Oracle カタログを照会する方法です。

コントロール・センターのディスカバー機能の使用

コントロール・センターでフェデレーテッド・プロシージャーを作成する場合は、ディスカバリー機能によって、Oracle プロシージャーが多重定義になっているかどうかを確認できます。各 Oracle プロシージャーについて、多重定義の値が UNIQUE ID フィールドに表示されます。

Oracle カタログの照会

UNIQUE ID を確認するために、Oracle の SYS.ALL_ARGUMENTS カタログの OVERLOAD 列を照会できます。 UNIQUE ID の値は、数値 (1 など) を含んだ文字リテラルです。フェデレーテッド・サーバーのパススルー・モードを使用することも、Oracle クライアントを直接照会することも可能です。

例えば、Oracle カタログを照会して、HJZ で始まるプロシージャーのシグニチャーと多重定義の列を表示するには、以下の SELECT ステートメントを使用します。

```
SELECT owner, package_name, object_name, overload, position, argument_name, in_out,
data_type
   FROM all_arguments aa
   WHERE object_name like 'HJZ%'
   ORDER BY owner, package_name, object_name,overload, position;
```

上記の照会の出力を以下に示します。その出力からわかるとおり、HJZ_PACK1 パッケージには HJZTEST1 という名前のプロシージャーが 3 つ含まれています。プロシージャーの数を確認するには、OBJECT_NAME 列と OVERLOAD 列の両方に注目します。最初のプロシージャーには、1 つの IN パラメーター (数値データ・タイプ) があります。 2 番目のプロシージャーには、1 つの IN パラメーター (文字データ・タイプ) があります。3 番目のプロシージャーには、1 つの OUT パラメーター (文字データ・タイプ) と 1 つの IN パラメーター (数値データ・タイプ) があります。この出力からは、HJZ_PACK1 パッケージに HJZTEST3 という名前のプロシージャーが 2 つあることもわかります。さらに、パッケージに組み込まれていない HJZTEST1 という名前のプロシージャーもあります。この最後のプロシージャーには、1 つの IN パラメーター (数値データ・タイプ) があります。

```
OWNER    PACKAGE_NAME OBJECT_NAME OVERLOAD POSITION ARGUMENT_NAME IN_OUT DATA_TYPE
-------- ------------ ----------- -------- -------- ------------- ------ ---------
J15USER1 HJZ_PACK1    HJZTEST1    1        1        A             IN     NUMBER
J15USER1 HJZ_PACK1    HJZTEST1    2        1        A             IN     CHAR
J15USER1 HJZ_PACK1    HJZTEST1    3        0        -             OUT    CHAR
J15USER1 HJZ_PACK1    HJZTEST1    3        1        A             IN     NUMBER
J15USER1 HJZ_PACK1    HJZTEST3    1        1        A             IN     NUMBER
J15USER1 HJZ_PACK1    HJZTEST3    1        2        B             OUT    NUMBER
J15USER1 HJZ_PACK1    HJZTEST3    2        1        A             IN     CHAR
J15USER1 HJZ_PACK1    HJZTEST3    2        2        B             OUT    CHAR
J15USER1 -            HJZTEST1    -        1        A             IN     NUMBER
9 record(s) selected.
```

CHAR データ・タイプの IN パラメーターがある 2 番目の多重定義プロシージャーのためのフェデレーテッド・プロシージャーを作成するには、以下の CREATE PROCEDURE ステートメントを実行します。

```
CREATE PROCEDURE HJZTEST1 SOURCE J15USER1.HJZ_PACK1.HJZTEST1
 NUMBER OF PARAMETERS 1 UNIQUE ID '2'
 FOR SERVER ORA_SERVER WITH RETURN TO CLIENT ALL;
```

重要: 上記の例の NUMBER OF PARAMETERS 文節では、プロシージャーを一意的に識別できていません。この表には、HJZTEST1 という名前のプロシージャーが 2 つあり、そのどちらもパラメーターの数は 1 つです。使用する多重定義プロシージャーを識別するには、UNIQUE ID 文節を指定する必要があります。UNIQUE_ID 文節の値としては、OVERLOAD 列の値を使用します。UNIQUE ID 文節を指定した場合、NUMBER OF PARAMETERS 文節を指定するかどうかは任意です。データ・ソース・プロシージャーのパラメーターの数が正しいかどうかを確認したいのであれば、NUMBER OF PARAMETERS 文節を使用できます。

フェデレーテッド・プロシージャーと Sybase

フェデレーテッド・プロシージャーを作成する前に、どのパラメーターがサポートされているのか、どのように結果セットが戻されるか、およびどのような制限が存在するのかを検討します。

パラメーター

Sybase プロシージャーは、INPUT パラメーターと OUTPUT パラメーターを使用します。Sybase ラッパーは、Sybase の INPUT パラメーターをフェデレーテッドの IN パラメーターに、Sybase の OUTPUT パラメーターをフェデレーテッドの INOUT パラメーターにそれぞれマッピングします。 Sybase プロシージャーでオプションのパラメーターを使用することはできますが、フェデレーテッド・プロシージャーでオプションのパラメーターを使用することはできません。したがって、CALL ステートメントを実行する際には、すべてのパラメーターを指定する必要があります。

Sybase バージョン 12.0 の場合は、すべてのパラメーターが入力パラメーターです。フェデレーテッド・プロシージャーで出力パラメーター値を戻すことはできません。これは Sybase カタログの制限であり、Sybase のそれ以降のバージョンには当てはまりません。

結果セット

出力パラメーターと結果セットを戻す Sybase プロシージャーの場合は、結果セットが廃棄されます。Sybase プロシージャーが結果セットと戻り値を戻す場合は、データ・ソース・プロシージャーからの実際の戻り値が何であっても、戻り値 0 が提供されます。どちらの場合も、警告メッセージを受け取ることはありません。

ただし、以下の両方の条件に当てはまる場合は、結果セットが廃棄され、SQL0464W のメッセージが戻されます。

- フェデレーテッド・プロシージャーが、結果セットを戻す Sybase プロシージャーのために定義されていること。
- フェデレーテッド・プロシージャーをトリガーまたはユーザー定義関数の中から呼び出すこと。

制限

以下の場合、Sybase ラッパーはプロシージャーを呼び出せません。

- 別のプロシージャーがすでに呼び出されている場合
- 1 つの接続で別のステートメントが実行されている場合

これらの制限を回避するには、別のサーバーでフェデレーテッド・プロシージャーを定義できます。この Sybase の例では、ニックネーム *syb_nick* とプロシージャー *syb_proc* が別々のサーバーで定義されている場合、以下のステートメントは正常に実行されます。ニックネームとプロシージャーが同じサーバーで定義されていれば、ステートメントは失敗します。

```
DECLARE clientcur CURSOR FOR SELECT colsml,coldec,colvch,coltsp
FROM syb_nick OPEN clientcur; CALL syb_proc();
```

例

この例では、CREATE PROCEDURE ステートメントを使用して、Sybase のデータ・ソース・プロシージャーのためのフェデレーテッド・プロシージャーを作成する方法を示しています。Sybase は CREATE PROCEDURE ステートメントの UNIQUE ID 文節をサポートしていないことに注意してください。

```
CREATE PROCEDURE PROC1 SOURCE BHATIA.PROC1_SYBASE
     NUMBER OF PARAMETERS 3 FOR SERVER SYBASE_SERVER
     SPECIFIC MYPROC1 WITH RETURN TO CLIENT ALL
     MODIFIES SQL DATA DETERMINISTIC EXTERNAL ACTION;
```

PROC1

必須です。フェデレーテッド・プロシージャーの名前を指定します。

SOURCE BHATIA.PROC1_SYBASE

必須です。Sybase プロシージャーのスキーマと名前を指定します。 Sybase プロシージャーの場合は、CREATE PROCEDURE ステートメントで 2 部構成の名前を指定します。その 2 部構成の名前の形式は、*source_schema_name.source_procedure_name*です。

NUMBER OF PARAMETERS 3

Sybase プロシージャーで使用している IN、OUT、INOUT パラメーターの総数を指定します。このパラメーターは、同じスキーマ名とプロシージャー名を使用しているプロシージャーが複数存在する場合に使用します。例えば、BHATIA というスキーマに、3 つのパラメーターを使用する PROC1 プロシージャーと 1 つのパラメーターを使用する PROC1 プロシージャーがあるとすれば、その両方のプロシージャーの名前はいずれも BHATIA.PROC1 になります。そのような場合に、データ・ソース・プロシージャーの NUMBER OF PARAMETERS の値によって、CREATE PROCEDURE ステートメントでどちらのプロシージャーを参照しているのかを明示できます。

FOR SERVER SYBASE_SERVER

必須です。フェデレーテッド・プロシージャーを作成するサーバー定義を指定します。

SPECIFIC MYPROC1

作成するフェデレーテッド・プロシージャーの固有の名前を指定します。このパラメーターはフェデレーテッド・プロシージャー専用であり、データ・ソース・プロシージャーには関連付けられません。固有の名前を指定しない場合は、フェデレーテッド・データベース・マネージャーによって名前が生成されます。

WITH RETURN TO CLIENT ALL

結果セットをクライアント・アプリケーションに戻すことを指定します。フェデレーションは、最大で 1 つの結果セットを戻します。このパラメーターを指定しない場合は、デフォルトで WITH RETURN TO CALLER ALL が使用されます。

MODIFIES SQL DATA

フェデレーテッド・プロシージャーに組み込まれている SQL ステートメントのデータ・アクセスのレベルを指定します。指定した文節が Sybase プロシージャーと合致していない場合は、エラー・メッセージが戻されます。この文節を指定しない場合は、Sybase プロシージャーの文節が使用されます。

DETERMINISTIC

フェデレーテッド・プロシージャーが特定の引数値セットについて常に同じ

結果を戻すかどうかを指定します。このパラメーターによって、フェデレーテッド・サーバーとデータ・ソースの間の対話のパフォーマンスを改善できます。

EXTERNAL ACTION
データベース・マネージャーによって管理されていないオブジェクトの状態を変更する操作をフェデレーテッド・プロシージャーによって実行するかどうかを指定します。

フェデレーテッド・プロシージャーを呼び出すための許可の付与または取り消し

フェデレーテッド・データベースの管理者は、フェデレーテッド・プロシージャーを呼び出すために必要な許可を他のユーザーに付与する必要があります。

始める前に

フェデレーテッド・プロシージャーを呼び出すユーザーは、フェデレーテッド・サーバーからデータ・ソースへの有効なユーザー・マッピングを持っている必要があります。リモート・ユーザー ID は、フェデレーテッド・サーバー上の EXECUTE 許可に相当する許可をデータ・ソース上で持っていなければなりません。ユーザーにはフェデレーテッド・プロシージャーの EXECUTE 許可を付与できます。しかし、データ・ソース上のユーザー許可がフェデレーテッド・サーバー上の EXECUTE 許可と同等でなければ、データ・ソース・プロシージャーに対する呼び出しが失敗します。

GRANT ステートメントの許可 ID には、少なくとも以下のいずれかの権限が必要です。

- フェデレーテッド・プロシージャーの EXECUTE に関する WITH GRANT OPTION
- SYSADM または DBADM 権限

手順

コマンド行から GRANT ステートメントを発行して許可特権を指定します。

例 1: BHATIA スキーマのすべてのプロシージャー (今後作成するプロシージャーも含む) の EXECUTE 特権を HR_DEPT グループのユーザーに付与するには、以下のステートメントを使用します。

```
GRANT EXECUTE ON PROCEDURE
 BHATIA.* TO HR_DEPT
```

例 2: PROC1 プロシージャーの EXECUTE 特権をユーザー ZELLER に付与し、そのプロシージャーの EXECUTE 特権を他のユーザーに付与する許可もユーザー ZELLER に与えるには、以下のステートメントを使用します。

```
GRANT EXECUTE ON PROCEDURE PROC1
 TO ZELLER
 WITH GRANT OPTION
```

例 3: MY_PROC2 という特定名で作成した PROC2 プロシージャーの EXECUTE 特権をユーザー ERFAN に付与するには、以下のステートメントを使用します。

```
GRANT EXECUTE ON SPECIFIC
 PROCEDURE MY_PROC2 TO ERFAN
```

パラメーター情報の表示およびフェデレーテッド・プロシージャーの呼び出し

パラメーター情報を表示するには、SYSCAT.ROUTINESFEDERATED カタログ・ビューを照会します。次に、CALL ステートメントを使用して、フェデレーテッド・プロシージャーを呼び出します。

始める前に

フェデレーテッド・プロシージャーを呼び出すユーザーは、データ・ソース上での EXECUTE 特権と、データ・ソース表にアクセスする特権を持つリモート・データ・ソースのユーザー ID への有効なユーザー・マッピングを持っている必要があります。

手順

パラメーター情報を表示し、フェデレーテッド・プロシージャーを呼び出すには、以下のいずれかの方法を使用します。

1. SELECT ステートメントを使用して、フェデレーテッド・データベース・カタログの SYSCAT.ROUTINEPARMS ビューを照会します。
2. CALL ステートメントを実行します。

この例では、SELECT ステートメントを使用して、フェデレーテッド・プロシージャーについての情報を取得します。例えば、フェデレーテッド・プロシージャー FEDPROC1 がフェデレーテッド・スキーマ BOB に存在する場合は、以下の SELECT ステートメントを実行します。

```
SELECT ordinal, char(parmname,30)
 AS name,
   rowtype, char(typename,30) AS type
   FROM syscat.routineparms
   WHERE routinename='FEDPROC1' AND
    routineschema = 'BOB'
   ORDER BY ordinal;
```

この照会の結果として、パラメーターのリストが戻されます。

```
ORDINAL NAME ROWTYPE TYPE
------- ---- ------- --------
1       P1   P       INTEGER
2       P2   O       VARCHAR
```

P という行タイプは、入力パラメーターを意味します。O という行タイプは、出力パラメーターを意味します。この例では、INOUT パラメーターを意味する B という行タイプは表示されていません。

フェデレーテッド・プロシージャーの変更またはドロップ

フェデレーテッド・プロシージャーの 1 つ以上のパラメーターのデータ・タイプを変更することにより、既存のフェデレーテッド・プロシージャーを変更することができます。

始める前に

DROP PROCEDURE ステートメントの許可 ID には、以下のいずれかの権限が必要です。

- SYSADM または DBADM 権限
- フェデレーテッド・プロシージャーのスキーマに対する DROPIN 特権
- フェデレーテッド・プロシージャーのカタログ・ビューの DEFINER 列に記録されているプロシージャーの定義者
- フェデレーテッド・プロシージャーに対する CONTROL 特権

このタスクについて

パラメーターのデータ・タイプの変更に加えて、フェデレーテッド・プロシージャーに他の変更を加えなければならない場合もあります。例えば、リモート・プロシージャーが変更された場合、フェデレーテッド・プロシージャーを変更しなければならない場合もあります。この場合、直接フェデレーテッド・プロシージャーを変更することはできません。最初にプロシージャーをドロップし、次に新しい設定でプロシージャーを再作成しなければなりません。そうしないと、CREATE OR REPLACE PROCEDURE ステートメントを使用して、既存のプロシージャーを置き換えなければならなくなります。

フェデレーテッド・プロシージャーをドロップすると、フェデレーテッド・データベースのシステム・カタログからそのプロシージャーが削除されますが、データ・ソース・プロシージャーは影響を受けません。フェデレーテッド・プロシージャーをドロップすると、そのドロップしたプロシージャーに依存するアプリケーションが好ましくない影響を受ける可能性があることを、念頭に置いてください。

コマンド行を使用して、フェデレーテッド・プロシージャーをドロップすることができます。

手順

DROP ステートメントを発行して、フェデレーテッド・プロシージャーをドロップします。
例:

```
DROP PROCEDURE federated_procedure_name
```

フェデレーテッド・プロシージャーの結果セットの結合

DB2FEDGENTF コマンドを使用して、フェデレーテッド・プロシージャーが返す結果セットを結合できます。

始める前に

- **DB2FEDGENTF** コマンドの **-create** パラメーターを使用するには、フェデレーテッド・データベースに対する DBADM 権限か、フェデレーテッド・データベースに対する以下の権限または特権の組み合わせのいずれかが必要です。
  1. 以下のいずれかの権限が必要です。
     - CREATE_EXTERNAL_ROUTINE
     - CREATETAB

2. 以下の権限または特権のいずれかも必要です。
   - *install_dir*/sqllib/function ディレクトリーに対する書き込み特権。*install_dir* はフェデレーテッド・サーバーがインストールされているディレクトリーです。
   - IMPLICIT_SCHEMA 権限 (関数の暗黙的または明示的スキーマ名が存在しない場合)
   - スキーマに対する CREATEIN 特権 (関数のスキーマ名が存在する場合)

- **DB2FEDGENTF** コマンドの **-drop** パラメーターを使用するには、フェデレーテッド・データベースに対する以下の権限または特権の少なくとも 1 つが必要です。
  - オブジェクトのスキーマに対する DROPIN 特権
  - オブジェクトの OWNER
  - DBADM 権限

このタスクについて

フェデレーテッド・プロシージャーによって返される結果セットを結合して、ローカルおよびリモートの表からの、特にフェデレーテッド・プロシージャーを使わないとアクセスできないシステムのデータを結合します。

手順

1. **DB2FEDGENTF** コマンドの **-create** パラメーターを使って、表関数を作成および登録します。例えば、以下のようにします。

```
DB2FEDGENTF –db sample –u user1 –p password1
  -create
    -stpn S1_INVENTORY
    -tfn S1_INVTRY_TF
    -c 'PART_NUM CHAR(10), S1_QTY INT'
```

2. 結合を使って、フェデレーテッド・プロシージャーの結果セットのデータを組み合わせます。 フェデレーテッド・プロシージャーの結果セットをローカル表と結合することも、2 つのフェデレーテッド・プロシージャーの結果セットを結合することもできます。

例

以下の例では、ProINVENTORY と ProINVENTORY2 という名前のストアード・プロシージャーがあり、2 つの供給業者の在庫をそれぞれ結果セットとして返します。また、メーカーの在庫が含まれている PARTS 表があります。この表の列名とタイプは以下のとおりです。

表 5. PARTS 表の列

列名	列タイプ
PART_NUM	CHAR(10)
PART_DESC	VARCHAR(400)
INVENTORY	INT

結果セットとローカル表の結合の例

この例では、1 つの供給業者とメーカーの在庫を組み合わせて、入手可能な合計在庫を判断します。

1. CREATE PROCEDURE ステートメントを使用して、ProINVENTORY ストアード・プロシージャーから S1_INVENTORY フェデレーテッド・プロシージャーを作成します。

```
CREATE PROCEDURE S1_INVENTORY
 SOURCE ProINVENTORY
 FOR SERVER ORA1;
```

2. **DB2FEDGENTF** コマンドを使用して、PART_NUM と S1_QTY 結果セットを含む S1_INVTRY_TF 表関数を作成します。

```
DB2FEDGENTF –db sample –u user1 –p password1
  -create
    -stpn S1_INVENTORY
    -tfn S1_INVTRY_TF
    -c 'PART_NUM CHAR(10), S1_QTY INT'
```

3. 結合を使って、PARTS 表のデータを S1_INVTRY_TF 表関数の結果セットと組み合わせます。

```
SELECT a.PART_NUM, a.PART_DESC, a.INVENTORY + b.S1_QTY as MAX_QTY
 FROM PARTS a, TABLE(S1_INVTRY_TF()) b
 WHERE a. PART_NUM = b. PART_NUM
```

2 つの結果セットの結合の例

この例では、メーカーは両方の供給業者の在庫を組み合わせて、入手可能な合計在庫を判断します。

1. 1 番目の供給業者に S1_INVTRY_TF 表関数を作成します。
   a. CREATE PROCEDURE ステートメントを使用して、ProINVENTORY ストアード・プロシージャーから S1_INVENTORY フェデレーテッド・プロシージャーを作成します。

```
CREATE PROCEDURE S1_INVENTORY
 SOURCE ProINVENTORY
 FOR SERVER ORA1;
```

   b. **DB2FEDGENTF** コマンドを使用して、PART_NUM と S1_QTY 結果セットを含む S1_INVTRY_TF 表関数を作成します。

```
DB2FEDGENTF –db sample –u user1 –p password1
  -create
    -stpn S1_INVENTORY
    -tfn S1_INVTRY_TF
    -c 'PART_NUM CHAR(10), S1_QTY INT'
```

2. 2 番目の供給業者に S2_INVTRY_TF 表関数を作成します。
   a. CREATE PROCEDURE ステートメントを使用して、ProINVENTORY2 ストアード・プロシージャーから S2_INVENTORY フェデレーテッド・プロシージャーを作成します。

```
CREATE PROCEDURE S2_INVENTORY
 SOURCE ProINVENTORY2
 FOR SERVER SYBA1;
```

   b. **DB2FEDGENTF** コマンドを使用して、PART_NUM と S2_QTY 結果セットを含む S2_INVTRY_TF 表関数を作成します。

```
DB2FEDGENTF –db sample –u user1 –p password1
  -create
    -stpn S2_INVENTORY
    -tfn S2_INVTRY_TF
    -c 'PART_NUM CHAR(10), S2_QTY INT'
```

3. 結合を使って、S1_INVTRY_TF 表関数の結果セットと S2_INVTRY_TF 表関数の結果セットを組み合わせます。

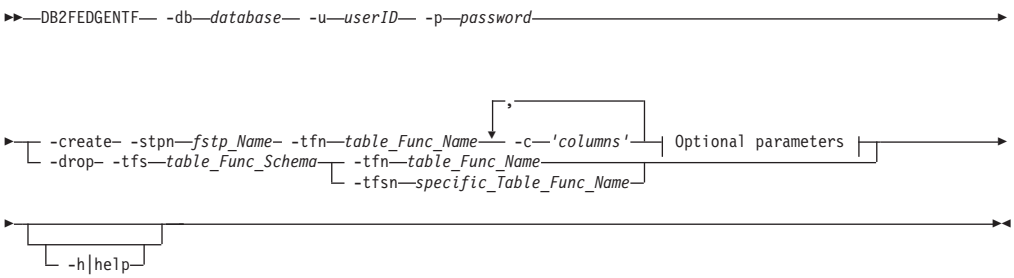
```
SELECT a.PART_NUM, a.PART_DESC, b.S1_QTY + c.S2_QTY as MAX_SUPP_QTY
 FROM PARTS a, TABLE(S1_INVTRY_TF()) b, TABLE(S2_INVTRY_TF()) c
 WHERE a. PART_NUM = b. PART_NUM
 AND a. PART_NUM = c. PART_NUM
```

DB2FEDGENTF コマンド

フェデレーテッド・プロシージャーから結果セットを返す表関数を作成および登録します。

db2fedgentf コマンドは、DBM 構成パラメーター JDK_PATH から JDK パスを取得します。

構文



Optional parameters:

パラメーター

-db *database*
接続先のデータベースの名前を指定します。

-u *userID*
フェデレーテッド・データベースのユーザー ID を指定します。

-p *password*
ユーザー ID のパスワードを指定します。

-create
-tfs パラメーターが指定されない場合に、現在のスキーマに表関数を作成および登録します。この表関数は、フェデレーテッド・プロシージャーからの結果セットの指定された列を返します。

-stpn *fstp_Name*
フェデレーテッド・プロシージャーの名前を指定します。

-tfn *table_Func_Name*
表関数の名前を指定します。表関数の名前が指定されない場合は、フェデレーテッド・プロシージャーの名前が使用されます。

-c '*columns***'**
フェデレーテッド・プロシージャーが返す結果セットのシグニチャーの列名と列タイプのペアを含むコンマ区切りのリストを指定します。

例 列名は PID、PRICE、QTY で、列タイプが CHAR(10)、DOUBLE、INT である場合。

```
'PID CHAR(10), PRICE DOUBLE, QTY INT'
```

-stps *fstp_Schema*
フェデレーテッド・プロシージャーのスキーマを指定します。このパラメーターはオプションです。名前を指定しないと、CURRENT SCHEMA 特殊レジスターで定義されたデフォルト SQL スキーマが使用されます。

-stpc *number_Fstp_Params*
フェデレーテッド・プロシージャーの入力の数を指定します。このパラメーターはオプションです。入力の数が指定されないと、フェデレーテッド・プロシージャーは指定されたフェデレーテッド・プロシージャー名によって決定されます。このフェデレーテッド・プロシージャーが多重定義されている場合は、エラーが返されます。

-tfs *table_Func_Schema*
表関数のスキーマを指定します。このパラメーターはオプションです。表関数のスキーマが指定されないと、CURRENT SCHEMA 特殊レジスターで定義されたデフォルト SQL スキーマが使用されます。

- drop
指定する表関数をドロップします。カタログからの記述も削除され、指定される表関数を参照するパッケージはすべて無効になります。

-tfs *table_Func_Schema*
ドロップする表関数のスキーマを指定します。表関数のスキーマが指定されないと、CURRENT SCHEMA 特殊レジスターで定義されたデフォルト SQL スキーマが使用されます。

-tfn *table_Func_Name*
ドロップする表関数の名前を指定します。

-tfsn *specific_Table_Func_Name*
多重定義関数の場合、ドロップする表関数の特定名を指定します。このパラメーターは、**-tfn** *table_Func_Name* パラメーターと相互に排他的です。表関数が名前とスキーマで一意的に識別される場合には、このオプションを指定する必要はありません。このパラメーターはオプションです。

-h|help
DB2FEDGENTF コマンドの使用法情報を指定します。

例

この例では、EAST_INVTRY フェデレーテッド・プロシージャーに **DB2FEDGENTF** コマンドを実行して、PRODID、PRICE、および QTY 列を返す表関数 S1_INVTRY_TF を作成します。

```
DB2FEDGENTF  -db sample  -u user1  -p password1
   -create
     -stpn EAST_INVTRY
     -tfn E_INVTRY_TF
     -c 'PRODID INT, PRICE DOUBLE, QTY INT'
```

フェデレーテッド・プロシージャーのトラブルシューティング

フェデレーテッド・プロシージャーに関する問題が発生した場合は、トラブルシューティングの方法がいくつかあります。

フェデレーテッド・プロシージャーに関する情報を確認するときに役立つ照会や診断ツールを以下に示します。その情報は、フェデレーテッド・プロシージャーに関する問題の解決に役立ちます。

データ・ソース・プロシージャーに関する情報の確認

CREATE PROCEDURE ステートメントの実行時に SQL1253N のエラーが戻された場合は、データ・ソースのカタログ表に以下の照会を実行して、データ・ソース・プロシージャーに関する情報を確認できます。SQL1253N は、CREATE PROCEDURE (ソース派生) ステートメントに指定されているソース・プロシージャーがデータ・ソースで検出されなかったという趣旨のエラーです。Oracle サーバーに対しては、照会を直接実行することも、パススルー・セッションを使用して照会を実行することも可能です。

DB2 for Linux, UNIX, and Windows 内のプロシージャーの場合:

```
SELECT parm_count, result_sets,
 sql_data_access, deterministic, external_action
 FROM syscat.routines
WHERE routineschema = ''
AND routinename = ''
AND routinetype = 'P'
AND parm_count = '' <-- optional
```

DB2 for iSeries® 内のプロシージャーの場合:

```
SELECT in_parms+out_parms+inout_parms,
number_of_results, sql_data_access, deterministic,
external_action
FROM qsys2.sysroutines
WHERE routine_schema = ''
and routine_name = ''
and routine_type = 'PROCEDURE'
and in_parms+out_parms+inout_parms = ''; <-- optional
```

DB2 for z/OS 内のプロシージャーの場合:

```
SELECT parm_count, result_sets,
 sql_data_access, deterministic, external_action
 FROM sysibm.sysroutines
WHERE schema = ''
AND name = ''
AND routinetype = 'P'
```

Microsoft SQL Server の場合

```
SELECT id
 FROM dbo.sysobjects
  WHERE id = object_id AND
   (TYPE = 'P' or TYPE = 'X')
```

パッケージに組み込まれている Oracle プロシージャーの場合:

```
SELECT owner, package_name, object_name, overload, parm_count
   FROM (
      SELECT owner, package_name, object_name, overload,
      SUM(case
         WHEN data_type IS NULL
         THEN 0
         ELSE 1
         END)
      AS parm_count
      FROM sys.all_arguments
      WHERE data_level = 0
      GROUP BY owner, package_name, object_name, overload
      ) aa
   WHERE object_name = '' AND
      package_name = '' AND
      owner = '' AND
      overload = '' AND   <-- optional
      parm_count =;       <-- optional
```

パッケージに組み込まれていない Oracle プロシージャーの場合:

```
SELECT object_name, object_type, status
   FROM sys.all_objects
   WHERE owner = '' AND
     object_name = '' AND
     object_type IN ('PROCEDURE', 'FUNCTION')
```

Sybase プロシージャーの場合:

```
SELECT id
   FROM dbo.sysobjects
   WHERE id = object_id('.') AND
     (TYPE = 'P' OR TYPE ='XP')
```

診断ツール

フェデレーテッド・プロシージャーに関する問題の診断には、Explain ユーティリティー、DESCRIBE コマンド、db2audit コマンドを使用できます。

例えば、FED_PROC1 プロシージャーに 3 つの OUTPUT パラメーターがあるとします。その FED_PROC1 プロシージャーに対して DESCRIBE コマンドを使用する場合は、以下のコマンドを実行します。

```
DESCRIBE CALL FED_PROC1(?,?,?);
```

システム・モニター

フェデレーテッド・データベースのシステム・モニター・エレメントには、フェデレーテッド・プロシージャーに関する情報が含まれています。モニター・エレメントは、以下のとおりです。

- 「ストアード・プロシージャー時間」モニター・エレメント stored_proc_time には、データ・ソースがフェデレーテッド・プロシージャー・ステートメントに応答するまでにかかった時間が含まれています。
- 「ストアード・プロシージャーによって戻された行数」モニター・エレメント sp_rows_selected には、データ・ソースからフェデレーテッド・サーバーに送信された行数が含まれています。このエレメントに基づいて、各フェデレーテッド・プロシージャーごとに、データ・ソースからフェデレーテッド・サーバーに送信

された平均行数を計算できます。さらに、データ・ソースからフェデレーテッド・サーバーに 1 行を戻すためにかかった平均時間を計算することも可能です。

- 「ストアード・プロシージャー数」モニター・エレメント stored_procs には、フェデレーテッド・サーバーがこのデータ・ソースから呼び出したプロシージャーの総数のカウントが含まれています。

SQL エラー SQL30090N (戻りコード 21)

SQL30090N エラー (戻りコード 21) が戻される状況はいくつかあります。最も一般的なのは、フェデレーテッド・プロシージャーを fenced ラッパーで作成しようとした場合です。フェデレーテッド・プロシージャーは、トラステッド・ラッパーでのみ作成できます。

結果セットが戻されない

結果セットがクライアントまたは呼び出し元に戻されない場合は、以下のいずれかの理由が考えられます。

- 結果セットを戻すための文節がフェデレーテッド・プロシージャーの中で正しく指定されていません。
- データ・ソースによっては、プロシージャーの呼び出しごとに、毎回同じ順序でセットが戻されるとは限りません。フェデレーテッド・プロシージャーは最初の結果セットだけを戻すので、フェデレーテッド・プロシージャーの呼び出しごとに、データ・ソースから異なる結果セットが戻される可能性があります。

例えば、データ・ソースに PROCEDURE A と PROCEDURE B という 2 つのプロシージャーがあり、PROCEDURE B が PROCEDURE A を呼び出すとします。それらのプロシージャーを作成するためのステートメントは、以下のようになります。

```
CREATE PROCEDURE A ()
BEGIN
   DECLARE cur1 CURSOR WITH RETURN TO CLIENT
   FOR SELECT * FROM t;
   OPEN cur1
END

CREATE PROCEDURE B (arg1 INT)
BEGIN
   DECLARE cur2 CURSOR WITH RETURN TO CLIENT
   FOR SELECT * FROM t;
   IF arg1<10) THEN
      CALL A();
   END IF;
   OPEN cur2
END;
```

フェデレーテッド・プロシージャー FEDPROC1 がデータ・ソース・プロシージャー PROCEDURE B を参照するとします。FEDPROC1 プロシージャーのステートメントは、以下のようになります。

```
CREATE PROCEDURE FEDPROC1
SOURCE newton.B
FOR SERVER s1
NUMBER OF PARAMETERS 1
WITH RETURN TO CLIENT 1;
```

ローカル・プロシージャーがフェデレーテッド・プロシージャー FEDPROC1 を呼び出すとします。そのローカル・プロシージャーのステートメントは、以下のようになります。

```
CREATE PROCEDURE local (arg1 INT)
   BEGIN
   CALL FEDPROC1 (arg1)
END;
```

CALL LOCAL(1) ステートメントを実行すると、PROCEDURE A から結果セット cur1 が戻されます。結果セット cur2 は戻されません。

一方、CALL LOCAL(20) ステートメントを実行すると、PROCEDURE B から結果セット cur2 が戻されます。

パススルー・セッション (Oracle のみ)

データ・ソースのプロシージャー、関数、パッケージをパススルー・セッションで作成する場合は、オブジェクト定義にエラーがあっても正常実行のメッセージが戻されます。 Oracle サーバーにオブジェクトが作成されますが、そのオブジェクトは INVALID とマークされます。INVALID のオブジェクトに対してフェデレーテッド・プロシージャーを作成することはできません。INVALID の Oracle オブジェクトを参照するフェデレーテッド・プロシージャーを作成しようとしても、CREATE PROCEDURE (ソース派生) ステートメントは失敗します。

オブジェクトが無効になっている理由を確認するには、以下のいずれかの方法を使用します。

- Oracle の SQL*Plus ユーティリティーで SHOW ERRORS コマンドを使用します。
- Oracle の sys.all_errors カタログ表を照会します。

第 7 章 透過 DDL を使用したリモート表の作成および変更

透過 DDL を使用すると、既によく知っている手順を使用してリモート表を作成および変更できます。これは、パススルー・セッションを使用せずにフェデレーテッド・データベースを使用して行うことができます。

透過 DDL とは

透過 DDL を使用すると、パススルー・セッションを使用せずにフェデレーテッド・データベースでリモート表を作成、変更することができます。

透過 DDL でユーザーが使用する SQL ステートメントは、CREATE TABLE、ALTER TABLE、および DROP TABLE です。

透過 DDL の CREATE TABLE ステートメントは、データ・ソース側にリモート表を作成し、さらにフェデレーテッド・サーバー側にその表のニックネームを作成します。このステートメントは、デフォルトの逆方向タイプ・マッピングを使用して、指定した DB2 データ・タイプをリモート・データ・タイプにマップします。通常は、ラッパーがタイプ・マッピングを提供します。ユーザー定義の逆方向タイプ・マッピングを作成して、デフォルトのマッピングをオーバーライドすることもできます。

透過 DDL を使用する利点は、データベース管理者が、慣れた手順に従って、ローカル表とリモート表の両方を作成できる点にあります。透過 DDL を使用すると、表管理を集中化できるため、許可の付与が容易になります。

透過 DDL は、以下のデータ・ソースでサポートされています。

- DB2 for z/OS
- DB2 for System i
- DB2 Database for Linux, UNIX, and Windows
- DB2 Server for VM and VSE
- Informix
- JDBC
- Microsoft SQL Server
- ODBC
- Oracle
- Sybase
- Teradata

データベース管理者は、DB2 コントロール・センターを使用するか、 DB2 コマンド行プロセッサー (CLP) で DDL ステートメントを使用して、表を作成できます。透過 DDL を使用すると、各データ・ソースで必要とされる異なる DDL 構文を学ぶ必要がなくなります。

フェデレーテッド・データベースを使用してデータ・ソース上にリモート表を作成するには、次のようにデータ・ソースへのアクセスを構成しておく必要があります。

- データ・ソース用のラッパーをグローバル・カタログに登録する
- リモート表を置くサーバーのサーバー定義を作成する
- 必要な場合は、フェデレーテッド・サーバーとデータ・ソース・サーバーの間にユーザー・マッピングを作成する

リモート表を作成するには、 DB2 コントロール・センターの「リモート表」ウィザードを使用してください。

透過 DDL ステートメントの許可 ID が持つ特権には、少なくとも以下の 1 つが含まれている必要があります。

- SYSADM または DBADM 権限
- データベースに対する CREATETAB 権限および表スペースに対する USE 特権に加えて、以下のいずれか。
  - データベースに対する IMPLICIT_SCHEMA 権限 (表の暗黙的または明示的スキーマ名が存在しない場合)
  - スキーマに対する CREATEIN 特権 (表のスキーマ名が既存のスキーマを指す場合)

透過 DDL ステートメントを発行するには、許可 ID がニックネームに必要な特権 (フェデレーテッド・サーバーが要求を受け入れるための) およびリモート・データ・ソース・サーバーに同等の特権 (データ・ソースが要求を受け入れるため) を持っている必要があります。

リモート LOB 列および透過 DDL

透過 DDL を使用する際には、LOB 列の長さを指定します。

一部のデータ・ソース (Oracle および Informix など) では、 LOB 列の長さはシステム・カタログに保管されません。表に対してニックネームを作成する場合は、データ・ソース・システム・カタログの情報が列の長さも含めて検索されます。 LOB 列については長さが存在しないため、フェデレーテッド・データベースでは、長さは DB2 Database for Linux, UNIX, and Windows の LOB 列の最大長であると想定されます。フェデレーテッド・データベースは、最大長をニックネーム列の長さとしてフェデレーテッド・データベース・カタログに保管します。

ただし、透過 DDL を使用してリモート表を作成する場合は、LOB 列の長さを指定する必要があります。フェデレーテッド・サーバーがリモート表に対してニックネームを作成する場合は、フェデレーテッド・データベース・カタログに指定した長さがニックネーム列の長さとして保管されます。 LOB 列の最大長は 2 ギガバイトです。

リモート表と透過 DDL の作成

リモート表が、透過 DDL を使用してフェデレーテッド・データベースで作成された場合、その他のアクションがいくつか発生します。

リモート表を作成すると、次のようになります。

- リモート表に対して、ニックネームが自動的に作成されます。ニックネームの名前は、CREATE TABLE ステートメントで指定されている表名と同じです。REMOTE_TABNAME オプションを使用して別の名前を指定しなければ、リモート表の名前は表名と同じになります。
- REMOTE_SCHEMA オプションを使用して別のスキーマを指定しなければ、リモート表のスキーマはニックネームのスキーマになります。
- 透過 DDL を使用して作成されたニックネームは、他のニックネームと同様に使用できます。加えて、リモート表を ALTER および DROP できます (これは CREATE NICKNAME で作成されたニックネームではできません)。
- SYSCAT.TABOPTIONS カタログ・ビューに、オプション名 TRANSPARENT と値 Y が付いた行が追加されます。

透過 DDL を使用した新規リモート表の作成

透過 DDL を使用してリモート表を作成するには、CREATE TABLE ステートメントを指定します。

始める前に

リモート表の作成の前に、そのデータ・ソースにアクセスするようにフェデレーテッド・サーバーを構成する必要があります。これには、次の作業が含まれます。

- データ・ソース・タイプ用のラッパーを作成する
- リモート表を置くサーバーのサーバー定義を提供する
- フェデレーテッド・サーバーとデータ・ソース・サーバー間のユーザー・マッピングを作成する

透過 DDL ステートメントを発行するには、許可 ID がニックネームに必要な特権 (フェデレーテッド・サーバーが要求を受け入れるための) およびリモート・データ・ソース・サーバーに同等の特権 (データ・ソースが要求を受け入れるため) を持っている必要があります。

透過 DDL ステートメントを発行する許可 ID が持つ特権には、少なくとも以下の 1 つが含まれている必要があります。

- SYSADM または DBADM 権限
- データベースに対する CREATETAB 権限および表スペースに対する USE 特権に加えて、以下のいずれか。
  - データベースに対する IMPLICIT_SCHEMA 権限 (表の暗黙的または明示的スキーマ名が存在しない場合)
  - スキーマに対する CREATEIN 特権 (表のスキーマ名が既存のスキーマを指す場合)

このタスクについて

制約事項

透過 DDL を使用したリモート表の作成には、以下の制約事項があります。

- 元々リモート・データ・ソースで作成された表を変更またはドロップすることはできない。
- マテリアライズ照会表をリモート・データ・ソースに作成することはできない。
- 表の定義に基本的な列情報を指定できるが、表オプションや列オプションを指定することはできない。例えば、LOB オプション (LOGGED および COMPACT) はサポートされていません。
- 列にコメントを指定することはできない。
- 列の内容を生成することはできない。
- 主キーを指定することはできるが、外部キーやチェック制約を指定することはできない。主キーに使用される列は、NOT NULL でなければならず、LOB を含む列にはできません。
- データ・タイプやデータ長などの、既存の列のパラメーターを変更することはできない。
- CREATE TABLE ステートメントでは DEFAULT 文節はサポートされない。

手順

リモート表をコマンド行プロンプトから作成するには、 CREATE TABLE ステートメントに適切なパラメーター・セットを指定して発行します。

透過 DDL を使用した新規リモート表の作成例

以下の例は、透過 DDL を使用したリモート表の作成およびデータ・タイプ・マッピングの使用を指定する方法を示しています。

透過 DDL を使用してリモート表を作成する際には、次のようにします。

- リモート・データ・ソースは、CREATE TABLE ステートメント内の列データ・タイプおよび主キー・オプションをサポートする必要があります。

  例: リモート・データ・ソースが主キーをサポートしないとします。データ・ソースの、サポートしていない要求への応答方法によって、エラーが戻されるか、または要求が無視される可能性があります。
- リモート・サーバーは、OPTIONS 文節に指定する必要があります。作成される表のリモート名またはリモート・スキーマをオーバーライドするには、OPTIONS 文節を使用できます。 SQL_SUFFIX オプションは、CREATE TABLE ステートメントの末尾で使用できます。リレーショナル・データ・ソースにこのオプションを指定して、データ・ソースで発行される CREATE TABLE ステートメントにデータ・ソース固有のオプションを追加できます。

例: Oracle サーバー上に表 EMPLOY を作成したいと仮定します。CREATE TABLE ステートメントで、各列を指定するときに DB2 データ・タイプを使用します。CLP を使用して表を作成する際の構文は以下のとおりです。

```
CREATE TABLE EMPLOY
   ( EMP_NO       CHAR(6) NOT NULL,
     FIRSTNAME    VARCHAR(12) NOT NULL,
     MIDINT       CHAR(1) NOT NULL,
     LASTNAME     VARCHAR(15) NOT NULL,
     HIREDATE     DATE,
     JOB          CHAR(8),
     SALARY       DECIMAL(9,2),
     PRIMARY KEY (EMP_NO) )
     OPTIONS (REMOTE_SERVER 'ORASERVER',
     REMOTE_SCHEMA 'J15USER1', REMOTE_TABNAME 'EMPLOY' )
```

EMPLOY
表に関連したニックネームの名前。

REMOTE_SERVER *'ORASERVER'*
CREATE SERVER ステートメントでサーバーに指定した名前。この値には大文字小文字の区別があります。

REMOTE_SCHEMA *'J15USER1'*
リモート・スキーマ名。このパラメーターはオプションですが、スキーマ名を指定することをお勧めします。このパラメーターが指定されていないと、ニックネーム・スキーマがリモート・スキーマ名に使用されます。この値には大文字小文字の区別があります。

REMOTE_TABNAME *'EMPLOY'*
リモート表名。このパラメーターはオプションです。このパラメーターが指定されていないと、ローカル表名がリモート表名に使用されます。この値は、データ・ソースで有効な名前で、既存の表名であってはなりません。この値には大文字小文字の区別があります。

上の例では、フェデレーテッド・データベースは DB2 データ・タイプから Oracle データ・タイプにマップするのに、逆方向データ・タイプ・マッピングを使用します。リモート Oracle サーバーでは、 EMPLOY 表は Oracle データ・タイプを使用して作成されます。次の表に、例に指定されている列に関して、 DB2 データ・タイプから Oracle データ・タイプへのマッピングを示します。

表 6. フェデレーテッド・データベースから Oracle への逆方向データ・タイプ・マッピングの例

列	CREATE TABLE ステートメントに指定された DB2 データ・タイプ	リモート表で使用される Oracle データ・タイプ
EMP_NO	CHAR(6) NOT NULL	CHAR(6) NOT NULL
FIRST_NAME	VARCHAR(12) NOT NULL	VARCHAR2(12) NOT NULL
MID_INT	CHAR(1) NOT NULL	CHAR(1) NOT NULL
LAST_NAME	VARCHAR(15) NOT NULL	VARCHAR2(15) NOT NULL
HIRE_DATE	DATE	DATE
JOB	CHAR(8)	CHAR(8)
SALARY	DECIMAL(9,2)	NUMBER(9,2)

透過 DDL を使用したリモート表の変更

フェデレーテッド・データベースを介して作成されたリモート・データ・ソース表は、透過 DDL を使用して変更できます。リモート・データ・ソースで直接作成された表を変更することはできません。

始める前に

透過 DDL ステートメントの許可 ID が持つ特権には、少なくとも以下の 1 つが含まれている必要があります。

- SYSADM または DBADM 権限
- データベースに対する CREATETAB 権限および表スペースに対する USE 特権に加えて、以下のいずれか。
  - データベースに対する IMPLICIT_SCHEMA 権限 (表の暗黙的または明示的スキーマ名が存在しない場合)
  - スキーマに対する CREATEIN 特権 (表のスキーマ名が既存のスキーマを指す場合)

透過 DDL ステートメントを発行するには、許可 ID がニックネームに必要な特権 (フェデレーテッド・サーバーが要求を受け入れるための) およびリモート・データ・ソース・サーバーに同等の特権 (データ・ソースが要求を受け入れるため) を持っている必要があります。

このタスクについて

IBM InfoSphere Federation Server で作成した表を透過 DDL を使用して変更するには、ALTER TABLE ステートメントを使用できます。ALTER TABLE ステートメントを使用すると、次の作業を行うことができます。

- 新規列の追加。
- 表主キーの追加。

列オプションを追加または変更するのに、ALTER TABLE ステートメントを使用しないでください。代わりに ALTER NICKNAME ステートメントを使用してください。

制約事項

透過 DDL を使用したリモート表の変更には、以下の制約事項があります。

- もともとリモート・データ・ソースで作成された表を変更することはできない。
- 既存の主キーは、リモート表内で変更またはドロップすることはできない。
- リモート表の変更を行うと、リモート表に関連したニックネームに依存しているパッケージはいずれも無効になる。
- リモート・データ・ソースは、ALTER TABLE ステートメント内の変更をサポートする必要がある。例えば、リモート・データ・ソースが主キーをサポートしないとします。データ・ソースの、サポートしていない要求への応答方法によって、エラーが戻されるか、または要求が無視される可能性があります。
- 列にコメントを指定することはできない。
- 列の内容を生成することはできない。

- 主キーを指定することはできるが、外部キーやチェック制約を指定することはできない。主キーに使用される列は、NOT NULL でなければならず、LOB を含む列にはできません。
- データ・タイプやデータ長などの、既存の列のパラメーターを変更することはできない。
- ALTER TABLE ステートメントでは DEFAULT 文節はサポートされない。

手順

透過 DDL を使用してリモート表を変更するには、ALTER TABLE ステートメントを発行します。
例: 透過 DDL を使用して作成したリモート表 EMPLOYEE に主キーを追加したいと仮定します。表を変更するには、以下の ALTER TABLE ステートメントを使用します。

```
ALTER TABLE EMPLOYEE
   ADD PRIMARY KEY (EMP_NO, WORK_DEPT)
```

主キーに使用される列は、NOT NULL でなければならず、LOB を含む列にはできません。
例: 透過 DDL を使用して作成したリモート表 SPALTEN に、列 ORDER_DATE と列 SHIP_DATE を追加したいと仮定します。表を作成するには、以下の ALTER TABLE ステートメントを使用します。

```
ALTER TABLE SPALTEN
   ADD COLUMN ORDER_DATE  DATE
   ADD COLUMN SHIP_DATE  DATE
```

透過 DDL を使用したリモート表のドロップ

フェデレーテッド・データベースを介して作成されたリモート・データ・ソース表は、透過 DDL を使用してドロップできます。リモート・データ・ソースで直接作成された表をドロップすることはできません。

始める前に

透過 DDL ステートメントの許可 ID が持つ特権には、少なくとも以下の 1 つが含まれている必要があります。

- SYSADM または DBADM 権限
- データベースに対する CREATETAB 権限および表スペースに対する USE 特権に加えて、以下のいずれか。
  - データベースに対する IMPLICIT_SCHEMA 権限 (表の暗黙的または明示的スキーマ名が存在しない場合)
  - スキーマに対する CREATEIN 特権 (表のスキーマ名が既存のスキーマを指す場合)

透過 DDL ステートメントを発行するには、許可 ID がニックネームに必要な特権 (フェデレーテッド・サーバーが要求を受け入れるための) およびリモート・データ・ソース・サーバーに同等の特権 (データ・ソースが要求を受け入れるため) を持っている必要があります。

このタスクについて

透過 DDL を使用してフェデレーテッド・データベースで作成されたリモート表をドロップするには、 DB2 コントロール・センターまたは DROP ステートメントのどちらかを使用できます。

透過 DDL を使用して作成されたリモート表のニックネームをドロップすると、単にその表のローカル・ニックネームがドロップされます。 DROP NICKNAME ステートメントでは、リモート表をドロップできません。リモート表をドロップするには、DROP TABLE ステートメントを使用する必要があります。

リモート表を先にドロップすると、データ・ソース上の表が削除され、それからフェデレーテッド・データベース内のリモート表の対応するニックネームが削除されます。ニックネームを削除すると、そのニックネームを基にしたパッケージが無効になります。

制約事項

もともとリモート・データ・ソースで作成された表をドロップすることはできない。

手順

リモート表をドロップするには、DROP TABLE ステートメントを発行します。
例: SPALTEN という名前の表をドロップするには、以下の DROP ステートメントを発行します。

```
DROP TABLE SPALTEN
```

SPALTEN はリモート表のローカル名です。

第 8 章 フェデレーテッド・システム内のトランザクションの管理

フェデレーテッド・システム・トランザクション処理により、データの整合性を保守しながら、単一トランザクションでデータベースを読み取って更新できます。フェデレーテッド・システムでは、1 フェーズ・コミット・プロトコルと 2 フェーズ・コミット・プロトコルがサポートされます。フェデレーテッド・システムを管理する際には、該当するプロトコルをセットアップする必要があります。

フェデレーテッド・システム・トランザクション・サポートの概要

DB2 Database for Linux, UNIX, and Windows 分散環境におけるトランザクション処理概念の知識は、フェデレーテッド・システムのトランザクションを理解する上で助けとなります。

フェデレーテッド・システムのトランザクション処理について理解するには、以下の分散トランザクション処理の概念に精通している必要があります。

- 作業単位 (UOW)
- リモート作業単位 (RUOW)
- 分散作業単位 (DUOW)
- マルチサイト更新
- トランザクション・マネージャー (TM)
- リソース・マネージャー (RM)
- タイプ 1 接続
- タイプ 2 接続
- 1 フェーズ・コミット
- 2 フェーズ・コミット

これらの概念は、フェデレーテッド・データベース・システムでも非フェデレーテッド・データベース・システムでも同じです。ただし、フェデレーテッド・システムでは各概念の対象範囲が異なってきます。

例えば、作業単位は、データベース内の何らかのデータが読み取られた、または書き込まれた時に、暗黙的に開始されます。フェデレーテッド・システムの作業単位の場合は、データベースはフェデレーテッド・データベースまたはデータ・ソース・データベースのどちらかです。フェデレーテッド・システムの分散作業単位の場合は、フェデレーテッド・データベースとデータ・ソース・データベースの両方にアクセスできます。

アプリケーションは、アクセスされるデータベースの数に関係なく、 COMMIT または ROLLBACK ステートメントを実行して、作業単位を終了させる必要があります。 COMMIT ステートメントは、作業単位内のすべての変更を永続的なものにし

ます。 ROLLBACK ステートメントは、それらの変更をデータベースから除去します。作業単位で行われた変更は、コミットが成功した後、他のアプリケーションにも見えるようになります。

推奨事項: アプリケーションでは、必ず明示的に作業単位をコミットまたはロールバックするようにしてください。

複数のサイトにわたる複数のデータベースの更新が伴う分散作業単位では、データの整合性が必要です。マルチサイト更新または 2 フェーズ・コミット・プロトコルは、分散作業単位内の複数のデータベースにまたがるデータの整合性を保証するためによく使用されます。

フェデレーテッド・トランザクションは、1 フェーズ・コミット・プロトコルと 2 フェーズ・コミット・プロトコルの両方をサポートします。
DB2_TWO_PHASE_COMMIT サーバー・オプションを使用すると、以下のデータ・ソースにおける 2 フェーズ・コミットのサポートが使用可能になります。

- DB2 ファミリーのデータ・ソース (fenced モードとトラステッド・モード)
- Informix (トラステッド・モード)
- Oracle (fenced モードとトラステッド・モード)
- Sybase (fenced モード)
- MS SQL Server (トラステッド・モード)

データ・ソースがフェデレーテッド 2 フェーズ・コミット・データ・ソースとして宣言されている時、つまり、DB2_TWO_PHASE_COMMIT サーバー・オプションが『Y』 に設定されている時、このデータ・ソースに対するコミットでは、単一サイトの更新トランザクションであっても複数サイトの更新トランザクションであっても、2 フェーズ・コミット・プロトコルを使用します。

データ・ソースがフェデレーテッド 1 フェーズ・コミット・データ・ソースとして宣言されていて (デフォルト)、これが単一サイトの更新トランザクションである場合、このデータ・ソースに対するコミットでは、1 フェーズ・コミット・プロトコルを使用します。

以下の 1 フェーズ・コミット操作例では、Oracle は 1 フェーズ・コミット・データ・ソースとして定義されています。

```
SELECT * FROM oracle_nickname
UPDATE oracle_nickname
COMMIT
```

以下の 2 フェーズ・コミット操作例では、Oracle および DRDA は 2 フェーズ・コミット・データ・ソースとして定義されています。

```
SELECT * FROM oracle_nickname
UPDATE oracle_nickname

SELECT * FROM drda_nickname
UPDATE drda_nickname
COMMIT
```

フェデレーテッド・システムでの更新とは ?

フェデレーテッド・システムでは、更新とは INSERT、UPDATE、または DELETE ステートメントが組み込まれた単なるトランザクションではありません。フェデレーテッド・システムにおいて更新と見なされる特定の操作があり、フェデレーテッド・システムにおいて許可されている特定のタイプの更新があります。

フェデレーテッド・システムでは、更新は、ローカルでもリモートでも行えます。

- ローカル・サイト更新は、ニックネームを参照しない DB2 表またはビューに対する更新です。
- リモート・サイト更新は、リモート・データ・ソース上のオブジェクトに対する更新です。リモート・データ・ソースには、以下が含まれます。
  - フェデレーテッド・サーバー上の別の DB2 Database for Linux, UNIX, and Windows のデータベースまたはインスタンス
  - 別のサーバー上の別の DB2 Database for Linux, UNIX, and Windows のデータベースまたはインスタンス
  - DB2 Database for Linux, UNIX, and Windows 以外のデータ・ソース (DB2 for System i、Informix、Oracle、および Teradata など)

フェデレーテッド・サーバーが更新トランザクションとみなすアクションには、4 つのタイプがあります。次の表に、フェデレーテッド・システムで実行できる更新のタイプを示します。

表 7. 更新のタイプと更新が実行されるサイト

アクションのタイプ	ローカル・サイト	リモート・サイト	説明
ローカル更新 (DDL および DML)	Y	N	フェデレーテッド・データベース内のオブジェクトに対する更新。
リモート更新 (ニックネーム)	N	Y	ニックネームを作成したリモート・データ・ソース・オブジェクトに対する更新。
パススルー・セッションでの動的 SQL	N	Y	リモート・データ・ソース・オブジェクトに対する更新。パススルー・セッションを使用して、ローカル・オブジェクトを更新することはできません。パススルー・セッションで送信された SELECT 照会でさえ更新アクションと見なされます。
透過 DDL	Y	Y	リモート表およびフェデレーテッド・データベース内の対応するニックネームを作成、変更、またはドロップするトランザクションの対。例えば、データ・ソース上にリモート表を作成し、フェデレーテッド・サーバー上にニックネームを作成するトランザクションの対など。

パススルー・セッションでの更新トランザクションとは ?

フェデレーテッド・サーバーは、パススルー・セッションを介して送信されるすべての動的 SQL ステートメントを更新として扱います。この動作によって、データ保全性が保証されます。

パススルー・セッションを介して送信される動的 SQL ステートメントが成功すると、トランザクションは更新として記録されます。 SQL は、SELECT ステートメントも含めたどのタイプのステートメントでも構いません。

DDL ステートメントを自動的にコミットするデータ・ソース

ある種のデータ・ソース (Oracle など) は、DDL ステートメント実行の一環として、現行のトランザクションをそのデータ・ソース・サイトで自動的にコミットします。

リモート表の作成に透過 DDL を使用するか、またはそれをパススルー・セッションで行う場合は、これらのデータ・ソースは表の作成後にリモート表をロールバックできません。リモート表を手動で削除する必要があります。

処理のためにデータ・ソースにプッシュダウンされるユーザー定義関数

リモート・ユーザー定義関数がデータ・ソースに対して更新を実行する場合は、フェデレーテッド・サーバーは更新を認識しません。

フェデレーテッド・サーバーはこれらのユーザー定義関数を更新ステートメントとして扱わないため、フェデレーテッド・システムが更新操作に適用するすべてのステートメント・レベルの保護は、適用されません。結果として、ある状況ではデータ保全性が犠牲になる可能性があります。

重要: データ・ソースにプッシュダウンされるユーザー定義関数が更新を実行する場合は、データ保全性は保証されません。

第 9 章 2 フェーズ・コミット・トランザクションの実行

フェデレーテッド・システムの 2 フェーズ・コミット機能を使用して、単一トランザクションで 1 つ以上のデータ・ソースのデータを更新できます。

フェデレーテッド・トランザクションにおける 2 フェーズ・コミット

フェデレーテッド・システムは、1 つ以上のデータ・ソースにアクセスするトランザクションにおいて 2 フェーズ・コミットを使用することができます。 2 フェーズ・コミットでは、業界標準 X/Open XA プロトコルを使用して分散作業単位トランザクションの処理を調整します。

2 フェーズ・コミット操作では、コミット処理は、準備フェーズとコミット・フェーズという 2 つのフェーズで行われます。フェデレーテッド・システムにおける準備フェーズ中、フェデレーテッド・サーバーは、トランザクションに関係するフェデレーテッド 2 フェーズ・コミット・データ・ソースをすべてポーリングします。このポーリング・アクティビティーでは、各データ・ソースがデータをコミットまたはロールバックする準備ができているかを確認します。コミット・フェーズ中、フェデレーテッド・サーバーは、各 2 フェーズ・コミット・データ・ソースに対し、データのコミットまたはトランザクションのロールバックのいずれかを指示します。

1 フェーズ・コミット環境では、複数のデータ・ソースは別個のコミット操作を使用して一度に 1 つずつ更新されます。このため、一部のデータ・ソースが正常に更新され、残りが更新されなかった場合には、データ同期の問題が発生する場合があります。

例えば、あるトランザクションが 1 フェーズ・コミットを使用して 1 つの口座から資金を引き出し、それを別の口座に預金する場合、システムは引き出し操作のコミットには成功するものの、預金操作のコミットには失敗するかもしれません。引き出し操作は既に正常にコミットされているため、預金操作はロールバックできますが、引き出し操作はロールバックできません。結果として、その資金は実質上「失われて」しまいます。

2 フェーズ・コミット環境では、引き出しおよび預金のトランザクションは一緒に準備され、コミットまたはロールバックのいずれかが一緒に行われます。結果として、資金額の整合性は保たれます。

フェデレーテッド 2 フェーズ・コミットの計画

フェデレーテッド 2 フェーズ・コミットは、すべてのビジネス環境で役立つわけではありません。また、フェデレーテッド 2 フェーズ・コミットをデプロイする決定をする前に考慮するいくつかの要素もあります。

フェデレーテッド・トランザクションで 2 フェーズ・コミットを使用するためには、以下の問題を考慮する必要があります。

- オペレーティング・システムおよびデータ・ソース環境がフェデレーテッド・トランザクションでの 2 フェーズ・コミットをサポートできるか。
- ビジネス環境がフェデレーテッド・トランザクションの 2 フェーズ・コミットを必要とするか。

  ご使用のビジネス環境に 2 フェーズ・コミットが適切であるかどうかを決定するため、フェデレーテッド・トランザクションにおける 2 フェーズ・コミットの動作方法、および 2 フェーズ・コミットが解決する問題について理解する必要があります。
- フェデレーテッド・トランザクションで 2 フェーズ・コミットを使用するために、どのようにフェデレーテッド・サーバーおよび互換性のあるデータ・ソースを構成するか。

  フェデレーテッド・トランザクションで 2 フェーズ・コミットを使用するためのフェデレーテッド・サーバーとデータ・ソースの基本的な要件に加えて、2 フェーズ・コミットをデプロイする際に考慮する必要があるパフォーマンスの考慮事項があります。
- 未確定トランザクションを手動で解決するためには、フェデレーテッド・トランザクションの 2 フェーズ・コミットの内部動作を理解する必要がある。

  フェデレーテッド・トランザクションの 2 フェーズ・コミットでは、手動の介入なく問題を解決できます。ただし、長期に渡るネットワーク障害、ハードウェア障害、またはシステム・リソースを緊急に解放する必要が生じた場合には、ヒューリスティック処理 により、手動で問題を解決することができます。

2 フェーズ・コミットにおけるフェデレーテッド・アーキテクチャー

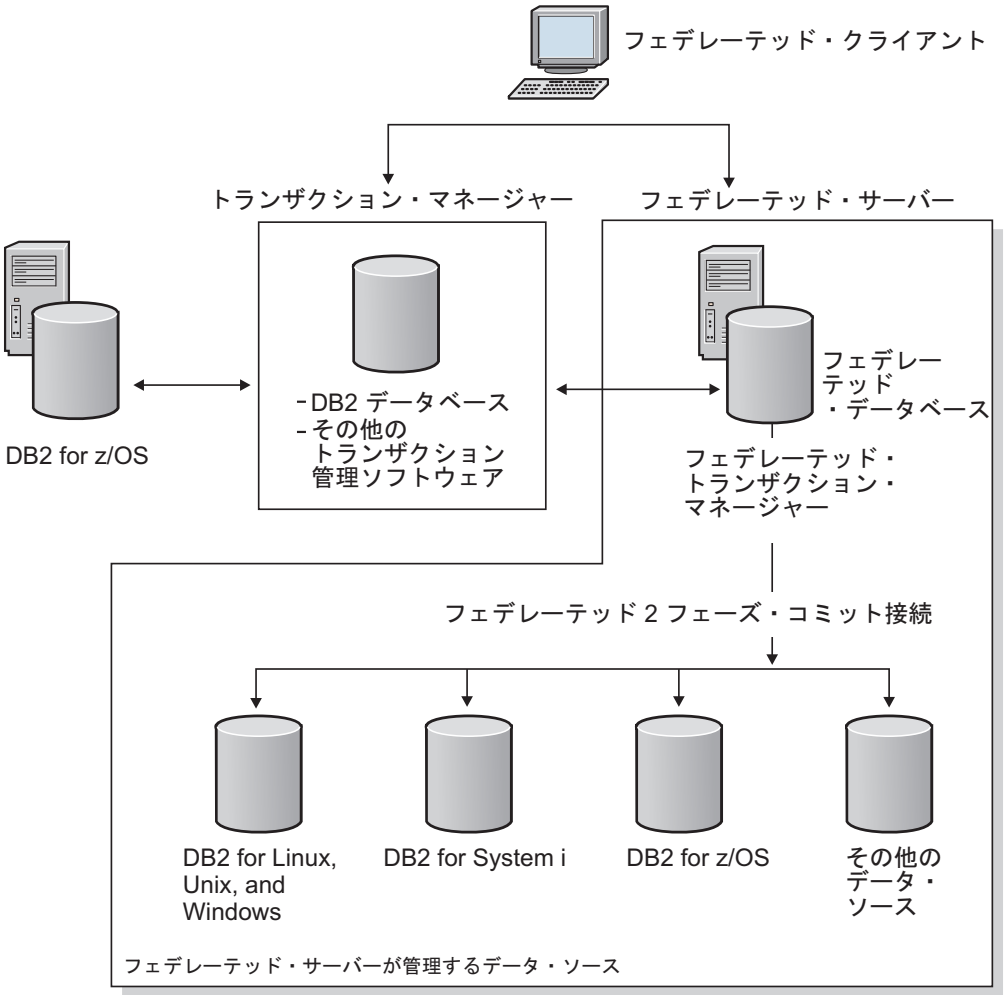
フェデレーテッド 2 フェーズ・コミットは、DB2 で使用可能な 2 フェーズ・コミット・フィーチャーに基づいています。 2 フェーズ・コミットの場合、X/Open 分散トランザクション処理 (DTP) モデルには、トランザクション ID、トランザクション・マネージャー、およびリソース・マネージャーという複数のコンポーネントがあります。フェデレーテッド 2 フェーズ・コミットを使用するフェデレーテッド・システムでは、フェデレーテッド・トランザクション・マネージャーという別のコンポーネントが追加されます。

2 フェーズ・コミット・プロトコルを使用する 1 つ以上のリモート・データ・ソースのアクティビティーをフェデレーテッド・サーバーが調整する場合、このサーバーがフェデレーテッド・トランザクション・マネージャーになります。フェデレーテッド・トランザクション・マネージャーは、トランザクション・マネージャーの代わりに一部のトランザクション管理機能を実行します。分散作業単位トランザクションを開始するクライアントまたはアプリケーションと、トランザクション・マネージャーは、フェデレーテッド・トランザクション・マネージャーがリモート・データ・ソースで調整するアクティビティーを認識しません。フェデレーテッド・トランザクション・マネージャーは、XA インターフェースを使用して DB2 データベース・トランザクション・マネージャーと通信します。2 フェーズ・コミットに関する X/Open 要件に加えて、トランザクション・マネージャー・データベースはフェデレーテッド・インスタンスからアクセスできなければなりません。リソー

ス・マネージャーは、フェデレーテッド・トランザクション・マネージャーが出す指示に従って、トランザクションのコミットまたはロールバックを行います。

以下の図は、標準的なフェデレーテッド・システムにおける、クライアントの開始からデータ・ソースの更新に至るまでの単純な 2 フェーズ・コミット・トランザクションの例を示しています。

図 3. 単純なフェデレーテッド 2 フェーズ・コミット・トランザクション



直前の図では、クライアントからトランザクション・マネージャーへの接続は、タイプ 2 接続です。各データベース接続にも独自の同期点設定があります。同期点とは、プログラムがアクセスするすべてのリカバリー可能データが整合している時点です。同期点の 2 フェーズ接続は、複数のデータ・ソースへの更新を含む分散作業単位トランザクションをサポートします。

クライアントが DB2 データベースに接続する際、トランザクション・マネージャーはトランザクションを認識しますが、フェデレーテッド・サーバーからの追加の調整が求められることはありません。フェデレーテッド・サーバーが 2 フェーズ・コミット・プロトコルを使用してデータ・ソースに接続する際、フェデレーテッド・サーバーはフェデレーテッド・トランザクション・マネージャーになります。フェデレーテッド・サーバーは、2 フェーズ・コミットをモニターし、調整しま

す。この時点で、トランザクション・マネージャーは、データ・ソースとの 2 フェーズ・コミット・トランザクションを認識していません。トランザクション・マネージャーは、フェデレーテッド・サーバーとの単一トランザクションが処理されていることしか認識していません。

データ・ソースは、フェデレーテッド・システムで障害が発生した場合に再同期を開始することができません。フェデレーテッド・サーバーが再同期プロセスを開始します。

フェデレーテッド 2 フェーズ・コミットを使用する同じトランザクションで複数のパスを使用してデータ・ソースへのアクセスを試行する場合、結果は予測不能となる場合があります。例えば、フェデレーテッド・サーバーが外部トランザクション・マネージャーに対するリソース・マネージャーとなっている場合、データ・ソースはフェデレーテッド・サーバーから、間接的なアクセスと、トランザクション・マネージャーに対するリソース・マネージャーとして直接的なアクセスを受ける可能性があります。この場合、データ・ソースは、これら 2 つのパスが同じグローバル・トランザクションからのものであるかを判別できない可能性があります。データ・ソースは同じグローバル・トランザクションに対して 2 つのトランザクション項目を作成し、それぞれのトランザクションを別個のものとして扱う可能性があるため、予測不能の結果になる可能性があります。データ・ソースは、2 つのパスは同じグローバル・トランザクションからのものであることを検出して 2 番目のパスをリジェクトする可能性もあります。

フェデレーテッド・トランザクションにおける 2 フェーズ・コミット - 例

2 フェーズ・コミットを使用するフェデレーテッド・システムは、いくつかの異なる方法で構成できます。構成の選択項目は、必要なソリューションにより異なります。

構成は、タイプ 1 またはタイプ 2 接続を使用することができます。

タイプ 1 接続は、アプリケーション・プロセスがリモート作業単位の規則に従ってアプリケーション・サーバーに接続される接続です。

タイプ 2 接続は、アプリケーション・プロセスがアプリケーション・サーバーに接続され、アプリケーション制御の分散作業単位の規則を確立する接続です。アプリケーション・サーバーはその時、プロセスの現行サーバーとなります。

以下の図は、トランザクション・マネージャーとして機能しているフェデレーテッド・サーバーとの DB2 タイプ 1 接続を示しています。

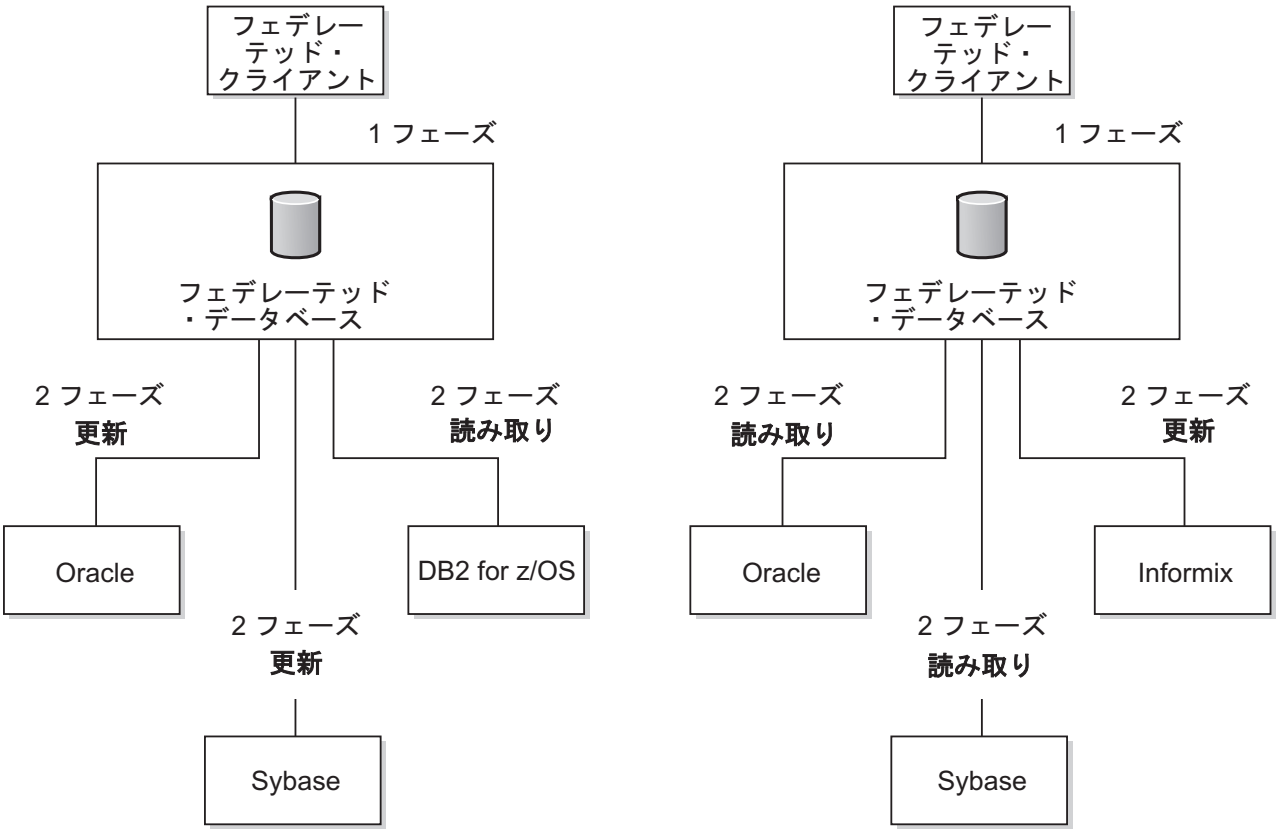


図 4. トランザクション・マネージャーの役目を果たすフェデレーテッド・サーバーとの DB2 タイプ 1 接続

以下の図は、リソース・マネージャーとして機能しているフェデレーテッド・サーバーとの DB2 タイプ 2 接続を示しています。この構成では、すべてのフェデレーテッド・データ・ソースでフェデレーテッド 2 フェーズ・コミットがサポートされていて、さらにフェデレーテッド 2 フェーズ・コミットが使用可能になっている必要があります。

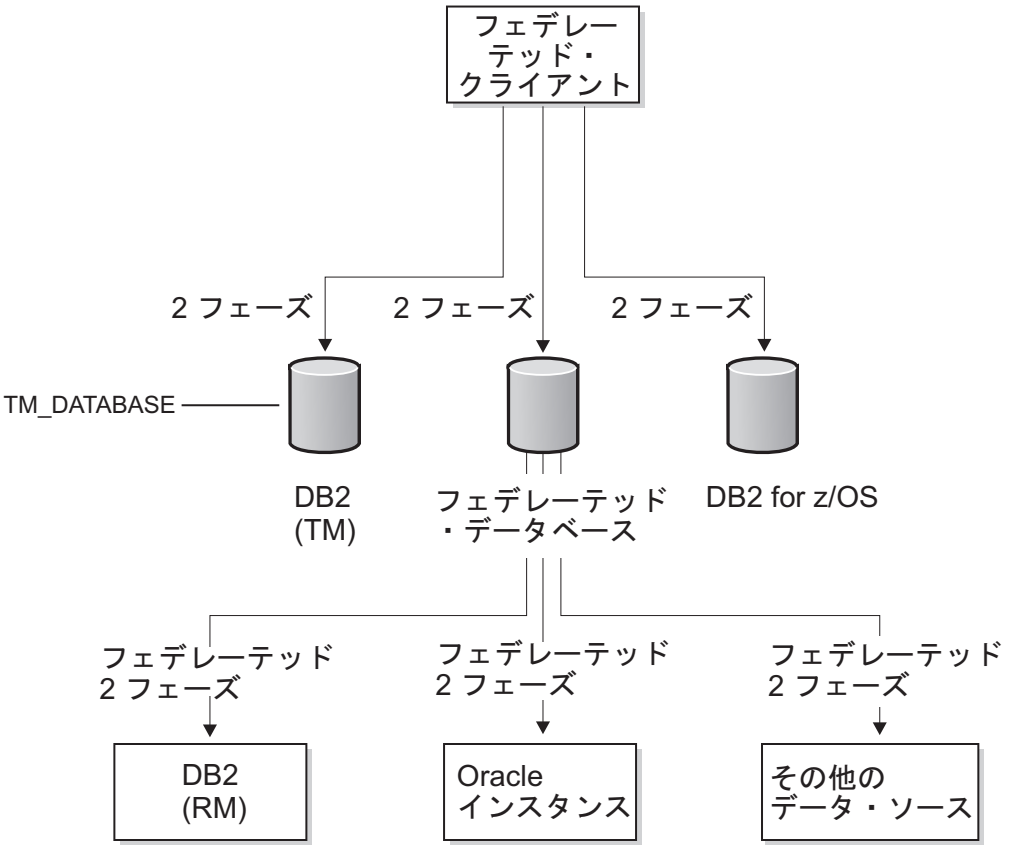


図 5. リソース・マネージャーの役目を果たすフェデレーテッド・サーバーとの DB2 タイプ 2 接続

以下の図は、トランザクション・マネージャーとして機能しているフェデレーテッド・サーバーとの DB2 タイプ 2 接続を示しています。

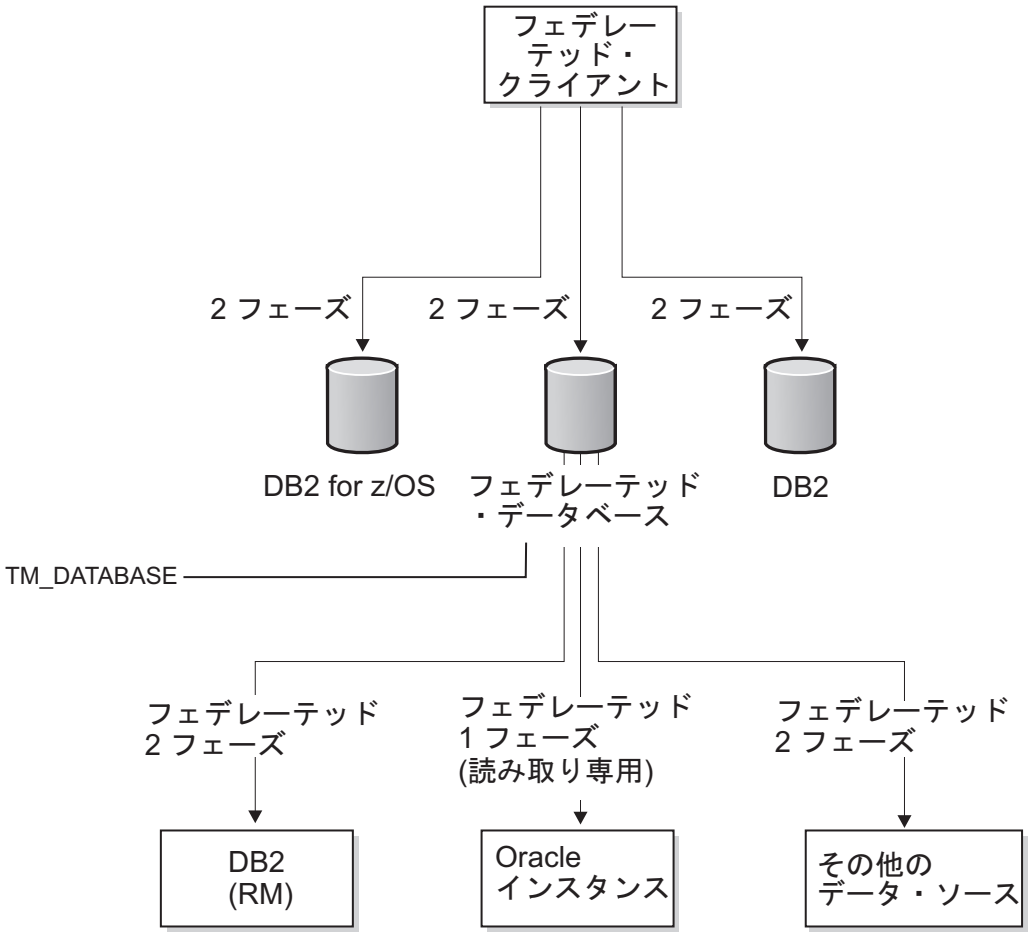


図 6. トランザクション・マネージャーの役目を果たすフェデレーテッド・サーバーとの DB2 タイプ 2 接続

以下の図は、リソース・マネージャーとして機能しているフェデレーテッド・サーバーとの XA 接続を示しています。

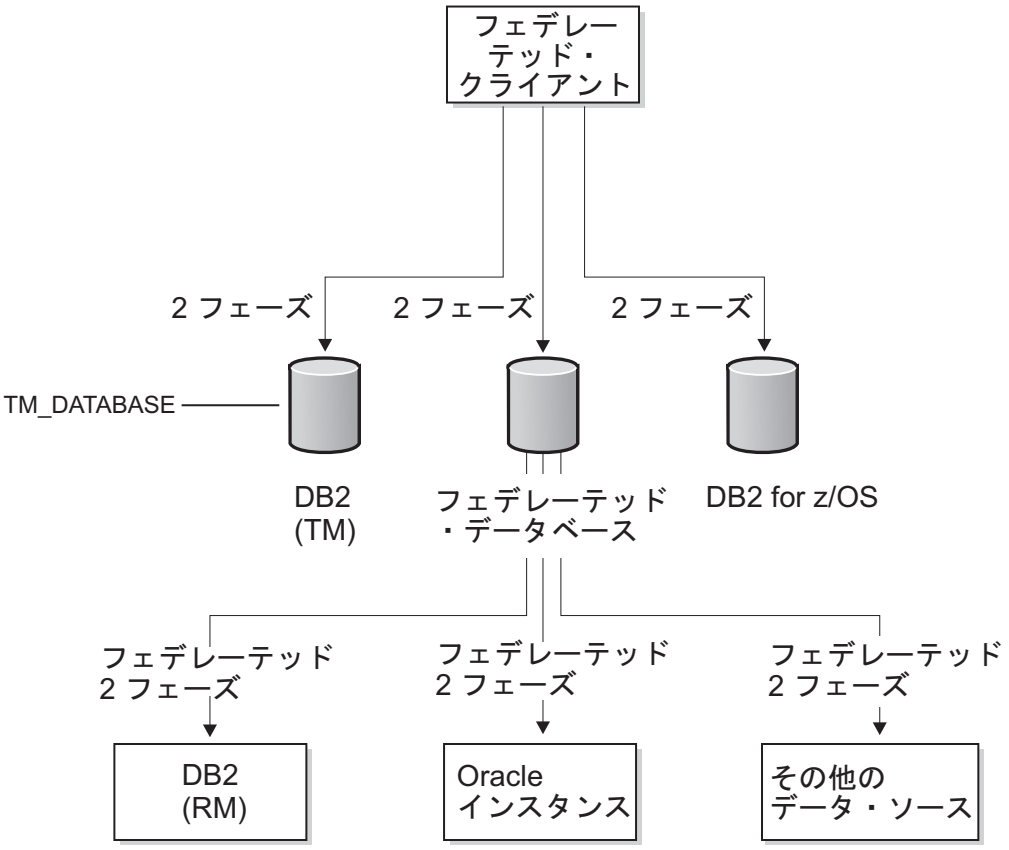


図 7. リソース・マネージャーの役目を果たすフェデレーテッド・サーバーとの XA 接続

フェデレーテッド 2 フェーズ・コミット・トランザクションの処理方法

フェデレーテッド・サーバーは、管理するデータ・ソースのデータの整合性とアトミシティを維持します。考えられるトランザクションの範囲は、接続のタイプと、フェデレーテッド・サーバーが、その接続のトランザクション・マネージャーであるか、リソース・マネージャーであるかによって異なります。

アトミシティとは、一連の操作を不可分のトランザクション内に定義するというデータベースの基本のことです。この基本に従えば、不可分のトランザクションの中の単一の操作が失敗した場合、部分的に変更を加えてデータ保全性を犠牲にするのではなく、トランザクション全体が失敗することになり、こうしてデータベースの整合性が常に保たれることになります。

例えば、資金をある口座から別の口座に送金するトランザクションには、最初の口座からの資金の引き出しと、2 番目の口座への資金の追加が関係します。引き出しだけに成功した場合、資金は本質的に最初の口座から存在しなくなります。

フェデレーテッド・サーバーは、フェデレーテッド更新要求を厳密な規則の下で処理します。フェデレーテッド更新は、以下のいずれかのアクションとなります。

- フェデレーテッド挿入、更新、または削除操作 (対応するデータ・ソースが挿入、更新、または削除操作をサポートする場合)。例えば、一部のデータ・ソース

は更新操作をサポートしません。一部のデータ・ソースでは読み取り専用であるため、フェデレーテッド挿入、更新、または削除操作は許可されません。

- パススルー・セッション内部の正常なパススルー操作。
- 透過 DDL 操作。これは、データベース更新をローカル側とリモート側の両方で実行するため、ローカル更新とフェデレーテッド更新の両方であると見なされます。
- MODIFY SQL ACCESS を使用したフェデレーテッド・ストアード・プロシージャー。

注: 単一トランザクションの中では、同じデータ・ソースにアクセスするために複数のパスを使用しないでください。そのようなトランザクションはデッドロックになる場合があります。つまり、トランザクションがハングする可能性があります。例えば、同じトランザクション内で同じデータ・ソースを参照するために複数のフェデレーテッド・サーバーを使用しないでください。

以下の表は単一の分散作業単位トランザクションで生じる事柄をリストしています。接続のタイプ、コミットのタイプ、トランザクションにおけるフェデレーテッド・サーバーの役割、許可される操作、および透過 DDL の使用方法を含みます。

表 8. 単一の分散作業単位トランザクションで発生する内容

コミットのタイプ	接続タイプ	フェデレーテッド・サーバーの役割	操作	透過 DDL
1 フェーズ	DB2 タイプ 1 または XA ローカル・トランザクション	サブトランザクション・マネージャー。さらに、トランザクション・コーディネーターとして動作し、トランザクションの結果を判別し、それを関係する各リソース・マネージャーに配信します。	1 フェーズ・コミットおよび 2 フェーズ・コミットの読み取り操作が許可されています。 1 つの 1 フェーズ・データ・ソースでは、トランザクションで更新が 1 回だけであれば更新できます。	1 フェーズ・コミット・データ・ソースの規則に従って許可および管理されます。発行される各ステートメントは、1 つの 1 フェーズ・コミット・トランザクションにおける唯一の更新でなければなりません。同じトランザクションの中で、他のフェデレーテッド 2 フェーズ・コミット・データ・ソース更新と共存することはできません。 COMMIT または ROLLBACK ステートメントは、透過 DDL トランザクションが発生する前および後に発行することを強くお勧めします。

表 8. 単一の分散作業単位トランザクションで発生する内容 (続き)

コミットのタイプ	接続タイプ	フェデレーテッド・サーバーの役割	操作	透過 DDL
2 フェーズ	DB2 タイプ 1 または XA ローカル・トランザクション	トランザクション・マネージャー。さらに、トランザクション・コーディネーターとして動作し、トランザクションの結果を判別し、それを関係する各リソース・マネージャーに配信します。	1 フェーズ・コミットおよび 2 フェーズ・コミットの読み取り操作が許可されています。複数の 2 フェーズ・データ・ソースを更新できます。	2 フェーズ・コミット・データ・ソースの規則に従って許可および管理されます。同じトランザクションで他のフェデレーテッド 2 フェーズまたは 1 フェーズ・コミット・データ・ソース更新と共存できます。
1 フェーズ	DB2 タイプ 2 または XA グローバル・トランザクション	トランザクション・マネージャーとなることができます。トランザクション・マネージャーではない場合は、外部トランザクション・コーディネーターからそれぞれの関係するリソース・マネージャーに結果を中継するに過ぎません。	1 フェーズ・コミットおよび 2 フェーズ・コミットの読み取り操作が許可されています。 1 フェーズ更新は許可されていません。ただし、分散リレーショナル・データベース体系 (Distributed Relational Database Architecture™: DRDA) 2 フェーズ・インバウンド接続を介してフェデレーテッド 1 フェーズ更新を実行可能な、DB2 で調整されるトランザクションは除きます。	1 フェーズ・コミット・データ・ソースの規則に従って許可および管理されます。発行される各ステートメントは、1 つの 1 フェーズ・コミット・トランザクションにおける唯一の更新でなければなりません。同じトランザクションの中で、他のフェデレーテッド 2 フェーズ・コミット・データ・ソース更新と共存することはできません。 COMMIT または ROLLBACK ステートメントは、透過 DDL トランザクションが発生する前および後に発行することを強くお勧めします。

表 8. 単一の分散作業単位トランザクションで発生する内容 (続き)

コミットのタイプ	接続タイプ	フェデレーテッド・サーバーの役割	操作	透過 DDL
2 フェーズ	DB2 タイプ 2 または XA グローバル・トランザクション	トランザクション・マネージャーとなることができます。トランザクション・マネージャーではない場合は、外部トランザクション・コーディネーターからそれぞれの関係するリソース・マネージャーに結果を中継するに過ぎません。	1 フェーズ・コミットおよび 2 フェーズ・コミットの読み取り操作が許可されています。複数の 2 フェーズ・データ・ソースを更新できます。	2 フェーズ・コミット・データ・ソースの規則に従って許可および管理されます。同じトランザクションで他のフェデレーテッド 2 フェーズまたは 1 フェーズ・コミット・データ・ソース更新と共存できます。

データの整合性とアトミシティを維持する方法

フェデレーテッド・サーバーは、データの整合性の確保とデータ・ソースのトランザクション・アトミシティの維持を図ろうとします。

アプリケーション同期点の設定とターゲット・データ・ソースの更新能力との間で競合が起きると、エラー (SQL30090、理由コード 18) になります。

フェデレーテッド・データベースに対して作成された DDL が含まれるローカル更新は、フェデレーテッド 1 フェーズ・データ・ソースの更新と同じトランザクション内に混在させることはできません。

DDL および透過 DDL の使用

フェデレーテッド・データベースに対して作成された DDL が含まれるローカル更新は、フェデレーテッド 1 フェーズ・データ・ソースの更新と同じトランザクション内に混在させることはできません。透過 DDL は例外です。透過 DDL では、ローカル更新とデータ・ソース更新の両方とも、接続タイプに関わらず、またデータ・ソースが 1 フェーズ・コミットまたは 2 フェーズ・コミットのどちらに構成されているかに関わらず許可されます。

透過 DDL は、リモート・データ・ソース上に表を作成し、ローカルのフェデレーテッド・データベース内にリモート表のニックネームを作成します。フェデレーテッド・サーバーは、透過 DDL トランザクションを更新として扱います。

透過 DDL には、パススルー・セッションを使用せずに、DB2 データベース・システムでリモート表を作成および変更する機能があります。透過 DDL 用の SQL ステートメントは、CREATE TABLE、ALTER TABLE、および DROP TABLE です。例えば、透過 DDL の CREATE TABLE ステートメントは、データ・ソース側にリモート表を作成し、さらにフェデレーテッド・サーバー側にその表のニックネームを作成します。そのステートメントは、ローカル更新操作とリモート更新操作を含んでいます。

一部のデータ・ソース (Oracle など) では、フェデレーテッド 2 フェーズ・コミット接続で透過 DDL を許可していません。

フェデレーテッド・トランザクションにおける 2 フェーズ・コミットの使用可能化

特定のデータ・ソースにおいてフェデレーテッド 2 フェーズ・コミットを使用するには、関連したフェデレーテッド・サーバーを使用可能にする必要があります。使用可能化プロセスには、フェデレーテッド・サーバーを準備し、データ・ソースのサーバー定義を変更することが含まれます。

始める前に

- データ・ソースでフェデレーテッド 2 フェーズ・コミットを使用可能にすると、フェデレーテッド・サーバーのデータベース・ログとデータ・ソースのデータベース・ログの両方に書き込まれるレコードの数が増えます。このことがこれらのログ・ファイルの管理および保守に与える影響を考慮し、これらの管理および保守がユーザーのローカル・ポリシーに準拠することを確認します。
- 使用するデータ・ソースは、サポートされるフェデレーテッド 2 フェーズ・コミット・データ・ソースでなければなりません。
- フェデレーテッド 2 フェーズ・コミットは、超並列処理 (MPP) 環境ではサポートされていません。
- フェデレーテッド 2 フェーズ・コミットは、次のデータ・ソースの fenced 環境でサポートされています。
  - DB2 ファミリーのデータ・ソースは、fenced モードとトラステッド・モードをサポートしています。
  - Informix は、トラステッド・モードをサポートしています。
  - Oracle は、fenced モードとトラステッド・モードをサポートしています。
  - Sybase は、fenced モードをサポートしています。
  - MS SQL Server は、トラステッド・モードをサポートしています。
- フェデレーテッド 2 フェーズ・コミットは、DB2 pureScale 環境ではサポートされていません。
- DB2 for System i、バージョン 5.3 以前、および DB2 for z/OS のデータ・ソースでは、構成パラメーター SPM_NAME がデフォルト値であるサーバー・ホスト名に設定されていることを確認します。SPM_NAME のデフォルトは、TCP/IP ホスト名の最初の 7 文字を変形したものとなります。DB2 for System i、バージョン 5.4 以降では、SPM_NAME を設定する必要はありません。

このタスクについて

このタスクについて

DB2_TWO_PHASE_COMMIT サーバー・オプションを使用すると、データ・ソースにおける 2 フェーズ・コミットが使用可能になります。データ・ソースのサーバー定義の登録は、CREATE SERVER ステートメントを使用して行います。
DB2_TWO_PHASE_COMMIT に設定した値は、そのサーバー定義の下で確立されるすべての接続において保持されます。この値は、ALTER SERVER ステートメントを使用していつでも変更できます。CREATE SERVER または ALTER SERVER ス

テートメントが正常にコミットされた後、以降のアウトバウンド接続要求において新しい設定値が使用できるようになります。

クライアントおよびアプリケーション・プログラムは SET SERVER OPTION を使用して、DB2_TWO_PHASE_COMMIT サーバー・オプションの現行値を一時的にオーバーライドすることができます。SET SERVER OPTION ステートメントは、フェデレーテッド・サーバー・データベースへの接続の直後、かつリモート・データ・ソースへの接続が確立される前に実行する必要があります。コマンドは、フェデレーテッド・データベースへ接続されている間のみ効果があります。フェデレーテッド・サーバーがリモート・データ・ソースへの接続を確立した後は、DB2_TWO_PHASE_COMMIT サーバー・オプションは変更できません。

CREATE SERVER ステートメントに XA_OPEN_STRING_OPTIONS オプションを含めると、デフォルトの XA_OPEN ストリングに特殊な情報を組み込むことができます。この組み込まれた情報は、以下のいずれかの種類の情報になります。

- IBM InfoSphere Federation Server が提供する内容に加えてトランザクションの固有 ID
- トランザクションの処理方法に関するユーザー定義のパラメーター
- XA_OPEN 要求に追加するユーザー定義のストリング

XA_OPEN 呼び出しが行われると、通常、2 フェーズ・コミットを使用するリモート・データ・ソースへの最初のトランザクションの始めに、ラッパーはユーザー定義のストリングの値を XA_OPEN 呼び出しのデフォルトの XA_OPEN ストリングに追加します。

DB2_TWO_PHASE_COMMIT および XA_OPEN_STRING_OPTIONS の両方を、CREATE SERVER、SET SERVER、または ALTER SERVER ステートメントに含めることができます。

手順

1. DB2_TWO_PHASE_COMMIT オプションを Y に設定して CREATE SERVER、ALTER SERVER、または SET SERVER ステートメントを実行します。
2. オプション: XA_OPEN_STRING_OPTIONS オプションを設定して CREATE SERVER、ALTER SERVER、または SET SERVER ステートメントを実行します。

サーバー・オプションの例

次の例は、CREATE SERVER ステートメントを使用して 2 フェーズ・コミットを設定する方法を示しています。

```
CREATE SERVER Net8_Server TYPE ORACLE VERSION 8.1.7 WRAPPER NET8
OPTIONS (DB2_TWO_PHASE_COMMIT 'Y');
```

次の例は、ALTER SERVER ステートメントを使用して 2 フェーズ・コミットを使用不可にする方法を示しています。

```
ALTER SERVER Net8_Server OPTIONS (SET DB2_TWO_PHASE_COMMIT 'N');
```

次の例は、ALTER SERVER ステートメントと XA_OPEN_STRING_OPTIONS サーバー・オプションを使用して Sybase ラッパー用の XA トレース・ファイルを D:¥Temp¥sybase_xa.log に設定する方法を示しています。

```
ALTER SERVER Ctlib_Server OPTIONS (ADD XA_OPEN_STRING_OPTIONS
'-LD:¥Temp¥sybase_xa.log');
```

次の例は、SET SERVER OPTION ステートメントを使用して 2 フェーズ・コミットを一時的に使用不可にする方法を示しています。

```
SET  SERVER  OPTION  DB2_TWO_PHASE_COMMIT TO 'N'  FOR SERVER  Net8_Server;
```

フェデレーテッド 2 フェーズ・コミット・トランザクションのデータ・ソースの要件と構成

データ・ソースでのフェデレーテッド 2 フェーズ・コミットを使用可能にする前に、それがサポートされるデータ・ソースであることを確認する必要があります。

フェデレーテッド・システムは、以下のデータ・ソースでの 2 フェーズ・コミット操作をサポートします。

- 分散リレーショナル・データベース体系 (Distributed Relational Database Architecture: DRDA) プロトコルを介した以下の DB2 ファミリー・データ・ソース
  - DB2 Universal Database for Linux, UNIX, and Windows、バージョン 8.1 またはそれ以降
  - DB2 Universal Database for z/OS、バージョン 7.1 またはそれ以降
  - DB2 Universal Database for System i、バージョン 5.3 またはそれ以降
- Informix IDS、バージョン 7.31 またはそれ以降、バージョン 9.40 またはそれ以降、バージョン 10.0 またはそれ以降
- Informix XPS、バージョン 8.40 またはそれ以降
- Microsoft SQL Server 2000 および Microsoft SQL Server 2005 (フェデレーテッド・サーバーの場合。Windows のみ)
- Oracle、バージョン 8.1.7 またはそれ以降、XA ライブラリー付き
- Sybase Adaptive Server Enterprise、バージョン 12 またはそれ以降、XA ライブラリー付き (フェデレーテッド・サーバーの場合。Windows のみ)

サポートされないデータ・ソースでフェデレーテッド 2 フェーズ・コミットを使用可能にしようとすると、SQL1881N エラーを受け取ります。

DRDA データ・ソースの構成

フェデレーテッド・サーバーは、オープン DRDA プロトコルを使用して DB2 データ・ソースへの接続を提供します。このサポートは、DB2 Connect™ サーバーにより提供されるものと同等です。

始める前に

加えて、2 フェーズ・コミットでは、フェデレーテッド・サーバーは業界標準の XA モデルを使用して各データ・ソースと対話します。

始める前に

制約事項:

- すべての DB2 データ・ソースが、DRDA を介した XA をネイティブでサポートするわけではありません。サポートしないデータ・ソース (DB2 for z/OS、DB2 for System i など) では、フェデレーテッド・サーバーは同期点マネージャー (SPM) を使用します。同期点マネージャーは、すべての DB2 サーバーがサポートする XA 2 フェーズ・コミット・フローと非 XA 2 フェーズ・コミット・フローの間のマッピングを実行します。フェデレーテッド 2 フェーズ・コミット・アクセスが同期点マネージャーを使用して提供される場合、フェデレーテッド・サポートと同期点マネージャーの間に互換性がないため、すべての XA セマンティクスがサポートされるわけではありません。例えば、トランザクションはネストできません。すべてのトランザクションは、新規トランザクションを開始する前にコミットまたはロールバックする必要があります。
- フェデレーテッド 2 フェーズ・コミットは DB2 for z/OS をサポートしますが、DB2 for z/OS はフェデレーテッド 2 フェーズ・コミット・トランザクションでの SAVEPOINT ステートメントの発行を許可しません。
- DB2 で調整されるトランザクションでは、DB2 for z/OS クライアントは DRDA 2 フェーズ・インバウンド接続を介してフェデレーテッド 1 フェーズ更新を実行できます。ただし、そのような更新を XA DRDA 2 フェーズ・インバウンド接続を介して完了することはできません。さらに、DRDA 2 フェーズ・インバウンド接続を介して行う、1 フェーズと 2 フェーズを混合した更新も完了できません。
- XA_OPEN_STRING_OPTIONS サーバー・オプションは、DRDA データ・ソースではサポートされません。このオプションを使用すると、SQL1881 エラーが返されます。

要件:

- ネイティブで XA をサポートするのではなく SPM を使用することにより XA をサポートする DB2 データ・ソースでは、データベース・マネージャー構成の SPM_NAME および SVCENAME パラメーターが適切にデフォルトに設定されていることを確認してください。

このタスクについて

手順

DRDA データ・ソースを構成するには、以下のようにします。

手順

DB2_TWO_PHASE_COMMIT オプションを Y に設定して CREATE SERVER、ALTER SERVER、または SET SERVER ステートメントを実行します。DRDA ラッパーは、以下の XA OPEN ストリングを DRDA データ・ソースのために自動的に生成します。

```
DB=dbname,UID=uid,PWD=password,TPM=FDB2,HOLD_CURSOR=T
```

Oracle データ・ソースの構成

フェデレーテッド 2 フェーズ・コミットで Oracle データ・ソースを使用するための要件および制約事項がいくつかあります。

始める前に

始める前に

制約事項:

- Oracle に送信されるパススルー DDL および透過 DDL は両方とも SQL30090 理由コード 21 (ORA-2089) で失敗します。パススルー・セッションでサブミットされる通常の SQL は作動します。

要件:

- フェデレーテッド・サーバーから 2 フェーズ・コミット・トランザクションを実行するすべてのユーザーに以下の特権を付与する必要があります。
  - grant select on dba_pending_transactions to USERID;
  - grant select on dba_2pc_pending to USERID;
  - grant force transaction to USERID;
- オプションで、フェデレーテッド・サーバーから 2 フェーズ・コミット・トランザクションを実行するユーザーに以下の特権を付与することもできます。
  - grant force any transaction to USERID;
- 10 を超える 2 フェーズ・コミット・トランザクションを同時に実行しようとする場合、Oracle サーバーの init.ora ファイル内にある distributed_transactions パラメーターを増やすことを考慮してください。

このタスクについて

手順

Oracle データ・ソースを構成するには、以下のようにします。

手順

1. DB2_TWO_PHASE_COMMIT オプションを Y に設定して CREATE SERVER、ALTER SERVER、または SET SERVER ステートメントを実行します。

   Oracle ラッパーは、以下の XA OPEN ストリングを Oracle データ・ソースのために自動的に作成します。

   ```
   Oracle_XA=Acc=P<i>uid/password</i>+SesTm=0+DB=<i>dbname</i>+SqlNet=<i>dblink</i>+Threads=true
   ```

   例:

   ```
   XA_OPEN_STRING_OPTIONS '+LogDir=/home/user/directory+DbgFl=0x7'
   ```

2. オプション: XA_OPEN_STRING_OPTIONS サーバー・オプションを使用して、追加の XA オプションを指定します。

Informix データ・ソースの構成

フェデレーテッド 2 フェーズ・コミットで Informix データ・ソースを使用するための要件および制約事項がいくつかあります。

始める前に

始める前に

制約事項:

- フェデレーテッド・サーバーへの 1 回の接続で、2 フェーズ・コミット・サーバーと 1 フェーズ・コミット・サーバーを混合して、Informix のニックネームにアクセスすることはできません。
- WITH HOLD カーソル・オプションは、サポートされていません。
- XA_OPEN_STRING_OPTIONS サーバー・オプションは、Informix データ・ソースではサポートされません。

要件:

- Informix データベースでは、ロギングを使用可能にする必要があります。
- Informix XA ライブラリーは、1 つのスレッドごとに 1 つの接続のみを許可します。その結果、フェデレーテッド・サーバーは、1 つの接続で、フェデレーテッド 2 フェーズ・コミットに使用できる複数のサーバーによって Informix データ・ソースにアクセスすることはできません。フェデレーテッド 2 フェーズ・コミットに使用できる複数のサーバーをアプリケーションが使用する必要がある場合、以下の手順でオプションのステップを完了します。

このタスクについて

手順

Informix データ・ソースを構成するには、以下のようにします。

手順

1. DB2_TWO_PHASE_COMMIT オプションを Y に設定して CREATE SERVER、ALTER SERVER、または SET SERVER ステートメントを実行します。

   Informix ラッパーは、以下の XA OPEN ストリングを Informix データ・ソースのために自動的に生成します。

   `DB=dbname;RM=rmname;CON=con;USER=user;PASSWD=password`
2. オプション: フェデレーテッド 2 フェーズ・コミットに使用できる複数のサーバーをアプリケーションが使用する必要がある場合、以下のステップを完了します。
   a. Informix ラッパー・ライブラリー `libdb2informix.a`、`libdb2informixF.a`、および `libdb2informixU.a` をコピーする。
   b. CREATE SERVER ステートメント内の LIBRARY 文節に Informix ラッパー・ライブラリーの別のコピーを指定することによって、Informix ラッパーの複数インスタンスを定義する。

c. 別々のラッパー・インスタンス用に、それぞれフェデレーテッド 2 フェーズ・コミット・サーバーを定義する。

例:

```
CREATE WRAPPER wrapper1 library 'libdb2informix.a'
CREATE SERVER server1 type informix version 9.4 wrapper wrapper1 options
  (node 'inf1', dbname 'firstdb', db2_two_phase_commit 'Y');
CREATE WRAPPER wrapper2 library 'libdb2informix2.a'
CREATE SERVER server2 type informix version 9.4 wrapper wrapper2 options
  (node 'inf2', dbname 'seconddb', db2_two_phase_commit 'Y');
```

Microsoft SQL Server データ・ソースの構成

フェデレーテッド 2 フェーズ・コミットで Microsoft SQL Server データ・ソースを使用するための要件および制約事項がいくつかあります。

始める前に

始める前に

制約事項:

- Microsoft SQL Server データ・ソースの場合、フェデレーテッド 2 フェーズ・コミットは Windows にインストールされている IBM InfoSphere Federation Server によってのみサポートされます。
- DB2 分離レベルは、Microsoft SQL server には伝搬されません。

要件:

- フェデレーテッド 2 フェーズ・コミットで Microsoft SQL を処理する場合、追加のサーバー・オプション、XA_OPEN_STRING_OPTIONS をサーバーに追加する必要があります。

  ```
  alter server S1 options(add xa_open_string_options
  'RMRecoveryGuid=c200e360-38c5-11ce-ae62-08002b2b79ef');
  ```

  ここで、RMRecoveryGuid はリソース・マネージャー ID です。

  リソース・マネージャー ID は、Microsoft SQL Server レジストリーの以下の場所を見れば分かります。

  ```
  [HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥MSSQLServer]
   "ResourceMgrID" = "{resource manager ID}"
  ```

このタスクについて

手順

Microsoft SQL Server データ・ソースを構成するには、以下のようにします。

手順

1. DB2_TWO_PHASE_COMMIT オプションを Y に設定して CREATE SERVER、ALTER SERVER、または SET SERVER ステートメントを実行します。

   Microsoft SQL Server ラッパーは、以下の XA OPEN ストリングを Microsoft SQL Server データ・ソースのために自動的に生成します。

TM=*tmname*

2. オプション: XA_OPEN_STRING_OPTIONS サーバー・オプションを使用して、必要な RMRRecovery Guid 値に加えて、追加の XA オプションを指定します。

Sybase データ・ソースの構成

フェデレーテッド 2 フェーズ・コミットで Sybase データ・ソースを使用するための要件および制約事項がいくつかあります。

始める前に

始める前に

制約事項:

- Sybase データ・ソースの場合、フェデレーテッド 2 フェーズ・コミットは、Windows にインストールされている IBM InfoSphere Federation Server によってのみサポートされます。
- Sybase に送信されるパススルー DDL および透過 DDL は両方とも SQL910N エラーで失敗します。パススルー・セッションでサブミットされる通常の SQL は作動します。

要件:

- Sybase データベース管理者は、Sybase Adaptive Server Enterprise (ASE) の分散トランザクション管理のライセンスがなければならず、isql ツールの以下のコマンドを使ってフィーチャーを使用可能にする必要があります。

  ```
  sp_configure 'enable dtm', 1
  ```

  このパラメーターを有効にするには、Sybase ASE を再始動する必要があります。
- オープン・ストリングで指定されるユーザー名は、対応する Sybase ASE 内の dtm_tm_role でなければなりません。管理ユーザーは、isql ツールの以下のコマンドを使って、この役割を割り当てることができます。

  ```
  sp_role "grant", dtm_tm_role, user_name
  ```
- Sybase Adaptive Server Enterprise (ASE) のデータ・ソースがフェデレーテッド・データベースへのリソース・マネージャーとして機能するには、$SYBASE/$SYBASE_OCS/config ディレクトリー (バージョン 12 またはそれ以降) にある xa_config ファイルに論理リソース・マネージャー (LRM) の項目がなければなりません。これは、リソース・マネージャー名を Sybase ASE 名にマップします。詳しくは、Sybase ASE XA の資料を参照してください。

  LRM 名は、XA OPEN ストリングでフェデレーテッド・サーバーによって使用されます。フェデレーテッド・サーバーは、LRM 名に Sybase ASE ノード名を使用します。
- XA 構成ファイル xa_config で指定されるサーバー名が、$SYBASE/ini ディレクトリーにある初期設定ファイル sql.ini に存在することを確認してください。

このタスクについて

手順

Sybase データ・ソースを構成するには、以下のようにします。

手順

1. Sybase XA ライブラリー・ファイル libxadtm.dll をフェデレーテッド・サーバー上にインストールします。
2. フェデレーテッド 2 フェーズ・コミット機能を使用する前に、$SYBASE/$SYBASE_OCS/config/xa_config ファイルに以下の LRM の項目を作成します。 xa_config ファイルへの書き込み許可がない場合、xa_config ファイルを別のディレクトリーに作成し、その絶対パスを db2dj.ini ファイルにある XACONFIGFILE 環境変数に設定します。

   ```
   ;one comment line is required
   lrm=lrm_name
   server=server_name
   ```

   ここで、*server_name* は、$SYBASE/ini/sql.ini ファイルにある項目名です。
3. Sybase ラッパーは、以下のデフォルトの XA_OPEN ストリングを Sybase データ・ソースのために自動的に作成します。

   ```
   -Nrmname -Uuserid -Ppassword
   ```

   XA_OPEN ストリングの他のオプションを指定する必要がある場合、XA_OPEN_STRING_OPTIONS サーバー・オプションを使用します。

フェデレーテッド 2 フェーズ・コミットの問題のリカバリー

フェデレーテッド・システムは、2 フェーズ・コミット中の問題を、未確定トランザクションの自動再同期または手動によってリカバリーすることができます。

フェデレーテッド・システムの再同期

フェデレーテッド 2 フェーズ・コミットには、コミット・トランザクション中のエラーを処理しようとする自動プロセスが含まれています。

フェデレーテッド・サーバー上のエラー以外に、フェデレーテッド環境では、ネットワーク、通信、またはデータ・ソースの障害から生じるエラーの可能性が増大します。

データ保全性を確実にするために、フェデレーテッド・サーバーは、フェデレーテッド 2 フェーズ・コミット・プロセス中にこれらのエラーを処理します。

第 1 フェーズでのエラー
作業単位のコミット準備に失敗したことをデータベースが伝える場合、フェデレーテッド・サーバーはコミット・プロセスの第 2 フェーズで作業単位をロールバックします。第 2 フェーズ中に、フェデレーテッド・サーバーはトランザクションの結果を待っているすべての参加データ・ソースに、ロールバック・メッセージを送信します。

第 2 フェーズでのエラー
このフェーズでのエラー処理は、第 2 フェーズがトランザクションをコミット

するかロールバックするかによって異なります。第 1 フェーズでエラーが検出された場合、第 2 フェーズではトランザクションのロールバックしか実行されません。

参加データ・ソースのうちの 1 つが、おそらく通信障害のせいで、作業単位のコミットまたはロールバックに失敗する場合、フェデレーテッド・サーバーは再同期と呼ばれるプロセスによってコミットまたはロールバックを再試行します。再同期は、フェデレーテッド・サーバーによって自動的に開始および管理されます。呼び出し側のアプリケーションが 2 フェーズ・コミット接続によってフェデレーテッド・サーバーに接続している場合、アプリケーションは、コミットが成功したことを SQLCA (SQL 連絡域) を通して通知されます。呼び出し側のアプリケーションが 1 フェーズ・コミット接続によってフェデレーテッド・サーバーに接続している場合、アプリケーションは、フェデレーテッド・サーバーから切断されます。

ほとんどのデータ・ソースは、フェデレーテッド・システムで障害が発生した場合に再同期を開始することができません。フェデレーテッド・サーバーが再同期プロセスを開始します。

特定の状況で、例えば電源障害のために、フェデレーテッド・サーバーがトランザクション処理中に障害を受ける場合があります。通常、再同期によってあらゆる分散作業単位トランザクションが介入なしに解決されます。

再同期では、すべての未確定トランザクションの完了が試みられます。通常の再同期の一部として、再同期エージェントはトランザクションのリソース・マネージャー・データベースに接続し、コミットまたはロールバックの決定を発行します。そして、フェデレーテッド・トランザクション・マネージャーは、分散作業単位トランザクションに参加したデータ・ソースにその決定を伝搬します。

未確定トランザクションの手動でのリカバリー

再同期が未確定トランザクションを自動的に解決するまで待てない場合、手動で未確定トランザクションを解決することができます。このプロセスは、ヒューリスティック処理と呼ばれることもあります。

例えば、外部トランザクション・マネージャーとフェデレーテッド・トランザクション・マネージャーの間の通信リンクがトランザクションの途中で失敗するとします。トランザクションについて十分な情報がある場合、フェデレーテッド・サーバーからトランザクションをロールバックすることによって、フェデレーテッド・サーバーおよびリモート・データ・ソース上でリソースを解放することができます。

トランザクション障害の理由が分かっているときにのみ、ヒューリスティック処理を使用してください。また、ロックされたリソースを即時に解放する必要があります。ほとんどの状態では、自動化再同期にトランザクションをリカバリーさせます。フェデレーテッド・システムには、トランザクション管理の複数のレイヤーがあります。ヒューリスティックなトランザクションのリカバリーは複雑であり、危険なプロセスとなり得ます。

ヒューリスティック処理を実行するための 3 つの基本的な方法があります。

- LIST INDOUBT TRANSACTIONS コマンド

このコマンド行コマンドを使用してヒューリスティック処理を実行することができます。

- 「未確定トランザクション・マネージャー」ウィンドウ

  このグラフィカル・ユーザー・インターフェース・ツールを使用してヒューリスティック処理を実行することができます。

- ヒューリスティック API

  アプリケーション内でこれらの API を使用してヒューリスティック処理を実行することができます。

ヒューリスティック処理を実行するのに使用する特定の操作およびタスクは、エラーの状況によって異なります。

フェデレーテッド・システムでは、ヒューリスティック処理要求がフェデレーテッド・トランザクション・マネージャーに送信されるとき、コミットするかロールバックするかの結果の決定は、フェデレーテッド・サーバーの未確定トランザクションの実際の状況と矛盾のないものでなければなりません。そうでなければ、エラー・メッセージが戻されます。

フェデレーテッド 2 フェーズ・コミットの未確定トランザクションの状況は、基本的な DB2 2 フェーズ・コミットの未確定トランザクションとわずかに異なります。

- (d) の状況は、トランザクションで 1 つ以上のフェデレーテッド・データ・ソースからのコミット確認通知が欠落していることを意味します。
- (b) の状況は、トランザクションで 1 つ以上のフェデレーテッド・データ・ソースからのロールバック確認通知が欠落していることを意味します。

(d) または (b) の状況で、トランザクションを正常にコミットまたはロールバックできない場合、オプション (f) を使用してトランザクションが取り消されるように指定することができます。ただし、(f) オプションを使用すると、トランザクションのすべてのレコードがフェデレーテッド・サーバーから削除されます。そして、関係するデータ・ソース上の同期の問題の残りをどれも手動でクリーンアップしなければなりません。リモート・サーバーが異常終了したか、リモート・サーバーへの接続がドロップして、緊急にリソースを解放する必要があるときなどの絶対に必要なときにのみ、(f) オプションを使用します。これは注意して使用してください。

注: (d) および (b) の状況は WebSphere® Federation Server v9.1 用の新しいものなので、下位レベルのフェデレーテッド・クライアントはそれらをサポートすることができません。未確定トランザクションを手動でリカバリーするのに下位レベルのクライアントを使用する場合、(d) および (b) の状況は代わりに状況 (m) にマップされますが、それは正確な値ではありません。未確定トランザクションを手動でリカバリーするときに不正確な情報を避けるには、v9.1 のフェデレーテッド・クライアントを必ず使用してください。デフォルトでは、 WebSphere Federated Server v9.1 を実行するコンピューターには、v9.1 のフェデレーテッド・クライアントが組み込まれています。

複数データ・ソースでの分散作業単位トランザクション状態のトレース

同期に未確定トランザクションを自動的に解決させる代わりに手動でそれらを解決することにした場合、フェデレーテッド・システムのトランザクションのトレースが不可欠です。未確定分散作業単位トランザクションをトレースする場合、失敗したデータ・ソース (複数の場合もある) を判別する唯一の方法は、障害が発生したトランザクションの XID をキャプチャーすることです。

フェデレーテッド・サーバーが分散作業単位トランザクションの一環としてアクセスした可能性があるすべてのデータ・ソースのデータ・ソース・データベース・マネージャーの中で、この XID を探さなければなりません。

トレース対象の分散作業単位に関係する各トランザクションの ID と状態を判別するには、アプリケーション・データベース、フェデレーテッド・データベース、および分散作業単位トランザクション内のあらゆるデータ・ソースで、LIST INDOUBT TRANSACTIONS コマンドを発行します。

注: 2 フェーズ・コミット・プロトコルをサポートする各データ・ソースが使用するコマンドは、それぞれ異なる場合があります。特定のデータ・ソースのコマンド資料で、XID というストリングを検索してください。

トランザクションの処理中に、フェデレーテッド・トランザクション・マネージャーは XID を 16 進形式で生成します。この XID は F2PC (数値 46325243 の 16 進形式) というフォーマット ID で始まります。フェデレーテッド・トランザクション・マネージャーは、XID をデータ・ソースに送信します。ただし、フェデレーテッド・サーバーは、XID をデータ・ソースに送信する前に、そのデータ・ソースの XID 形式に準拠するように XID を変更します。これらの変更には、XID のブランチ修飾子の長さセクションの更新、および XID のブランチ修飾子セクションの追加が含まれます。

複数のデータ・ソース内で XID を比較しなければならない場合もあるので、XID をトレースするときには、XID のどの部分を使えば環境全体に渡って正しく比較が行えるのかを知っておく必要があります。

例えば、トランザクション・マネージャーが DB2 データベースであるとします。このデータベースで LIST INDOUBT TRANSACTIONS コマンドを発行すると、トランザクション XID として、以下のものに類似した 16 進表記のストリングが戻されます。

```
463250430000019 00000000473931453346313 5 2E47453934000000 000000000000E80000
```

このストリングは、実際にはいくつかの異なる部分が合成されたものです。

フォーマット ID	トランザクション ID の長さ	ブランチ修飾子の長さ	トランザクション ID
46325043	00000019	00000000	4739314533463135 2E47453934000000 000000000000E800 00

ストリングにリストされている値は、16 進数です。例えば、トランザクション ID の長さ (16 進数の 19) は 10 進数の値 25 を表します。

この同じ XID が、フェデレーテッド・トランザクション・マネージャーとして機能するフェデレーテッド・サーバーからリソース・マネージャーとして機能するデータ・ソースに受け渡される場合、XID ストリングが付加されます。例えば、Windows データ・ソースの DB2 データベース・システム用のリソース・マネージャーに受け渡される XID は、以下のフォーマットに変更されます。

```
463252430000019 0000000147393145334631352E47453934000000  000000000000E8000001
```

変更後の同じ XID ストリングを標準的なパーツに分けたものを以下に示します。

フォーマット ID	TID の長さ	ブランチ修飾子の長さ	トランザクション ID	ブランチ修飾子
46325043	00000019	00000001	4739314533463135 2E47453934000000 000000000000E800	00

ブランチ修飾子の長さが 1 になり、ブランチ修飾子のセクションが追加されました。ただし、トランザクション ID のフィールドは変更されていません。検索ストリングの入力をトランザクション ID のセクションに制限することによって、異なるデータ・ソース内で渡って引き続き XID をトレースすることができます。

フェデレーテッド 2 フェーズ・コミットの問題のトラブルシューティング

フェデレーテッド 2 フェーズ・コミットの問題のトラブルシューティングは、多くの場合、問題を引き起こしているデータ・ソースに特有です。

アプリケーションは -911 エラー・コードの検査に加えて、トランザクションがタイムアウトになったことを示すエラー・コード (特に -913 と -918 エラー・コード) を処理する必要があります。

Oracle データ・ソースのトラブルシューティング

Oracle データ・ソースの問題をトラブルシューティングするために、以下の方法を試行します。

- 情報をログに記録するには、以下のとおりにします。

  ```
  db2 "alter server ora1 options (add XA_OPEN_STRING_OPTIONS
  '+LogDir=C:¥temp+DbgFl=0x7')
  ```

  ここで、*C:¥temp* はログ・ファイルを作成する場所への絶対パスです。情報のロギングはパフォーマンスがかなり低下するので、問題をトラブルシューティングするときにのみ、情報をログに記録します。
- トレース情報を表示するには、Oracle クライアントの sqlnet.ora ファイルに以下の行を追加します。

  ```
  TRACE_LEVEL_CLIENT=16
  TRACE_DIRECTORY_CLIENT=C:¥temp
  ```

TRACE_LEVEL_CLIENT を 4 または 8 などの低い数値に設定することもできます。トレース情報の表示はパフォーマンスがかなり低下するので、問題をトラブルシューティングするときにのみ、トレース・レベル情報を使用できるようにします。

- Oracle データ・ソースに存在する保留トランザクションをリストするには、以下のとおりにします。

```
select * from dba_pending_transactions where formatid=1177702467;
select * from dba_2pc_pending;
```

トランザクションの状態およびトランザクションの ID をリストするには、以下のとおりにします。

```
select A.STATE, A.LOCAL_TRAN_ID, A.FAIL_TIME, A.GLOBAL_TRAN_ID,
B.FORMATID || '.' || B.GLOBALID || '.' || b.BRANCHID as fmt_xid
from dba_2pc_pending A, dba_pending_transactions B
where A.GLOBAL_TRAN_ID = B.FORMATID || '.' ||
B.GLOBALID and STATE='prepared' and B.FORMATID=1177702467;
```

- トランザクションを手動で解決するには、以下のとおりにします。

```
rollback force '4.31.157818';
commit force '10.24.154537'
```

ここで、'4.31.157818' は GLOBAL_TRAN_ID の XID に対応する Oracle フィールド A.LOCAL_TRAN_ID です。

Sybase データ・ソースのトラブルシューティング

Sybase データ・ソースの問題をトラブルシューティングするために、以下の方法を試行します。

- $SYBASE ディレクトリーにある Sybase XA ログ・ファイル syb_xa_log を確認します。
- db2diag ツールを使用して、db2diag.log ファイルを検査します。
- Sybase サーバー上のトランザクションを確認するには、以下のとおりにします。

```
$ isql -Uuser_name -Ppassword -Sserver_name
1> sp_transactions
2> go
```

無効または不必要なトランザクションを見つけた場合、Sybase 管理者にトランザクションの削除を依頼してください。

フェデレーテッド 2 フェーズ・コミットのパフォーマンス

1 フェーズ・コミット・トランザクション用に構成されたデータ・ソースと比較すると、2 フェーズ・コミット・トランザクション用に構成されたデータ・ソースにはパフォーマンスの低下が見られます。

データ・ソースが 2 フェーズ・コミットを使用するとき、フェデレーテッド・サーバーは、すべての参加者が正しく同期化されていることを確認し、コーディネーターとして機能します。この調整は、フェデレーテッド・サーバーへの追加のロギングおよびデータ・ソース間の追加の通信によってなされます。このように、2 フェーズ・コミットのデータ・ソースにアクセスするフェデレーテッド・トランザクションは、1 フェーズ・コミットのデータ・ソースにアクセスするトランザクション

より多くの処理が必要です。結果として、フェデレーテッド・トランザクションが 2 フェーズ・コミットを必要とするときにのみ、データ・ソースを 2 フェーズ・コミットで使用できるようにする必要があります。

単一サイトへの更新など、1 フェーズ・コミットしか必要でないのに、2 フェーズ・コミット用のデータ・ソースで実行されるフェデレーテッド・トランザクションは、2 フェーズ・コミットのないデータ・ソースで実行される同一のトランザクションと比較して、パフォーマンスの低下が見られる傾向にあります。

2 フェーズ・コミットの制御下にあるトランザクションには、2 フェーズ・コミットを必要とするかどうかにかかわりなく、以下の追加処理が必要になります。

1. すべてのトランザクションで、フェデレーテッド・サーバーのログ・ファイルへの追加のデータベース・ログ書き込みが必要になります。

   追加のログ書き込みにより、フェデレーテッド・サーバーはトランザクションの中のデータ・ソースをトラッキングすることが可能になり、こうして後続のコミットおよびロールバック操作を調整できるようになります。
2. すべてのトランザクションで、フェデレーテッド・サーバーとデータ・ソースの間の追加のデータベース通信が必要になります。
3. 挿入、更新、および削除操作により、リモート・データ・ソースに対する 1 つ以上の追加データベース・ログ書き込みが必要になります。

追加処理のほとんどは、トランザクションの境界で起こり、トランザクションの内容によって影響されることはありません。結果として、経過時間の増加の割合は、長いトランザクションよりも短いトランザクションで大きくなります。

並行トランザクションがあると、フェデレーテッド・サーバーは 1 回の書き込み操作で複数のログ・レコードをログ・バッファーからログ・ファイルに書き込むことになります。結果として、フェデレーテッド 2 フェーズ・コミット・トランザクションを並行して実行するアプリケーションは、同じトランザクションを逐次的に実行するアプリケーションに比べてオーバーヘッドが減少します。

フェデレーテッド 2 フェーズ・コミットのパフォーマンスの向上

フェデレーテッド 2 フェーズ・コミット構成でトランザクションのパフォーマンスを向上するためのステップを実行することができます。

以下の可能な構成を検討してください。

- 追加のログ・レコードのフェデレーテッド・サーバーへの書き込みに費やす時間を削減するために、フェデレーテッド・データベース・ログ・ファイルを、高速の書き込みトランザクションが可能なデバイス (できれば書き込みキャッシュがあるデバイス) に置きます。一般に、追加の処理時間のほとんどは、フェデレーテッド・サーバー・ログの書き込みトランザクションに起因します。ログ・ファイルを正しく配置することで、2 フェーズ・コミットのパフォーマンスが向上する可能性があります。この向上は、特に読み取り専用トランザクションの場合に当てはまります。フェデレーテッド・サーバー・ログ・ファイルの位置は、NEWLOGPATH データベース構成パラメーターによって制御されます。
- 挿入、更新、および削除トランザクションの場合は、リモート・データ・ソース・ログ・ファイルを高速の書き込みが可能なメディアに置きます。

- フェデレーテッド・サーバーとデータ・ソースの間で送信される追加の XA メッセージに起因するオーバーヘッドを削減する方法は、以下のとおりです。
  - フェデレーテッド・サーバーを 2 フェーズ・コミット・データ・ソースの 1 つと同一のコンピューターに置きます。
  - フェデレーテッド・サーバーをデータ・ソースと同一のコンピューターに置くことができない場合、パフォーマンスの向上を助けるためにネットワークの速度を上げ、フェデレーテッド・サーバーとデータ・ソースの間の待ち時間を削減します。

アプリケーションについては、2 フェーズ・コミットを必要とするトランザクションがアプリケーションに少なくとも 1 つある場合のみ、データ・ソースの 2 フェーズ・コミットを有効にするようにしてください。各サーバーごとに DB2_TWO_PHASE_COMMIT のデフォルト値を N に設定し、SET SERVER OPTION ステートメントを使用して、2 フェーズ・コミットを特に必要としているアプリケーション内で 2 フェーズ・コミットを有効にします。

第 10 章 フェデレーテッド・システム内のデータの挿入、更新、および削除

フェデレーテッド環境の開発中は、リモート情報を変更するときもデータ・ソース間でデータを移動するときも、データ・ソース上のデータを挿入、更新、および削除する必要があります。

リモート・データ・ソース上でデータを変更する前に、ニックネーム上で INSERT、UPDATE、および DELETE ステートメントを発行するための適切な許可特権があることを確認してください。さらに、参照整合性、ステートメント・アトミシティの保存、および代入や割り当てのセマンティクスに関して理解していることも必要です。

INSERT、UPDATE、および DELETE ステートメントの許可特権

ニックネームに対して INSERT、UPDATE および DELETE ステートメントを発行するのに必要な特権は、表に対してこれらのステートメントを発行するための特権と似ています。さらに、基になるオブジェクトに対して選択、挿入、更新、および削除操作を実行するには、データ・ソースに対する適切な特権を持っている必要があります。

ニックネームに対する SELECT、INSERT、UPDATE、および DELETE 特権を付与または取り消すことができます。

ただし、ニックネームに対する特権を付与したり取り消したりしても、データ・ソース側で特権が付与されたり取り消されたりすることはありません。データ・ソースでは、フェデレーテッド・サーバーのユーザー・マッピングに指定された REMOTE_AUTHID に特権を付与したり取り消したりする必要があります。

ステートメントの許可 ID が持つ特権には、ニックネームに対する必要な特権 (フェデレーテッド・データベースが要求を受け入れるための) が含まれている必要があります。許可 ID に (ユーザー・マッピングを介して) マップされるデータ・ソースでのユーザー ID は、基になる表オブジェクトに対する必要な特権 (データ・ソースが要求を受け入れるための) を持っている必要があります。

フェデレーテッド・データベースに照会がサブミットされると、照会内のニックネームに対する許可特権がチェックされます。ニックネームで参照されるデータ・ソース・オブジェクトの許可要件は、照会が実際に処理される時にのみ適用されます。ニックネームに対する SELECT 特権を持っていない場合は、そのニックネームが指すデータ・ソース・オブジェクトから選択することはできません。

同様に、ニックネームに対する UPDATE 特権を持っていたとしても、そのニックネームが指すデータ・ソース・オブジェクトの更新を自動的に許可されたことにはなりません。フェデレーテッド・サーバーでの特権のチェックをパスしても、リモート・データ・ソース側での特権のチェックをパスするとはかぎりません。ユーザ

ー・マッピングを使用して、フェデレーテッド・サーバーの許可 ID はデータ・ソースのユーザー ID と対応付けられます。特権のチェックをデータ・ソース側で強制します。

フェデレーテッド・システムでの INSERT、UPDATE、および DELETE の制約事項

フェデレーテッド・システムでの INSERT、UPDATE、または DELETE ステートメントの使用には、いくつかの制約事項が適用されます。

ニックネームに対する更新には、以下の制約事項が適用されます。

- 読み取り専用のデータ・ソース・オブジェクト (JOIN ビューなど) は更新できない
- UNION ALL ステートメントで作成されたフェデレーテッド・ビューに対して挿入、更新、および削除操作を実行することはできない。 UNION ALL ステートメントで作成されるフェデレーテッド・ビューは、読み取り専用ビューです。
- フェデレーテッド挿入、更新、削除操作、または SQL データ・アクセス標識 MODIFIES SQL DATA を指定したフェデレーテッド・プロシージャーへの呼び出しは、以下のいずれかが該当する関数、データ変更表参照、ATOMIC を指定する複合ステートメント (DB2 for Linux, UNIX, and Windows データ・ソースを除く)、トリガー、およびアプリケーション実行環境では無効です。
  - SAVEPOINT が有効 (DB2 for Linux, UNIX, and Windows データ・ソースを除く)
  - 両方向スクロール・カーソルが使用されている。
  - ターゲット・ビューに複数の表やニックネームが含まれる。

サポートされないデータ・ソース

フェデレーテッド・システムは、非リレーショナル・データ・ソースのニックネームに対する挿入、更新、および削除操作はサポートしません。

フェデレーションがサポートしないデータ・ソースには、以下が含まれます。

- BioRS
- Excel
- 表構造ファイル
- Web サービス
- XML

フェデレーテッド・システムの参照整合性

フェデレーテッド・システムでは、フェデレーテッド・データベースはデータ・ソース間の参照整合性を強制しません。

ただし、データ・ソース側の参照整合性制約が、ニックネームの更新に影響を与えることはあります。フェデレーテッド・サーバーにあるデータをニックネームに挿入する必要があると仮定します。フェデレーテッド・サーバーが挿入をデータ・ソ

ースに対して送信すると、データ・ソース側の参照整合性制約に違反します。この場合、フェデレーテッド・サーバーは結果のエラーをフェデレーテッド・エラーと対応付けます。

データ・ソース間の参照整合性を担当するのはアプリケーションです。

INSERT、UPDATE、および DELETE ステートメントとラージ・オブジェクト (LOB)

フェデレーテッド・システムのリモート LOB に対して読み取り操作を実行できます。 LOB に対する書き込み操作は、一部のデータ・ソースではサポートされています。

フェデレーションを使用することにより、すべてのリレーショナル・データ・ソースにある LOB に対して読み取り操作を実行することができます。書き込み操作は、以下のデータ・ソースにある LOB に対して実行することができます。

- Oracle。NET8 ラッパーを使用
- DB2 for z/OS、DB2 for System i、および DB2 Database for Linux, UNIX, and Windows。DRDA ラッパーを使用
- Teradata。Teradata ラッパーを使用

特定の条件では、ニックネーム列のタイプを VARCHAR に変更することにより、他のデータ・ソースの LOB に対する書き込み操作を実行することができます。

フェデレーテッド・システムでのステートメント・アトミシティの保持

更新操作中、フェデレーテッド・システムは DML ステートメントの完了時に常にデータをアトミック状態に保とうとします。データがアトミック状態であれば、データを正常に処理したり未変更のままにしておくことができます。

クライアントまたはアプリケーションが、ニックネームに対して INSERT、UPDATE、または DELETE ステートメントを発行すると、フェデレーテッド・サーバーはそのステートメントを、単一の DML ステートメントまたは一連の DML ステートメントとして内部処理します。フェデレーテッド・サーバーが複数の DML ステートメントを処理のためにターゲット・データ・ソースに送信する必要がある場合は、データ・アトミシティが失われる可能性があります。データ・アトミシティが失われることを避けるために、フェデレーテッド・システムはデータ・ソース・セーブポイント API を使用して、一連の DML ステートメントをモニターします。

一部のデータ・ソースは、フェデレーテッド・システムからアクセスできるセーブポイント API を外部に公開していません。このような状況では、フェデレーテッド INSERT、UPDATE、または DELETE ステートメントは、セーブポイント API の保護を受けずに実行されます。

つまり、このようなデータ・ソースでは、フェデレーテッドの挿入、更新、削除トランザクション中にエラーが発生すると、部分的な更新が行われてしまう可能性が

あります。不整合の問題を正すために、フェデレーテッド・システムは、SQLCODE エラーをアプリケーションに戻す前に、自動的に内部トランザクション・ロールバックを実行します。

以下のデータ・ソースは、フェデレーテッド・サーバーが使用できるセーブポイント API を外部に公開していません。

- DB2 for System i
- DB2 for VM and VSE
- Informix
- JDBC
- Microsoft SQL Server
- ODBC
- Teradata

挿入、更新、または削除トランザクション全体が処理のためにデータ・ソースにプッシュダウンされる場合は、フェデレーテッド・サーバーは、エラーが発生してもデータ・ソースでステートメント・アトミシティが保持されるとみなします。挿入、更新、または削除トランザクションの一部だけが処理のためにデータ・ソースにプッシュダウンされる場合は、エラーが発生するとトランザクション全体がロールバックされます。

フェデレーテッド・システム内のデータの変更

データ・ソース・オブジェクト中のデータの挿入、更新、および削除を行うと、フェデレーテッド・システム中のデータを変更できます。

データ・ソース・オブジェクトへのデータの挿入

データ・ソースにデータを挿入するには、INSERT ステートメントでデータ・ソース・オブジェクトのニックネームを使用します。

始める前に

ニックネームを使用してデータを挿入するには、以下のすべての特権が必要です。

- ステートメントの許可 ID が持つ特権には、ニックネームに対する INSERT 特権 (フェデレーテッド・データベースが要求を受け入れるための) が含まれている必要があります。
- データ・ソース側のユーザー ID は、基になる表オブジェクトに対する INSERT 特権 (データ・ソースが要求を受け入れるための) を持っている必要があります。
- データ・ソース側のユーザー ID は、ユーザー・マッピングを介してフェデレーテッド・サーバー側の許可 ID にマップする必要があります。

このタスクについて

制約事項

フェデレーションは、非リレーショナル・データ・ソースが関係する INSERT 操作をサポートしていません。

手順

データをデータ・ソース・オブジェクトに挿入するには、INSERT ステートメントを発行します。

例

Informix 表が 2 つの列から構成されているとします。 1 番目の列には、INTEGER データが、2 番目の列には (最大 20 桁の) VARCHAR データが入っています。ニックネーム *infx_table_nn* は、 Informix 表用にフェデレーテッド・サーバーに登録されています。

infx_table_nn ニックネームを使用して、Informix 表に対して INSERT、UPDATE、および DELETE ステートメントを発行することができます。以下のステートメントは、Informix 表に新しい情報を 1 行挿入します。

```
INSERT INTO db2user1.infx_table_nn VALUES(1,'Walter')
```

データ・ソース・オブジェクト内のデータの更新

データ・ソースのデータを更新するには、UPDATE ステートメントでデータ・ソース・オブジェクトのニックネームを使用します。

始める前に

ニックネームを使用してデータを更新するには、以下のすべての特権が必要です。

- ステートメントの許可 ID が持つ特権には、ニックネームに対する UPDATE 特権 (フェデレーテッド・データベースが要求を受け入れるための) が含まれている必要があります。
- データ・ソース側のユーザー ID は、基になる表オブジェクトに対する UPDATE 特権 (データ・ソースが要求を受け入れるための) を持っている必要があります。
- データ・ソース側のユーザー ID は、ユーザー・マッピングを介してフェデレーテッド・サーバー側の許可 ID にマップする必要があります。

このタスクについて

制約事項

フェデレーションは、一部のデータ・ソースが関係する UPDATE 操作をサポートしていません。 136 ページの『フェデレーテッド・システムでの INSERT、UPDATE、および DELETE の制約事項』を参照してください。

データ・ソース・オブジェクト内でデータを更新するには、UPDATE ステートメントを発行します。

例: Informix 表が 2 つの列から構成されているとします。 1 番目の列には、INTEGER データが、2 番目の列には (最大 20 桁の) VARCHAR データが入っています。ニックネーム *infx_table_nn* は、 Informix 表用にフェデレーテッド・サーバーに登録されています。

infx_table_nn ニックネームを使用して、Informix 表に対して INSERT、UPDATE、および DELETE ステートメントを発行することができます。以下のステートメントは、Informix 表の 1 行の情報を更新します。

```
UPDATE db2user1.infx_table_nn SET c2='Bill' WHERE c1=2
```

データ・ソース・オブジェクトからのデータの削除

データ・ソースからデータを削除するには、DELETE ステートメントでデータ・ソース・オブジェクトのニックネームを使用します。

始める前に

ニックネームを使用してデータを削除するには、以下のすべての特権が必要です。

- ステートメントの許可 ID が持つ特権には、ニックネームに対する DELETE 特権 (フェデレーテッド・データベースが要求を受け入れるための) が含まれている必要があります。
- データ・ソース側のユーザー ID は、基になる表オブジェクトに対する DELETE 特権 (データ・ソースが要求を受け入れるための) を持っている必要があります。
- データ・ソース側のユーザー ID は、ユーザー・マッピングを介してフェデレーテッド・サーバー側の許可 ID にマップする必要があります。

このタスクについて

制約事項

フェデレーションは、一部のデータ・ソースが関係する DELETE 操作をサポートしていません。 136 ページの『フェデレーテッド・システムでの INSERT、UPDATE、および DELETE の制約事項』を参照してください。

手順

データをデータ・ソース・オブジェクトから削除するには、DELETE ステートメントを発行します。

例

Informix 表が 2 つの列から構成されているとします。 1 番目の列には、INTEGER データが、2 番目の列には (最大 20 桁の) VARCHAR データが入っています。ニックネーム *infx_table_nn* は、 Informix 表用にフェデレーテッド・サーバーに登録されています。

infx_table_nn ニックネームを使用して、Informix 表に対して INSERT、UPDATE、および DELETE ステートメントを発行することができます。以下のステートメントは、Informix 表の 1 行の情報を削除します。

```
DELETE FROM infx_table_nn WHERE c1=3
```

フェデレーテッド・システムでの割り当てのセマンティクス

データをニックネーム列に割り当てる際に、フェデレーテッド・システムが使用する割り当て規則に基づいて、データ・タイプが変わる場合があります。予想通りの結果を得られるように、割り当て規則について理解する必要があります。

ニックネーム列への割り当てのターゲット・データ・タイプを判別する際の規則を、以下に示します。

- ローカル・ソース・タイプの判別: ローカル・ソース・タイプは、ローカル列タイプまたは式のローカル結果タイプによって判別されます。ソースが定数の場合、ローカル・ソース・タイプは定数のタイプと同じです。
- ターゲット・タイプの判別:
  - パラメーター・マーカーや NULL など、割り当てのソースがない場合、ターゲット・タイプは MIN(local_target_type, remote_target_type) になります。この local_target_type は更新される列のローカル・データ・タイプ、remote_target_type は更新される列のデータ・ソースのデータ・タイプです。remote_target_type は、リモート・ターゲット列のデータ・タイプに該当するデフォルトの順方向タイプ・マッピングのタイプから導きます。
  - 割り当てのソースが NULL またはパラメーター・マーカーではない場合、ターゲット・タイプは MIN(local_target_type, remote_target_type, local_source_type) です。

MIN(type1, type2) の定義

- type1 と type2 は全く同じという訳ではありません。
- MIN(type1,type2) = MIN(type2, type1)
- MIN(type1, type2) = DECIMAL(0,0) の場合、 MIN(type1, type2) = remote_target_type(local_target_type)。
- BLOB が互換性があるのは BLOB だけであるため、MIN(BLOB(x), BLOB(y))=BLOB(z) です。ここで、z=min(x,y) です。
- TIME データ・タイプと DATE データ・タイプの間には互換性はありません。
- 日時タイプと文字ストリングの間には互換性があります。
- Unicode データベースでは、文字ストリングと GRAPHIC ストリングの間に互換性があります。

下記の表では、数値、文字ストリング、GRAPHIC ストリング、および日時のデータ・タイプでの、 2 つのデータ・タイプの最小値をリストしています。

表 9. 数値データ・タイプ

type1	type2	MIN(type1, type2)
SMALLINT	SMALLINT、 INTEGER、 BIGINT、 REAL、または DOUBLE	SMALLINT
INTEGER	BIGINT、 REAL、または DOUBLE	INTEGER
BIGINT	REAL または DOUBLE	BIGINT
REAL	DOUBLE	REAL
DECIMAL(w,x)	SMALLINT	DECIMAL(p,0)。ここで、p<5 の場合は p=w-x です。それ以外の場合は SMALLINT です。

表 9. 数値データ・タイプ (続き)

type1	type2	MIN(type1, type2)
DECIMAL(w,x)	INTEGER	DECIMAL(p,0)。ここで、p<11 の場合は p=w-x です。それ以外の場合は INTEGER です。
DECIMAL(w,x)	BIGINT	DECIMAL(p,0)。ここで、p<19 の場合は p=w-x です。それ以外の場合は BIGINT です。
DECIMAL(w,x)	DECIMAL(y,z)	DECIMAL(p,s)。ここで、p=min(w,y)+min(w-x,y-z)、s=min(x,z) です。
DECIMAL(w,x)	DOUBLE または REAL	DECIMAL(w,x)

下記の表は、文字ストリング・データ・タイプでの、2 つのデータ・タイプの最小値をリストしています。

表 10. 文字ストリング・データ・タイプ

type1	type2	MIN(type1, type2)
CHAR(x)	CHAR(y)、 VARCHAR(y)、LONG VARCHAR、または CLOB(y)	CHAR(z)。ここで、z=min(x,y) です。
VARCHAR(x)	VARCHAR(y)、 LONG VARCHAR、または CLOB(y)	VARCHAR(z)。ここで、z=min(x,y) です。
LONG VARCHAR	CLOB(y)	x>32700 の場合は LONG VARCHAR、x<=32700 の場合は CLOB(x) です。
CLOB(x)	CLOB(y)	CLOB(z)。ここで、z=min(x,y) です。

下記の表は、GRAPHIC ストリング・データ・タイプでの、 2 つのデータ・タイプの最小値をリストしています。

表 11. GRAPHIC ストリング・データ・タイプ

type1	type2	MIN(type1, type2)
GRAPHIC(x)	GRAPHIC(y)、VARGRAPHIC(y)、 LONG VARGRAPHIC、または DBCLOB(y)	GRAPHIC(z)。ここで z=min(x,y) です。
VARGRAPHIC(x)	VARGRAPHIC(y)、LONG VARGRAPHIC、または DBCLOB(y)	VARGRAPHIC(z)。ここで、z=min(x,y) です。
LONG VARGRAPHIC	DBCLOB(y)	x>32700 の場合は LONG VARGRAPHIC、x<=32700 の場合は DBCLOB(x) です。
DBCLOB(x)	DBCLOB(y)	DBCLOB(z)。ここで、z=min(x,y) です。

下記の表は、日時のデータ・タイプでの、2 つのデータ・タイプの最小値をリストしています。

表 12. 日時のデータ・タイプ

type1	type2	MIN(type1, type2)
DATE	TIMESTAMP	DATE
TIME	TIMESTAMP	TIME

挿入するデータ・タイプ CHAR のデータがターゲットの長さよりも短い場合、データ・ソースが列の残りを埋め込みます。

DATE または TIME データ・タイプのデータを TIMESTAMP データ・タイプのリモート列へ挿入する場合、データ・ソースが列の残りを埋め込みます。

フェデレーテッド・システムでの割り当てのセマンティクス - 例

以下の表には、ローカル・タイプおよびリモート・タイプを指定した照会での、フェデレーテッド割り当てのセマンティクスの適用に関するいくつかの例が示されています。

表 13. 割り当てのセマンティクスの例

ローカル・タイプ	リモート・タイプ	行う照会	生成されるリモート照会
FLOAT	INTEGER	set c1=123.23	set c1=INTEGER(123.23)
INTEGER	FLOAT	set c1=123.23	set c1=INTEGER(123.23)
FLOAT	INTEGER	set c1=123	set c1=123
CHAR(10)	CHAR(20)	set c1='123'	set c1='123' ('123' はタイプ VARCHAR(3) であり、すべての中で最も短い)
CHAR(10)	CHAR(20)	set c1=char23col	set c1=CHAR(char23col, 10)
CHAR(10)	CHAR(20)	set c1=expr1	• set c1=expr1 (expr1 が char(n) を戻し、n<= 10 の場合) • set c1=CHAR(expr1, 10) (expr1 が char(n) を戻し、n>10 の場合)
TIMESTAMP	DATE	set c1= 現在のタイム・スタンプ	set c1=DATE(現在のタイム・スタンプ)

データ・タイプ値に関するデータ・ソースの制約事項

一部のデータ・タイプの場合、フェデレーテッド・サーバーがサポートする値の範囲がデータ・ソースより広くなります。

これらのデータ・タイプを使用する際には、データ・ソースがサポートする値のみ使用してください。サポートの範囲外の値を挿入すると、データ・ソースは値を変換するかエラーを戻します。

サポート範囲外の値が含まれている述部を使用する場合、データ・ソースでこの述部が評価されると、以下のいずれかの結果になることがあります。

- この述部が戻す結果が、フェデレーテッド・サーバー上の同じ述部が戻す結果と違うことがあります。
- この述部が結果としてエラーになる場合があります。

時間フィールド内の 24 時のある時刻とタイム・スタンプ

時間フィールド内に 24 時を含む時刻またはタイム・スタンプがある述部を使用する場合は、エラーを受け取ることがあります。問題を避けるには、これらの時刻またはタイム・スタンプを変換できます。以下の UPDATE ステートメントのようなステートメントを使用できます。

```
UPDATE my_table SET timestamp_col = timestamp_col + 0 SECONDS;
```

この update ステートメントは、時刻フィールドを 00:00:00 に変更し、日付を 1 日ずつ増やしていきます。

フェデレーテッド・サーバーを使用すると、時間フィールド内の時刻やタイム・スタンプに 24 時を含めることができます。 24 時台のタイム・スタンプと 00 時台のタイム・スタンプは、比較においては違うものとして扱われますが、算術的には等しいものとして扱われます。

例

- 以下の述部は false を戻します。

  ```
  2008-05-14-24:00:00.000000 = 2008-05-15-00:00:00.000000
  ```

- 以下の述部は true を戻します。

  ```
  (2008-05-14-24:00:00.000000 + 0 SECONDS) =
   (2008-05-15-00:00:00.000000 + 0 SECONDS)
  ```

Oracle や Sybase などの一部のデータ・ソースでは、時間フィールドで 24 時のある時刻やタイム・スタンプを使用できません。

Oracle での空ストリング

Oracle ニックネームを使用するアプリケーションで空ストリングを使用しなければ、空ストリングの問題を避けることができます。代わりに、NULL 値か 1 つのブランク (' ') から成るストリングを使用できます。

VARCHAR2 互換ではないフェデレーテッド・サーバーの VARCHAR 列では、空ストリングと NULL 値が区別されます。その結果、ローカル表とニックネームで振る舞いに相違が生じます。一方、VARCHAR2 互換のリモート・データ・ソースの VARCHAR 列では、空ストリングと NULL 値は同じであると見なされます。VARCHAR 列に挿入された空ストリング ('') は、NULL 値に変換されます。

以下の例で、MY_TABLE はフェデレーテッド・データベース内のローカル表であり、MY_NICK は Oracle データベース内のリモート表のニックネームです。

例 1 この例では、いずれも NOT_NULL_COL という名前の NOT NULL 列を含む、MY_TABLE および MY_NICK に空ストリングが挿入されます。MY_TABLE に対する INSERT ステートメントは成功しますが、MY_NICK に対する INSERT はエラーが発生して失敗します。これは、空ストリングが NULL 値に変換されることで、NOT NULL 制約への矛盾が生じるためです。

```
INSERT INTO my_table(not_null_col) VALUES('');
INSERT INTO my_nick(not_null_col) VALUES('');
```

例 2 この例では、フェデレーテッド表とリモート表はいずれも最初は空の状態です。フェデレーテッド・サーバーでは空ストリングと NULL 値を区別するため、最初の SELECT ステートメントは行を戻しません。しかし、Oracle データベースは挿入中に空ストリングを NULL 値に変換するので、2 番目の SELECT ステートメントは 1 行を戻します。

```
INSERT INTO my_table(my_col) VALUES('');
SELECT * FROM my_table WHERE my_col IS NULL;

INSERT INTO my_nick(my_col) VALUES('');
SELECT * FROM my_nick WHERE my_col IS NULL;
```

アプリケーション・セーブポイントによるフェデレーテッド更新

フェデレーテッド・システムのアプリケーション・セーブポイントは、トランザクション内の一連の SQL ステートメントのサブセットが実行する作業を制御するのに役立ちます。

単一トランザクション内で、複数のセーブポイントを使用できます。セーブポイントの下でグループ化された SQL ステートメントのサブセットは、1 つのユニットとして扱えます。トランザクションを、セーブポイント・ユニットの単一レベルまたはネストされたレベルに、論理的に分割できます。

アプリケーション内でセーブポイントを設定後、その設定を解除することも、セーブポイントの設定以降に実行された作業をロールバックすることもできます。例えば、セーブポイント内の個々の SQL ステートメントが失敗しても、全体としてのユニットは失われません。以下のいずれかを行うことができます。

- セーブポイントまでロールバックして、そのセーブポイント内で実行したすべての SQL ステートメントを元に戻すことができます。
- システムでセーブポイントを維持する必要がなくなった場合は、セーブポイントを解除します。

アプリケーション・セーブポイントを伴うフェデレーテッド更新操作は、 DB2 Database for Linux, UNIX, and Windows で実行できます。

アプリケーション・セーブポイントによるフェデレーテッド更新 - 例

このアプリケーション・セーブポイントの例では、ネストされているセーブポイント内でのロールバック操作の効果について説明します。

例: サーバー DB2_SERVER で、スキーマ DEPTDATA 内の表 DEPARTMENT に NN_DEPT というニックネームを作成します。ニックネーム NN_DEPT には、列 DEPT_NO、DEPT_NAME、および MANAGER_NO が含まれます。

```
CREATE NICKNAME NN_DEPT FOR DB2_SERVER.DEPTDATA.DEPARTMENT;
```

セーブポイント SP1 を作成する前に、1 行挿入します。セーブポイント SP2 を作成する前に、もう 1 行挿入します。セーブポイント SP3 を作成する前に、さらに 2 行挿入します。 5 行目を挿入します。

```
INSERT INTO NN_DEPT VALUES ('A10', 'SALES', 'ADAM');
SAVEPOINT SP1 ON ROLLBACK RETAIN CURSORS;

INSERT INTO NN_DEPT VALUES ('B10', 'MARKETING', 'BRIAN');
SAVEPOINT SP2 ON ROLLBACK RETAIN CURSORS;

INSERT INTO NN_DEPT VALUES ('C10', 'ENGINEERING', 'CINDY');
INSERT INTO NN_DEPT VALUES ('C20', 'TESTING', 'DOUG');
SAVEPOINT SP3 ON ROLLBACK RETAIN CURSORS;

INSERT INTO NN_DEPT VALUES ('D10', 'SUPPORT', 'EMILY');
```

この時点で、NN_DEPT ニックネームには、A10、B10、C10、C20、および D10 の各部門を含む 5 つの行が含まれています。これらの行の一部をロールバックするとします。以下の例では、各セーブポイントに対するロールバックの結果について説明します。

- `ROLLBACK TO SAVEPOINT SP3;`

  部門 D10 を含む最後の行は、NN_DEPT ニックネーム内に存在しなくなります。セーブポイント SP3 の作成前に挿入した行 (A10、B10、C10、C20) は、依然として NN_DEPT ニックネームに存在しています。
- `ROLLBACK TO SAVEPOINT SP2;`

  部門 C10、C20、D10 を含む行は、NN_DEPT ニックネームに存在しなくなります。セーブポイント SP2 の作成前に挿入した行 (A10、B10) は、依然として NN_DEPT ニックネームに存在しています。
- `ROLLBACK TO SAVEPOINT SP1;`

  部門 B10、C10、C20、D10 を含む行は、NN_DEPT ニックネームに存在しなくなります。セーブポイント SP1 の作成前に挿入した行 (A10) は、依然として NN_DEPT ニックネームに存在しています。

アプリケーション・セーブポイントを伴うフェデレーテッド更新の制限

フェデレーテッド・システムでは、セーブポイント操作にいくつかの制限があります。

フェデレーテッド・アプリケーション内のリモート・データに対する書き込み操作へのセーブポイントのサポートは、 DB2 Database for Linux, UNIX, and Windows に限られます。以下の制限は、ニックネームが (書き込み操作のターゲットとして) 定義されるデータ・ソース・オブジェクトに適用されます。

- バージョン 9.5 では、セーブポイント操作のデータ・ソース・オブジェクトを、ニックネームそのものにすることが可能です。
- それ以前のバージョンでは、セーブポイント操作のデータ・ソース・オブジェクトをニックネームにすることはできません。

セーブポイントは、シリアル・モード、超並列処理 (MPP) 環境、およびフェデレーテッド 1 フェーズ・コミット環境またはフェデレーテッド 2 フェーズ・コミット環境で活動化できます。これらの環境における制限も、セーブポイント操作に適用されます。

第 11 章 ニックネームのデータのインポートとエクスポート

IMPORT コマンドを使用してニックネームにデータをインポートすることができます。また、EXPORT コマンドを使用して、ニックネームを参照する照会からデータをエクスポートすることができます。

IMPORT コマンドは、以下のデータ・ソースでのみ使用できます。

- DB2 ファミリー
- Informix
- Microsoft SQL Server
- Oracle
- Sybase
- Teradata

ニックネームにデータをインポートする場合の制約事項

IMPORT コマンドを使用してデータをニックネームにインポートする場合には制約事項があります。

IMPORT コマンドを使用してデータをニックネームにインポートするときには、以下の制約事項が適用されます。

- ニックネームが定義されているリモート・オブジェクトは表でなければなりません。ビューまたはシノニムに定義されているニックネームにデータをインポートすることはできません。
- サポートされるファイル・タイプは、IXF、ASC、および DEL です。
- ALLOW WRITE ACCESS 文節を指定する必要があります。この文節は、オンライン・インポート・モードを起動します。ALLOW WRITE ACCESS 文節は、同時アプリケーションにインポート・ターゲット表への読み取りおよび書き込み両方のアクセスを許可します。
- ニックネームでは、COMMITCOUNT AUTOMATIC モードを使用することができません。
- COMMITCOUNT *n* を指定する必要があります。ここで、*n* はゼロ以外の有効な数値です。
- ニックネームでは INSERT および INSERT_UPDATE 操作のみがサポートされます。
- ニックネームでサポートされない列タイプは、LOB および生成列です。 LOB データをリモート表にインポートするには、対応するニックネーム列が VARCHAR データ・タイプでなければなりません。
- 以下の `filetype` 修飾子は、ニックネームではサポートされていません。

```
dldelfiletype
generatedignore
generatedmissing
identityignore
identitymissing
```

```
indexixf
indexschema
lobsinfile
nodefaults
no_type_idfiletype
usedefaults
```

- 階層 (型付き表) はニックネームではサポートされていません。

これらの制約事項に従わない IMPORT コマンドをサブミットすると、SQL エラー・コード 27999N が戻されます。例:

```
SQL27999N リモート・ターゲット (ニックネーム) への要求された IMPORT 操作を実行できません。
理由コード = "reason_code"
```

ニックネームでの IMPORT コマンド - 例

この例は、異なるファイル・タイプからニックネームにデータをインポートする方法を示しています。

DEL ファイル・タイプ

この例では、INSERT_UPDATE オプションを使用してデータを DEL ファイル・タイプからインポートします。

```
IMPORT FROM import_file_1.del OF DEL
   ALLOW WRITE ACCESS
   COMMITCOUNT 50
   INSERT_UPDATE INTO NICKNAME_1;
```

IXF ファイル・タイプ

この例では、INSERT オプションを使用してデータを IXF ファイル・タイプからインポートします。

```
IMPORT FROM import_file_1.ixf OF IXF
   ALLOW WRITE ACCESS
   COMMITCOUNT 20
   INSERT INTO NICKNAME_1;
```

ASC ファイル・タイプ

この例では、INSERT オプションを使用してデータを ASC ファイル・タイプからインポートします。例には、データ内のすべての末尾ブランク・スペースを切り捨てるための STRIPTBLANKS ファイル修飾子が含まれています。 METHOD L パラメーターは列番号の先頭と末尾を指定します。

```
IMPORT FROM import_file_1.asc OF ASC MODIFIED BY STRIPTBLANKS
   METHOD L(1 6, 8 32, 34 44, 46 48)
   ALLOW WRITE ACCESS
   COMMITCOUNT 20
   INSERT INTO NICKNAME_1;
```

ニックネームを使用してデータをエクスポートする際の制約事項

EXPORT コマンドを使用して、ニックネームを参照する照会からデータをエクスポートする場合には、制約事項があります。

EXPORT コマンドでニックネームを使用してデータをエクスポートするときには、以下の制約事項が適用されます。

- import CREATE 操作を実行するために必要なターゲット表の説明は、IXF ファイル・フォーマットでは保管されません。表の再作成に必要な情報を収集するためには、db2look ユーティリティーを使用してください。
- データは IXF および DEL ファイル・タイプにエクスポートできます。ニックネームからのデータのエクスポートでは、ASC ファイル・タイプはサポートされていません。

第 12 章 ニックネームを使用した操作

ニックネームとは、アプリケーションが表やビューなどのデータ・ソース・オブジェクトを参照するのに使用する ID のことです。フェデレーテッド・システムでは、ニックネームを使用してデータ・ソース・オブジェクトにアクセスし、リモート・データ・ソース上の照会のパフォーマンスを改善できます。

フェデレーテッド・システムでのニックネーム

データ・ソースのデータを選択、または変更する場合は、
SELECT、INSERT、UPDATE、および DELETE ステートメントを使用してニックネームを照会します。 DB2 SQL の照会をフェデレーテッド・データベースへサブミットします。

1 つの SQL ステートメントを使用して、ローカル表からのデータとリモート・データ・ソースからのデータを、あたかもすべてのデータがローカルにあるように結合することができます。例えば、次のオブジェクトにあるデータを結合することができます。

- フェデレーテッド・データベース内のローカルの DB2 表、Oracle 表、および Sybase ビュー
- あるサーバー上の DB2 for z/OS の表、別のサーバー上の DB2 for z/OS の表、および Excel スプレッドシート

データ・ソースがフェデレーテッド・データベース内の通常のリレーショナル表またはビューであるかのように SQL ステートメントを処理することにより、フェデレーテッド・システムは、リレーショナル・データを非リレーショナルの形式のデータと結合することができます。

フェデレーテッド・データベースにある表とビューは、*ローカル・オブジェクト*です。これらのオブジェクトにはニックネームを作成しないでください。代わりに SQL ステートメントでは実際のオブジェクト名を使用します。

リモート・オブジェクト は、フェデレーテッド・データベース内にはないオブジェクトです。これらのオブジェクトにはニックネームを作成する必要があります。以下に例を示します。

- フェデレーテッド・システム上の別のデータベースまたはインスタンス内の表およびビュー
- 別のシステム上の別のデータベースまたはインスタンス内の表およびビュー
- Oracle、Sybase、ODBC などのデータ・ソース内の表およびビュー

WITH HOLD を指定したカーソル宣言

ニックネームで定義されたカーソルに WITH HOLD オプションを使用することができます。

DRDA ラッパーおよび DB2 Database for Linux, UNIX, and Windows データ・ソースの場合、WITH HOLD オプションを使用して宣言したカーソルは複数の作業単位でオープンしたままになります。

カーソル WITH HOLD を宣言するとき、データ・ソースがこのオプションをサポートしない場合、属性は無視されます。

トリガー

トリガー内では、ニックネームを更新の対象にすることはできません。

ニックネームに対する SELECT ステートメントは、トリガー本体に組み込めます。ニックネームに対する INSERT、UPDATE、または DELETE ステートメントは、トリガー本体に組み込めません。

ニックネームでのデータのアクセス

フェデレーテッド・システムでは、データがどこに保管されているかに関係なく、簡単にデータにアクセスできます。データにアクセスするには、表やビューなどのデータ・ソース・オブジェクトにニックネームを作成します。

例えば、ニックネーム DEPT がリモート表 EUROPE.PERSON.DEPT を表す場合、ステートメント SELECT * FROM DEPT を使用してリモート表にある情報を照会することができます。ニックネームを照会する際、データ・ソースに関する接続の詳細を覚えておく必要はありません。したがって、以下の問題を気に掛ける必要はありません。

- データ・ソース側のオブジェクトの名前
- データ・ソース・オブジェクトが存在するサーバー
- オブジェクトが存在するデータ・ソース・タイプ (Informix や Oracle など)
- データ・ソースが使用する照会言語や SQL ダイアレクト
- データ・ソースとフェデレーテッド・サーバー間でのデータ・タイプ・マッピング
- データ・ソースとフェデレーテッド・サーバー間での関数マッピング

フェデレーテッド・データベース・カタログにある基礎となるメタデータは、照会を処理するために必要な情報をフェデレーテッド・サーバーに提供します。このメタデータは、データ・ソースにアクセスするためにフェデレーテッド・サーバーとデータベースがセットアップされ、構成される際にデータ・ソースから収集されます。

フェデレーテッド・システムをセットアップした後、ニックネームを使用してデータ・ソースを照会したり、フェデレーテッド・システムの構成をさらに拡張することができます。

ニックネームを使用できる SQL ステートメント

SQL ステートメントはニックネームの使用をサポートします。

表 14. ニックネームを使用できる一般的な SQL ステートメント

SQL ステートメント	説明	必要な許可
ALTER NICKNAME	ローカルの列名、ローカルのデータ・タイプ、フェデレーテッド列オプション、またはインフォメーショナル制約を変更することにより、既存のニックネームを変更する。データ・ソース側の表またはビューは影響を受けない。	• SYSADM または DBADM • ニックネームに対する ALTER または CONTROL 特権 • ニックネームにスキーマ名がある場合は、スキーマに対する ALTERIN 特権 • ニックネームのカタログ・ビューの DEFINER 列に記録された、ニックネームの定義者
ALTER TABLE	透過 DDL を使用してフェデレーテッド・データベースで作成されたリモート表を変更する。データ・ソースでネイティブに作成された表を変更することはできません。インフォメーショナル制約を使用して、ニックネームに参照保全制約を追加できます。	• SYSADM または DBADM • ニックネームに対する ALTER または CONTROL 特権 • ニックネームにスキーマ名がある場合は、スキーマに対する ALTERIN 特権
COMMENT ON	各種のオブジェクト (関数、関数マッピング、索引、ニックネーム、サーバー、サーバー・オプション、タイプ・マッピング、およびラッパーを含む) のカタログ記述内のコメントを追加または置き換える。	• SYSADM または DBADM • オブジェクトに対する ALTER または CONTROL 特権 • スキーマに対する ALTERIN 特権 • オブジェクトのカタログ・ビューの DEFINER 列に記録されたオブジェクトの定義者
CREATE ALIAS	ニックネームの別名を定義する。	• SYSADM または DBADM • 別名の暗黙的または明示的スキーマ名が存在しない場合は、データベースに関する IMPLICIT_SCHEMA 権限 • 別名のスキーマ名が既存のスキーマを指す場合は、スキーマに対する CREATEIN 特権
CREATE INDEX (SPECIFICATION ONLY 文節を指定)	データ・ソース・オブジェクトが索引を持つことを照会オプティマイザーに知らせる索引の仕様 (メタデータ) を作成する。実際の索引は作成されず、指定のみが作成される。	• SYSADM または DBADM • 基になるデータ・ソース・オブジェクトに対する CONTROL または INDEX 特権 - およびデータベースに対する IMPLICIT_SCHEMA 権限またはスキーマに対する CREATEIN 特権

表 14. ニックネームを使用できる一般的な SQL ステートメント (続き)

SQL ステートメント	説明	必要な許可
CREATE TABLE (OPTIONS 文節を指定)	透過 DDL を使用してフェデレーテッド・データベースでリモート表を作成する。	• SYSADM または DBADM • データベースに対する CREATETAB 特権と表スペースに対する USE 特権 - および データベースに対する IMPLICIT_SCHEMA 権限またはスキーマに対する CREATEIN 特権
CREATE TABLE (AS 全選択および DATA INITIALLY DEFERRED REFRESH 文節を指定)	ニックネームを参照する全選択を使用してマテリアライズ照会表を作成する。	• SYSADM または DBADM • データベースに対する CREATETAB 特権と表スペースに対する USE 特権 - および データベースに対する IMPLICIT_SCHEMA 権限またはスキーマに対する CREATEIN 特権 • 表またはビューに対する CONTROL 特権 • 表またはビューに対する SELECT 特権と、 REFRESH DEFERRED が指定されている場合には ALTER 特権
CREATE VIEW	1 つ以上のニックネームを参照するビューを作成する。	• SYSADM または DBADM • ニックネームに対する CONTROL または SELECT 特権 - および データベースに対する IMPLICIT_SCHEMA 権限またはスキーマに対する CREATEIN 特権
DELETE	ニックネームを持つデータ・ソース・オブジェクト (表またはビューなど) から行を削除する。	• SYSADM または DBADM • ニックネームに対する DELETE 特権および基になるデータ・ソース・オブジェクトに対する DELETE 特権 • 基になるデータ・ソース・オブジェクトに対する CONTROL 特権

表 14. ニックネームを使用できる一般的な SQL ステートメント (続き)

SQL ステートメント	説明	必要な許可
DROP	ニックネーム、フェデレーテッド・ビュー、または索引の仕様などのオブジェクトを削除する。データ・ソース側の表、ビュー、または索引は影響を受けない。 透過 DDL を使用して作成される表がドロップされると、その表の対応するニックネームもドロップされます。	• SYSADM または DBADM • オブジェクトのスキーマに対する DROPIN 特権 • オブジェクトに対する CONTROL 特権
GRANT	ニックネームおよびフェデレーテッド・ビューに対する特権 (ALTER、DELETE、INDEX、INSERT、SELECT、または UPDATE など) を付与する。データ・ソース側の特権は、別々に付与する必要があります。	• SYSADM または DBADM • 指定されたそれぞれの特権に対する WITH GRANT OPTION • オブジェクトに対する CONTROL 特権
INSERT	ニックネームを持つデータ・ソース・オブジェクト (表やビューなど) に行を挿入する。	• SYSADM または DBADM • ニックネームに対する INSERT 特権および基になるデータ・ソース・オブジェクトに対する INSERT 特権 • 基になるデータ・ソース・オブジェクトに対する CONTROL 特権
LOCK TABLE	データ・ソース側のリモート・オブジェクトをロックする。ニックネームを持つデータ・ソース表を、並行するアプリケーション処理が同時に変更することを防止する。	• SYSADM または DBADM • 基礎表に対する SELECT 特権 • 基礎表に対する CONTROL 特権
REVOKE	ニックネームおよびフェデレーテッド・ビューに対する特権 (ALTER、DELETE、INDEX、INSERT、SELECT、または UPDATE など) を取り消す。データ・ソース側の特権は、別々に取り消す必要があります。	• SYSADM または DBADM • オブジェクトに対する CONTROL 特権
SELECT	ニックネームを持つデータ・ソース・オブジェクト (表またはビューなど) から行を選択する。	• SYSADM または DBADM • ニックネームに対する SELECT 特権および基になるデータ・ソース・オブジェクトに対する SELECT 特権 • 基になるデータ・ソース・オブジェクトに対する CONTROL 特権

表 14. ニックネームを使用できる一般的な SQL ステートメント (続き)

SQL ステートメント	説明	必要な許可
UPDATE	ニックネームを持つデータ・ソース・オブジェクト (表またはビューなど) の行にある指定された列の値を更新する。	• SYSADM または DBADM • ニックネームに対する UPDATE 特権および基になるデータ・ソース・オブジェクトに対する UPDATE 特権 • 基になるデータ・ソース・オブジェクトに対する CONTROL 特権

フェデレーテッド・データベースに照会をサブミットすると、照会内のニックネームに対する許可特権がチェックされます。ニックネームが参照するデータ・ソース・オブジェクトの許可要件は、照会がデータ・ソースで処理される時にのみ適用されます。

ニックネームを使用して、データを選択、挿入、更新、または削除するには、ステートメントの許可 ID には次の特権が必要です。

- フェデレーテッド・データベースが要求を受け入れるための、ニックネームに対する適切な特権
- データ・ソースが要求を受け入れるための、基になる表オブジェクトに対する適切な特権

例えば、ニックネームを使用してデータ・ソースを更新するには、ニックネームに対する UPDATE 特権および、基になるデータ・ソース・オブジェクトに対する UPDATE 特権が必要です。

テンポラル表のニックネームの作成

テンポラル表のフェデレーション・サポートにより、リモート・テンポラル表にニックネームを作成することができます。

テンポラル属性を使用して表を定義して、以下のタイプのテンポラル表を作成することができます。

- 行のヒストリカル・バージョンを維持するシステム期間テンポラル表
- 時間基準に基づいてデータを保管するアプリケーション期間テンポラル表
- システム期間テンポラル表とアプリケーション期間テンポラル表の組み合わせであるバイテンポラル表

テンポラル表にニックネームを作成すると、そのニックネームに対して挿入、更新、および削除操作を実行することができます。

例

- *policy_info_bus* という名前の表を定義します。

```
CREATE TABLE policy_info_bus ( policy_id CHAR(4) NOT NULL,
 bus_start DATE NOT NULL,  bus_end DATE NOT NULL,
 PERIOD BUSINESS_TIME(bus_start, bus_end) );
```

- テンポラル表のニックネームに対して照会を発行します。

```
SELECT * from policy_info_bus_nickname;
```

ただし、ニックネームに対するテンポラル操作はサポートされていません。ニックネームに対してテンポラル DDL 操作、変更操作、または照会操作を実行することはできません。例えば、次の例に示すように、特定の日付時点のシステム時刻を照会することはできません。

```
SELECT * FROM policy_info_bus_nickname FOR BUSINESS_TIME AS OF '2012-12-02';
```

新しいデータ・ソース・オブジェクトへのアクセス

新しいデータ・ソース・オブジェクトにアクセスするには、それらにニックネームを作成する必要があります。ニックネームを持たないデータ・ソースに、CREATE NICKNAME ステートメントを使用します。

始める前に

フェデレーテッド・システムは、データ・ソースにアクセスするように構成しておく必要があります。

オブジェクトが存在するデータ・ソース・サーバーのサーバー定義は、フェデレーテッド・データベースになければなりません。 CREATE SERVER ステートメントを使用してサーバー定義を作成します。

ニックネームを使用して、データを挿入、更新、または削除するには、以下のすべての特権が必要です。

- ステートメントの許可 ID が持つ特権には、フェデレーテッド・データベースが要求を受け入れるための、ニックネームに対する必要な SELECT、INSERT、UPDATE、および DELETE 特権が含まれている必要があります。
- データ・ソース側のユーザー ID は、データ・ソースが要求を受け入れるための、基になる表オブジェクトに対する必要な SELECT、INSERT、UPDATE、および DELETE 特権を持っている必要があります。
- データ・ソース側のユーザー ID は、ユーザー・マッピングを介してフェデレーテッド・サーバー側の許可 ID にマップする必要があります。

CREATE NICKNAME ステートメントに対する必要な許可を持っていなければなりません。

- SYSADM または DBADM
- ニックネームの暗黙的または明示的スキーマ名が存在しない場合は、フェデレーテッド・データベースに関する IMPLICIT_SCHEMA 権限
- ニックネームにスキーマ名がある場合は、スキーマに対する CREATEIN 特権

このタスクについて

定期的に、ニックネームのないデータ・ソース・オブジェクトにアクセスする必要があります。これらは、新しく作成されたビューなど、データ・ソースに追加された新しいオブジェクトかもしれません。あるいは、最初にセットアップされた時にフェデレーテッド・サーバーに登録されていない、既存のオブジェクトである場合

もあります。どちらの場合にも、オブジェクトのニックネームを作成する必要があります。

手順

新規データ・ソース・オブジェクトにアクセスするには、CREATE NICKNAME ステートメントを発行します。

リレーショナル・データ・ソースと非リレーショナル・データ・ソースのニックネームの作成

CREATE NICKNAME ステートメントは、リレーショナル・データ・ソースと非リレーショナル・データ・ソースとでは、わずかに異なります。

リレーショナル・データ・ソースの場合は、CREATE NICKNAME ステートメントの構文は次のとおりです。

```
CREATE NICKNAME nickname_name FOR server_name."remote_schema"."object_name"
```

nickname_name
データ・ソース・オブジェクトのユニークなニックネーム。ニックネームの長さは 128 文字までです。

ニックネームは、2 つの部分 (スキーマとニックネーム) からなる名前です。ニックネームの作成時にスキーマを省略すると、そのニックネームのスキーマはニックネームを作成したユーザーの認証 ID になります。スキーマ名のデフォルト値は、以下の階層を基に選択されます。

1. CURRENT SCHEMA 特殊レジスター
2. SESSION_USER 特殊レジスター
3. SYSTEM USER 特殊レジスター
4. データベースに接続された許可 ID

FOR *server_name."remote_schema"."object_name"*
リモート・データ・ソース・オブジェクトを表す、3 つの部分からなる ID。データ・ソースがスキーマをサポートしていない場合は、CREATE NICKNAME ステートメントからスキーマを省略してください。

- *server_name* は、CREATE SERVER ステートメントでデータ・ソース・サーバーに割り当てた名前です。
- *remote_schema* は、オブジェクトが属するリモート・スキーマの名前です。
- *object_name* は、アクセスしたいリモート・オブジェクトの名前です。

OPTIONS (*options_list*)
SQL 照会コンパイラーおよびラッパーが効果的に照会を実行できるようにするニックネームに関する情報。

非リレーショナル・データ・ソースの場合は、CREATE NICKNAME ステートメントの構文は次のとおりです。

```
CREATE NICKNAME nickname_name column_definition_list
     FOR SERVER server_name
     OPTIONS (options_list)
```

nickname_name
データ・ソース・オブジェクトの固有のニックネーム (上述のリレーショナル・データ・ソースのとおり)。

column_definition_list
ニックネーム列およびデータ・タイプのリスト。

FOR SERVER *server_name*
サーバー定義情報 CREATE SERVER ステートメントで作成した、リモート・サーバーのローカル名。

OPTIONS (*options_list*)
SQL 照会コンパイラーおよびラッパーが効果的に照会を実行できるようにするニックネームに関する情報。

ニックネーム列と索引名

ニックネームを作成する際に、フェデレーテッド・サーバーはニックネーム列と索引名を生成するアルゴリズムを使用します。

バージョン 9.5 ではこのアルゴリズムが拡張され、ニックネームの列名および索引名が元の名前に可能な限り近くなるように突き合わされます。

このアルゴリズムはリレーショナル・データ・ソースのみに適用されます。非リレーショナル・データ・ソースの場合、CREATE NICKNAME ステートメントで指定されている名前は変更されないままになります。

以下の表は、リモート列名からの生成結果のニックネーム列名の例を示しています。これらの例のリモート列は、それぞれ別のリモート表の列です。リモート列名とは違うニックネーム列名は太字で示されています。

表 15. リモート列名から生成されるニックネーム列名

	データ・ソースに関する現在のアルゴリズムの結果の名前			**バージョン 9.5 より前の結果の名前 (アルゴリズムは非英数字の文字を変換する)**
リモート列名	**ID を大文字に変換する**	**ID を小文字に変換する**	**ID を変更しない**	
column1	column1	**COLUMN1**	**COLUMN1**	**COLUMN1**
Column1	Column1	Column1	**COLUMN1**	**COLUMN1**
COLUMN1	COLUMN1	COLUMN1	COLUMN1	COLUMN1
column-1	column-1	column-1	column-1	**COLUMN_1**
Column-1	Column-1	Column-1	Column-1	**COLUMN_1**
COLUMN-1	COLUMN-1	COLUMN-1	COLUMN-1	**COLUMN_1**

リモート列が同じ表にある場合、フェデレーテッド・サーバーは固有の名前を生成する既存のアルゴリズムを使用します。例えば、ID が変更されないデータ・ソースのニックネーム列名の場合、リモート列名は以下の名前になります。

- column1 は COLUMN1 になります。

- Column1 は COLUMN10 になります。
- COLUMN1 は COLUMN11 になります。

パススルー・セッションを使用したデータ・ソースへのアクセス

パススルー と呼ばれる特殊モードを使用すると、SQL ステートメントをデータ・ソースに直接サブミットすることができます。パススルー・セッションでは、そのデータ・ソース用の SQL ダイアレクトで SQL ステートメントをサブミットできます。

パススルー・セッションは、SQL ではできない操作を実行する場合に使用してください。例えば、プロシージャーの作成、索引の作成、またはデータ・ソースに固有のダイアレクトを使用した照会の実行には、パススルー・セッションを使用します。

パススルーをサポートするデータ・ソースは、パススルー・セッションでのみ SQL ステートメントを受け入れます。

同様に、パススルー・セッションを使用して、SQL がサポートしていないアクション (例えば、ある種の管理用タスク) を実行することもできます。実行できる管理タスクは、データ・ソースによって異なります。例えば、DB2 Database for Linux, UNIX, and Windows の場合、データ・ソースのために統計情報ユーティリティーを実行することはできますが、リモート・データベースを開始または停止することはできません。

パススルー・セッションでは、同時に照会できるデータ・ソースは 1 つだけです。SET PASSTHRU コマンドを使用してセッションを開きます。 SET PASSTHRU RESET コマンドを使用すると、パススルー・セッションはクローズされます。SET PASSTHRU RESET コマンドではなく SET PASSTHRU コマンドを使用すると、現行のパススルー・セッションはクローズされ、新しいパススルー・セッションが開かれます。

パススルー・セッションでカーソル WITH HOLD を宣言できます。このオプションを、DRDA ラッパーおよび DB2 Database for Linux, UNIX, and Windows データ・ソースと併用すると、宣言したカーソルは複数の作業単位でオープンしたままになります。カーソル WITH HOLD を宣言するとき、データ・ソースがこのオプションをサポートしない場合、属性は無視されます。

パススルー・セッションは非リレーショナル・データ・ソースでは使用できません。

フェデレーテッド・ビューを使用した異種データへのアクセス

フェデレーテッド・ビュー とは、フェデレーテッド・データベース内のビューであり、その基本表はリモート・データ・ソース側にあります。フェデレーテッド・ビューは、データ・ソースの表名の代わりにニックネームを使用して基本表を参照します。

始める前に

CREATE VIEW ステートメントを発行するには、次のいずれかの権限が必要です。

- SYSADM または DBADM
- 全選択でのすべてのニックネームごとに:
  - 基になる表またはビューに対する CONTROL または SELECT 特権
  - 以下の権限または特権のいずれか
    - ビューの暗黙的または明示的スキーマ名が存在しない場合は、フェデレーテッド・データベースに関する IMPLICIT_SCHEMA 権限
    - ビューのスキーマ名が既存のスキーマを指す場合は、スキーマに対する CREATEIN 特権

フェデレーテッド・データベースのニックネームにビューを定義する場合、基になるオブジェクトに対する特権は考慮されません。

このタスクについて

フェデレーテッド・ビューから照会すると、データはリモート・データ・ソースから検索されます。データ・ソース・データのフェデレーテッド・データベース・ビューを作成するアクションは、「ニックネームのビューの作成」と呼ばれることがあります。この理由は、ビューの作成時にデータ・ソースの代わりにニックネームを参照するためです。

集中化されたリレーショナル・データベース・マネージャーにおいて、複数のローカル表に定義されたビューが果たす役割と同じように、これらのフェデレーテッド・ビューは、グローバルに統合されたデータベースに対して高度なデータの独立性を提供します。

制約事項:

- UNION ALL ステートメントで作成されるフェデレーテッド・ビューは、読み取り専用ビューです。
- FROM 文節に複数のニックネームが組み込まれているフェデレーテッド・ビューは、読み取り専用ビューです。
- FROM 文節に 1 つのニックネームしか組み込まれていないフェデレーテッド・ビューであっても、読み取り専用ビューである可能性があります。
  - FROM 文節のニックネームが非リレーショナル・データ・ソースに関するものであれば、フェデレーテッド・ビューは読み取り専用です。
  - ビューの作成時にその他のニックネームを述部または副照会として組み込む場合は、フェデレーテッド・ビューは更新可能です。

手順

フェデレーテッド・ビューを作成するには、CREATE VIEW ステートメントを発行します。
このニックネームで示されている表またはビューのデータ・ソースの許可要件は、照会の処理時に適用されます。ステートメントの許可 ID は、ユーザー・マッピングにより別のリモート許可 ID にマップすることができます。

フェデレーテッド・ビューの作成 - 例

以下の例は、いくつかのデータ・ソースからのデータにアクセスするための、フェデレーテッド・ビューの作成方法を示しています。例では、フェデレーションの CREATE VIEW ステートメントの構文を表示しています。

例: いくつかのデータ・ソース・オブジェクトから類似データをマージするフェデレーテッド・ビューの作成

ヨーロッパ、アジア、および南アメリカの別個のサーバー上にある顧客データを処理しているとします。ヨーロッパの顧客データは Oracle 表内にあります。その表のニックネームは ORA_EU_CUST です。アジアの顧客データは Sybase 表内にあります。その表のニックネームは SYB_AS_CUST です。南アメリカの顧客データは Informix 表内にあります。その表のニックネームは INFMX_SA_CUST です。各表には、顧客番号 (CUST_NO)、顧客名 (CUST_NAME)、製品番号 (PROD_NO)、および注文数量 (QUANTITY) を含む列があります。これらのニックネームから、この顧客データをマージするビューを作成するには、以下のステートメントを発行します。

```
CREATE VIEW FV1
   AS SELECT * FROM ORA_EU_CUST
   UNION
   SELECT * FROM SYB_AS_CUST
   UNION
   SELECT * FROM INFMX_SA_CUST
```

例: フェデレーテッド・ビューを作成するためのデータの結合

別々のサーバー上にある顧客データと販売データを処理するとします。顧客データは Oracle 表内にあります。その表のニックネームは ORA_EU_CUST です。販売データは Sybase 表内にあります。その表のニックネームは SYB_SALES です。顧客情報と顧客の購入品とを対応させたいとします。各表には、顧客番号 (CUST_NO) を含む列があります。これらのニックネームから、このデータを結合するフェデレーテッド・ビューを作成するには、以下のステートメントを発行します。

```
CREATE VIEW FV4
   AS SELECT A.CUST_NO, A.CUST_NAME, B.PROD_NO, B.QUANTITY
   FROM ORA_EU_CUST A, SYB_SALES B
   WHERE A.CUST_NO=B.CUST_NO
```

ニックネームに対するニックネームの作成

ニックネームに対してニックネームを作成することができます。

手順

1. Windows フェデレーテッド・サーバーで、IBM InfoSphere Federation Server をインストールする。
2. Excel データ・ソースにアクセスするように Windows フェデレーテッド・サーバーを構成する。
3. Windows フェデレーテッド・サーバー上で Excel スプレッドシートのニックネームを作成する。
4. AIX® フェデレーテッド・サーバーで、IBM InfoSphere Federation Server をインストールする。

5. DB2 ファミリー・データ・ソースにアクセスするように AIX フェデレーテッド・サーバーを構成する。
6. AIX フェデレーテッド・サーバー上で Windows フェデレーテッド・サーバー上の Excel ニックネームのニックネームを作成する。

例

AIX フェデレーテッド・サーバーから Microsoft Excel のスプレッドシートにアクセスする

AIX 上のフェデレーテッド・サーバーと Windows 上のフェデレーテッド・サーバーがあります。この両方のフェデレーテッド・サーバーから Excel のスプレッドシートにアクセスしたいとします。しかし、Excel ラッパーは Windows 上のフェデレーテッド・サーバー上でしかサポートされません。 AIX フェデレーテッド・サーバーから Excel スプレッドシートにアクセスするには、次のようにします。

フェデレーテッド・システム内のデータの選択

フェデレーテッド・システムでデータを選択する方法は、選択するデータ・ソースのタイプにより異なります。

フェデレーテッド・システムで使用される分散要求のタイプには、次のような照会を要求するものがあります。

- 単一のリモート・データ・ソース
- 1 つのローカル・データ・ソースと 1 つのリモート・データ・ソース
- 複数のリモート・データ・ソース
- リモート・データ・ソースとローカル・データ・ソースの組み合わせ

データ・ソースからデータを選択するには、SELECT ステートメントでデータ・ソース・オブジェクトのニックネームを使用します。

フェデレーテッド・データベースはローカル・データ・ソースです。フェデレーテッド・データベースにある表とビューは、ローカル・オブジェクトです。これらのオブジェクトにはニックネームを作成せず、SELECT ステートメントでは実際のオブジェクト名を使用します。

リモート・データ・ソースには、フェデレーテッド・サーバー上の別の DB2 Database for Linux, UNIX, and Windows のデータベース・インスタンス、別のサーバー上の別の DB2 Database for Linux, UNIX, and Windows のデータベース・インスタンス、および他のデータ・ソースが含まれます。リモートのデータ・ソースにあるオブジェクトは、リモート・オブジェクトです。

フェデレーテッド・システム内のデータの選択 - 例

このトピックでは、フェデレーテッド・サーバーが複数のデータ・ソースにアクセスするシナリオを示し、SELECT ステートメントの例を提供します。

例: フェデレーテッド・サーバーが DB2 for z/OS データ・ソース、DB2 for System i データ・ソース、および Oracle データ・ソースにアクセスするように構成されているとします。それぞれのデータ・ソースには、売上情報を含む 1 つの表

が保管されています。この構成は次の図に描かれています。

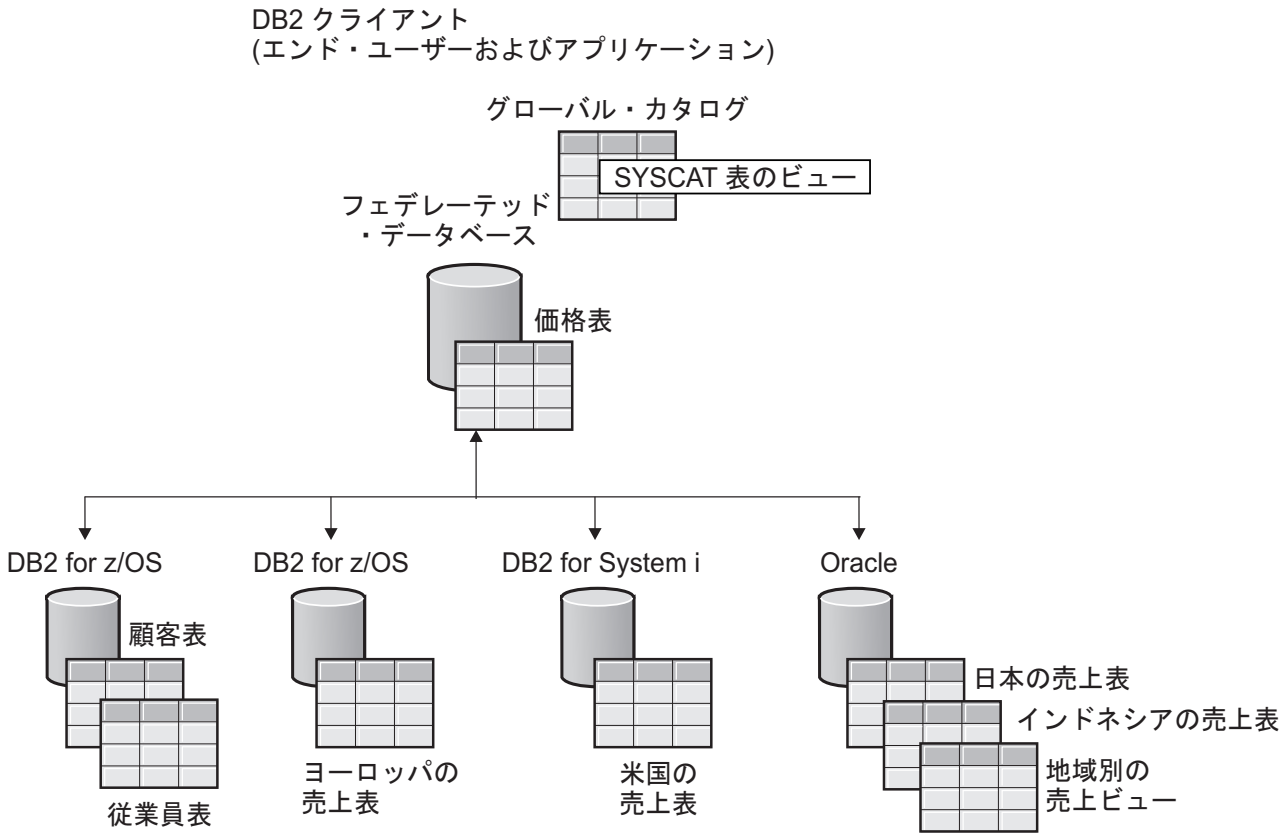


図 8. DB2 データ・ソースと Oracle データ・ソースを含むフェデレーテッド・システムの例

これらの表には、顧客番号 (CUST_NO)、注文数量 (QUANTITY) および注文商品番号 (PROD_NO) を記録する列があります。さらに、フェデレーテッド・データベースのローカル表には、価格情報が入っています。価格表には、商品番号 (PROD_NO) および現行価格 (PRICE) を記録する列があります。

リモート・データ・ソース・オブジェクトのニックネームは、次の表に示すような SYSCAT.TABLES 表に格納されています。 TYPE 列はオブジェクトのタイプを示します。例えば、ニックネーム (N)、ローカル表 (T)、またはビュー (V) などです。

表 16. データ・ソース情報

データ・ソース・オブジェクト名	オブジェクトのタイプ	場所
PRICES	ローカル表	フェデレーテッド・データベース
EUROPE_SALES	リモート表	DB2 for z/OS データベース
US_SALES	リモート表	DB2 for System i データベース
JAPAN_SALES	リモート表	Oracle データベース
SALES_BY_REGION	リモート・ビュー	Oracle データベース

表 17. SYSCAT 表

TABNAME	TYPE
PRICES	T
FED_PRICES	N
Z_EU_SALES	N
Si_US_SALES	N
ORA_JAPANSALES	N
ORA_REGIONSALES	N
...	

ニックネームを使用してデータを選択するには、以下のすべての特権が必要です。

- ステートメントの許可 ID が持つ特権には、ニックネームに対する SELECT 特権 (フェデレーテッド・データベースが要求を受け入れるための) が含まれている必要があります。
- データ・ソース側のユーザー ID は、基になる表オブジェクトに対する SELECT 特権 (データ・ソースが要求を受け入れるための) を持っている必要があります。
- データ・ソース側のユーザー ID は、ユーザー・マッピングを介してフェデレーテッド・サーバー側の許可 ID にマップする必要があります。

以下の SELECT ステートメントの例は、上述のサンプル・フェデレーテッド・システムを使用しています。

例: 単一のデータ・ソースを照会する

Z_EU_SALES には、ヨーロッパの顧客が注文した商品が入っています。また、個々の売上で注文された数量も含まれています。この照会は SELECT ステートメントで ORDER BY 文節を使用してヨーロッパでの売上をリストし、リストを顧客番号でソートしています。

```
SELECT CUST_NO, PROD_NO, QUANTITY
     FROM Z_EU_SALES
     ORDER BY CUST_NO
```

例: ローカル・データ・ソースとリモート・データ・ソースを結合する

PRICES は、フェデレーテッド・データベース内にある表です。これには、販売する商品の価格リストが入っています。このローカル表から、Z_EU_SALES にリストされた商品に対応する価格を選択します。また、結果セットは顧客番号でソートします。

```
SELECT sales.CUST_NO, sales.PROD_NO, sales.QUANTITY
     FROM Z_EU_SALES sales, PRICES
     WHERE sales.PROD_NO=PRICES.PROD_NO
     ORDER BY sales.CUST_NO
```

例: 複数のリモート・データ・ソースを照会する

それぞれの地域からすべての売上情報を収集し、結果セットは商品番号順に並べることとします。

```
WITH GLOBAL_SALES (Customer, Product, Quantity) AS
     (SELECT CUST_NO, PROD_NO, QUANTITY FROM Z_EU_SALES
       UNION ALL
     SELECT CUST.NO,PROD.NO, QUANTITY FROM iS_US_SALES
       UNION ALL
     SELECT CUST.NO,PROD.NO, QUANTITY FROM ORA_JAPANSALES)
     SELECT Customer, Product, Quantity
     FROM GLOBAL_SALES
     ORDER BY Product
```

Oracle データ・ソース側のビューに日本とインドネシアの売上がリストされているとします。このビューのニックネームは ORA_SALESREGION です。この情報を米国の売上と結合し、それぞれの売上の隣りに商品価格を表示することとします。

```
SELECT us_jpn_ind.CUST_NO, us_jpn_ind.PROD_NO,
        us_jpn_ind.QUANTITY, us_jpn_ind.QUANTITY*PRICES.PRICE
     AS SALEPRICE FROM
     (SELECT CUST_NO, PROD_NO, QUANTITY
     FROM ORA_SALESREGION
     UNION ALL
     SELECT CUST_NO, PROD_NO, QUANTITY
     FROM iS_US_SALES us ) us_jpn_ind,PRICES
     WHERE us_jpn_ind.PROD_NO = PRICES.PROD_NO
     ORDER BY SALEPRICE DESC
```

ニックネームのインフォメーショナル制約

ニックネームに対してインフォメーショナル制約を使用することで、照会のパフォーマンスを改善できます。インフォメーショナル制約は、パフォーマンスを改善するためにオプティマイザーが使用する規則ですが、この規則をデータベース・マネージャーが施行することはできません。

ニックネームに対して以下を指定できます。

- 参照制約
- チェック制約
- 関数依存関係制約
- 主キー制約
- ユニーク制約

ニックネームのインフォメーショナル制約の指定

リモート・データ・ソースでの照会のパフォーマンスを改善するために、ニックネームに対してインフォメーショナル制約を追加することができます。しかし、フェデレーテッド・サーバー自身がこの制約を施行または検査することはありません。

このタスクについて

制約事項

ニックネームのインフォメーショナル制約を定義したなら、インフォメーショナル制約を除去するまで、そのニックネームの列名は変更できません。

このタスクについて

リレーショナル・データ・ソースの場合は、ニックネームを変更するときにインフォメーショナル制約を指定できます。

非リレーショナル・データ・ソースの場合は、ニックネームを作成または変更するときにインフォメーショナル制約を指定できます。

手順

DB2 コマンド行からニックネームにインフォメーショナル制約を指定するには、CREATE NICKNAME ステートメントまたは ALTER NICKNAME ステートメントに適切な制約属性を指定して発行します。

ニックネームのインフォメーショナル制約の指定例

次の例は、ニックネームのインフォメーショナル制約の使い方を示したものです。チェック制約、参照制約、およびその他のデータ構造に対して、CREATE または ALTER NICKNAME ステートメントを使用します。

例: インフォメーショナル・チェック制約

次のリモート表では、saraly 列のデータは常に 10000 より大きい値です。

```
CREATE TABLE account.salary (
    empno INTEGER NOT NULL PRIMARY KEY,
    salary  INTEGER NOT NULL
);
```

この表のニックネームを、次のようにして作成します。

```
CREATE NICKNAME account.salary FOR myserv.account.salary;
```

次いで、次のステートメントを実行して、そのニックネームのインフォメーショナル・チェック制約を追加します。

```
ALTER NICKNAME account.salary ADD CONSTRAINT cons1 CHECK( salary > 10000 )
NOT ENFORCED
ENABLE QUERY OPTIMIZATION;
```

例: ニックネームからニックネームへのインフォメーショナル参照制約

この例では、N1 と N2 という 2 つのニックネームがあります。ニックネーム N2 の列 F1 には、ニックネーム N1 の列 P1 のキー値が入っています。次のステートメントを実行すると、ニックネーム N2 に参照制約を定義できます。

```
ALTER NICKNAME SCHEMA1.N2 ADD CONSTRAINT ref1
                          FOREIGN KEY (F1) REFERENCES SCHEMA1.N1 (P1)
                          NOT ENFORCED;
```

例: ニックネームから表へのインフォメーショナル参照制約

この例では、ニックネーム N3 の列 F1 には、表 T1 の列 P1 のキー値が入っています。次のステートメントを実行すると、ニックネーム N3 に参照制約を定義できます。

```
ALTER NICKNAME SCHEMA1.N3 ADD CONSTRAINT ref1
                          FOREIGN KEY (F1) REFERENCES SCHEMA1.T1 (P1)
                          NOT ENFORCED;
```

例: 表からニックネームへのインフォメーショナル参照制約

この例では、表 T2 の列 F1 には、ニックネーム N4 の列 P1 のキー値が入っています。次のステートメントを実行すると、表 T2 に参照制約を定義できます。

```
ALTER TABLE SCHEMA1.T2 ADD CONSTRAINT ref1
                        FOREIGN KEY (F1) REFERENCES SCHEMA1.N4 (P1)
                        NOT ENFORCED;
```

例: 関数依存関係

この例では、C1 と C2 の列のペアが列 P1 の値を一意的に決定します。次のステートメントを実行すると、関数依存関係を定義できます。

```
ALTER NICKNAME SCHEMA1.NICK1 ADD CONSTRAINT FD1 CHECK( P1 DETERMINED BY (C1,C2) )
                       NOT ENFORCED ENABLE QUERY OPTIMIZATION;
```

例: 表構造ファイル

このステートメントは、表構造ファイルの主キーを定義したものです。

```
CREATE NICKNAME MY_FILE (
       X INTEGER NOT NULL,
       Y INTEGER,
       PRIMARY KEY (X) NOT ENFORCED
)  FOR SERVER MY_SERVER OPTIONS(FILE_PATH '/usr/pat/DRUGDATA1.TXT');
```

スタースキーマ

ステートメントは、4 つのディメンション表と 1 つのファクト表を作成します。

```
CREATE TABLE SCHEMA.FACT (
       LOCATION_CODE   INTEGER NOT NULL,
       PRODUCT_CODE    INTEGER NOT NULL,
       CUSTOMER_CODE   INTEGER NOT NULL,
       SDATE           DATE NOT NULL,
       SALES           INTEGER NOT NULL
);

CREATE TABLE SCHEMA.LOCATION (
       LOCATION_CODE   INTEGER NOT NULL PRIMARY KEY,
       STATE           CHAR(2) NOT NULL,
       SHOP_ID         INTEGER NOT NULL,
       ...
);

CREATE TABLE SCHEMA.PRODUCT (
       PRODUCT_CODE    INTEGER NOT NULL PRIMARY KEY,
       PRODUCT_CAT     INTEGER NOT NULL,
       PRODUCT_NAME    VARCHAR(20) NOT NULL,
       ...
);

CREATE TABLE SCHEMA.CUSTOMER (
       CUSTOMER_CODE   INTEGER NOT NULL PRIMARY KEY,
       NAME            VARCHAR(20) NOT NULL,
       TEL             VARCHAR(10) NOT NULL,
       ...
);

CREATE TABLE SCHEMA.TIMEDIM (
       SDATE           DATE NOT NULL UNIQUE,
       YEAR            INTEGER NOT NULL,
       QUARTER         INTEGER NOT NULL,
       ...
);
```

フェデレーテッド・サーバーは、ファクト表とディメンション表に次のニックネームを作成します。

```
CREATE NICKNAME SCHEMA.FACT FOR SERVER.SCHEMA.FACT;
CREATE NICKNAME SCHEMA.LOCATION FOR SERVER.SCHEMA.LOCATION;
CREATE NICKNAME SCHEMA.PRODUCT FOR SERVER.SCHEMA.PRODUCT;
CREATE NICKNAME SCHEMA.CUSTOMER FOR SERVER.SCHEMA.CUSTOMER;
CREATE NICKNAME SCHEMA.TIMEDIM FOR SERVER.SCHEMA.TIMEDIM;
```

以下のステートメントを発行して、以下の外部キー関係を定義することができます。

```
ALTER NICKNAME SCHEMA.FACT ADD CONSTRAINT L1 FOREIGN KEY (LOCATION_CODE)
                               REFERENCES SCHEMA.LOCATION(LOCATION_CODE)
                               NOT ENFORCED ENABLE QUERY OPTIMIZATION;

ALTER NICKNAME SCHEMA.FACT ADD CONSTRAINT P1 FOREIGN KEY (PRODUCT_CODE)
                               REFERENCES SCHEMA.PRODUCT(PRODUCT_CODE)
                               NOT ENFORCED ENABLE QUERY OPTIMIZATION;

ALTER NICKNAME SCHEMA.FACT ADD CONSTRAINT C1 FOREIGN KEY (CUSTOMER_CODE)
                               REFERENCES SCHEMA.CUSTOMER(CUSTOMER_CODE)
                               NOT ENFORCED ENABLE QUERY OPTIMIZATION;

ALTER NICKNAME SCHEMA.FACT ADD CONSTRAINT S1 FOREIGN KEY (SDATE)
                               REFERENCES SCHEMA.TIMEDIM(SDATE)
                               NOT ENFORCED ENABLE QUERY OPTIMIZATION;
```

CUSTOMER ニックネームの TEL 列の値が固有である場合には、このステートメントを使って、次のインフォメーショナル・ユニーク制約を追加することができます。

```
ALTER NICKNAME SCHEMA.CUSTOMER ADD CONSTRAINT U1 UNIQUE( TEL )
                                    NOT ENFORCED ENABLE QUERY OPTIMIZATION;
```

LOCATION ニックネームの SHOP_ID 列の値が LOCATION_ID 列の値を一意的に決定する場合には、このステートメントを使って、次の関数依存関係を定義することができます。

```
ALTER NICKNAME SCHEMA.LOCATION
ADD CONSTRAINT F1 CHECK( LOCATION_ID DETERMINED BY SHOP_ID )
                        NOT ENFORCED ENABLE QUERY OPTIMIZATION;
```

TIMEDIM ニックネームの QUARTER 列の値は 1 から 4 なので、このステートメントを使って、次のインフォメーショナル・チェック制約を定義することができます。

```
ALTER NICKNAME SCHEMA.TIMEDIM
ADD CONSTRAINT Q1 CHECK( QUARTER BETWEEN 1 AND 4 )
                        NOT ENFORCED ENABLE QUERY OPTIMIZATION;
```

この例のステートメントはリモート表のニックネームを作成します。リモート表に主キーがある場合は、ニックネームも主キーを取ります。ビューのニックネームを作成した場合は、ニックネームには主キーがありません。この場合、ニックネームを変更して、インフォメーショナル主キー制約を追加することができます。以下に例を示します。

```
CREATE NICKNAME SCHEMA.LOCATION FOR SERVER.SH.V_LOCATION;
ALTER NICKNAME SCHEMA.LOCATION
      ADD CONSTRAINT P1 PRIMARY KEY ( LOCATION_CODE ) NOT ENFORCED;
```

ニックネーム統計情報の更新

ニックネーム統計情報を取り出すと、照会オプティマイザーが照会アクセス・プランを生成する際にニックネームに関する最新情報を使用していることを確認できます。

ニックネーム統計情報の更新機能 - 概要

ニックネームの統計情報を検索して、ニックネームに関する正確で最新の情報を照会オプティマイザーに提供します。オプティマイザーはこの情報を使用して、照会の最適なアクセス・プランを判別します。

データ・ソース・オブジェクトのニックネームを登録すると、フェデレーテッド・サーバーが、そのデータ・ソース・オブジェクトに関する情報を、フェデレーテッド・データベース上のシステム・カタログに追加します。照会オプティマイザーは、この情報を使用して、データ・ソース・オブジェクトからどのようにデータを取得するかについて計画を立てます。フェデレーテッド・データベースは、データ・ソース・オブジェクトの変更を自動的に検出しないため、システム・カタログ内の情報は古くなる可能性があります。

リモート・データ・ソースから現在入手可能なデータベース・ニックネーム、列、および索引についての統計情報を取ることができます。通常、リモート表で新しい統計情報を収集するときに、DB2 コマンド行プロセッサーを使用してニックネームの統計情報を手動で更新できます。

以下のリレーショナル・データ・ソースのニックネーム統計情報が取り出せます。

- DB2 ファミリー (DRDA)
- Informix
- JDBC
- Microsoft SQL Server
- ODBC
- Oracle (NET8)
- Sybase (CTLIB)
- Teradata

以下の非リレーショナル・データ・ソースのニックネーム統計情報が取り出せます。

- BioRS
- Excel
- 表構造ファイル
- ルート・ニックネーム上の XML

データ・ソースごとに収集される統計は異なります。

統計収集のためにカタログ・ベースの方法を使用して、以下の統計情報を更新できます。

- CARD
- FPAGES

- NPAGES
- OVERFLOW
- COLCARD
- HIGH2KEY
- LOW2KEY
- NLEAF
- NLEVELS
- CLUSTERFACTOR
- CLUSTERRATIO
- FULLKEYCARD
- FIRSTKEYCARD

統計収集のためにデータ・ベースの方法を使用して、以下の統計情報を更新できます。

- CARD
- COLCARD
- HIGH2KEY
- LOW2KEY
- FULLKEYCARD
- FIRSTKEYCARD

特定のサーバー定義についてのスキーマ内の単一またはすべてのニックネームの統計情報を取り出すことができます。また、Oracle Label Security を使用しているデータ・ソース・オブジェクトのニックネームの統計情報を取り出すこともできます。データベースが制限されている場合、適切なアクセス権を持つユーザーだけが HIGH2KEY および LOW2KEY 統計情報を表示することができます。制限されていないデータベースでは、Oracle Label Security を使用しているオブジェクトのニックネーム統計情報が公開され、セキュリティー・リスクが生じる場合があります。このような場合、機密情報が含まれるシステム・カタログへのアクセスを制限することができます。情報を取得する途中で障害が発生した場合、データベースは変更をロールバックします。

ニックネーム統計情報は、フェデレーテッド・サーバーをサポートする任意のオペレーティング・システムで取り出すことができます。

ニックネーム統計情報を取り出す方法

統計収集に使用する方法を選択することや、選択項目を特定の列および索引に限定することができます。カタログ・ベースの方法の方がパフォーマンスは良くなります。一方、データ・ベースの方法ではより最新の統計情報を得られますが、実行にかかる時間は長くなります。

統計収集のための以下の方法のいずれかを選択することができます。

- カタログ・ベースの方法

カタログ・ベースの方法では、統計情報をデータ・ソース・カタログ表からフェデレーテッド・カタログ表にコピーします。意味的にフェデレーテッド統計情報にマップできる統計情報のみがコピーされます。ただし、ニックネーム統計情報の正確さや新しさは、現在リモート・ソースのカタログにある情報と同程度に過ぎません。統計情報が古くなると、収集されるニックネーム統計情報も古くなります。カタログ・ベースの方法を使用する際、リモート・ソースの統計情報が最新であることを確認してください。

統計情報がリモート・ソース・カタログからフェデレーテッド・サーバーのカタログにコピーされるため、カタログ・ベースの方法の統計収集は一般的に非常に高速です。

- データ・ベースの方法

  データ・ベースの方法は、リモート・ソースの統計情報には依存していません。この方法は、ニックネームに対して発行する照会の結果から経験に基づいて独自の統計情報を生成します。この方法では、収集される統計情報はリモート・データを正確に反映します。

  データ・ベースの方法は、関係するニックネームの行サイズが大きい場合には遅くなる場合があります。通常、照会には大量データのソートおよび集約が含まれます。このため、カタログ・ベースの方法では満足のいく統計情報が得られない場合にのみ、データ・ベースの統計収集を選択してください。

収集される統計情報の品質を犠牲にしてでもデータ・ベースの方法のパフォーマンスを向上させたい場合には、統計収集を、利点の最も大きいタイプの列および索引に限定してください。これらのタイプの列には、述部、結合キー、またはグループ化操作に関係した列、または 1 つ以上の索引の一部である列が含まれます。

カタログ・ベースの方法では、統計収集を特定の列または索引に限定する必要は通常ありません。この収集方法のオーバーヘッドが非常に低いためです。

ニックネーム統計情報の取り出し

ニックネームの統計情報を取り出して、照会オプティマイザーがデータ・ソースで入手可能なニックネームに関する情報を確実に使用するようにできます。単一のニックネーム、複数のニックネーム、またはすべてのニックネームのニックネーム統計情報を更新できます。

このタスクについて

照会オプティマイザーは、ラベル・ベースのアクセス制御または Oracle Label Security を使用するデータ・ソース・オブジェクトの統計情報を収集します (HIGH2KEY および LOW2KEY ニックネーム統計情報を含む)。これらのタイプのセキュリティーを使用するデータベースの場合、適切なアクセス・レベルを持つユーザーのみが統計情報を参照できます。制限されていないデータベースの場合、アクセスを制限すること、あるいは HIGH2KEY および LOW2KEY ニックネーム統計情報を空または無意味な値に設定することができます。

デフォルトでは、統計情報の収集はニックネームのすべての列およびすべての索引に対して試行されます。統計情報を特定の列および索引に制限し、ログ・ファイルを指定することができます。

始める前に

以下の前提条件は、コマンド行プロンプトを使用して統計情報を更新する場合に当てはまります。

- フェデレーテッド・サーバーは、サーバー上にログ・ファイルを作成します。ユーザーがパスにリストするディレクトリーは、既存のものでなければなりません。
- フェデレーテッド・インスタンスの fenced ユーザー ID の特権には、指定された場所にログ・ファイルを作成する特権が含まれていなければなりません。

制約事項

フェデレーテッド・データベースへの接続に使用するユーザー ID は、リモート・データ・ソース表にマップされなければなりません。

ローカル列名またはタイプが、リモート列名またはタイプからのデフォルトのタイプ・マッピングを表していない場合、ニックネーム統計情報の更新ユーティリティーは列の統計情報を取り出しません。

手順

DB2 コマンド行、またはアプリケーション内からニックネーム統計を更新するには、ストアード・プロシージャー SYSPROC.NNSTATS を呼び出します。

ニックネーム統計の取り出し - 例

以下の例は、SYSPROC.NNSTAT ストアード・プロシージャーを使用してコマンド行からニックネーム統計情報を取り出す方法を示しています。

例: すべての統計情報の取り出し

フェデレーテッド・サーバーは、 DB2SERV サーバー上のニックネームの統計情報を取り出しますが、ログは作成しません。

```
CALL SYSPROC.NNSTAT('DB2SERV',NULL,NULL,NULL,NULL,0,NULL,?)
```

スキーマのすべての統計情報を取り出し、ログを戻す

例: カタログ・ベースの方法での統計情報の取り出し

フェデレーテッド・サーバーは、ADMIN スキーマ内のニックネーム STAFF の統計情報を取り出します。統計情報は列 1 から 5 および索引 1 と 2 について収集されます。カタログ・ベースの方法を使用して統計情報を収集します。フェデレーテッド・サーバーは、/home/iiuser/reportlogs/log1.txt ファイルにログを書き込みます。

```
CALL SYSPROC.NNSTAT(
 NULL, 'ADMIN','STAFF','COL1, COL2, COL3, COL4, COL5','IND1, IND2',1,
  '/home/iiuser/reportlogs/log1.txt',?)
```

次の例では、フェデレーテッド・サーバーは、 admin スキーマ内の DB2Serv サーバー上のすべてのニックネームの統計情報を取り出します。フェデレーテッド・サーバーは、/home/iiuser/stats/recent.log ファイルにログを書き込みます。

```
CALL SYSPROC.NNSTAT(
 'DB2Serv', 'admin', NULL, NULL, NULL, 0, '/home/iiuser/stats/recent.log', ?)
```

HIGH2KEY および LOW2KEY 統計情報に関する制限

DB2 ニックネームの HIGH2KEY および LOW2KEY 統計情報を収集する場合、一部の列の情報は収集されません。

このタスクについて

制約事項

ラッパーは、リモートの表またはニックネームについてニックネームを作成する場合、列カーディナリティーが 3 より大きい数値列についてのみ、HIGH2KEY および LOW2KEY 統計情報を収集します。

手順

HIGH2KEY および LOW2KEY 統計情報は、以下のいずれかの方法で収集します。

手順

- METHOD パラメーターを 2 に設定して SYSPROC.NNSTAT を使用する方法。この設定は、データ・ベースの統計情報の収集を指定します。データ・ベースの方法は、リモート表からのデータを照会して、ローカル統計情報の値を計算します。ただし、この方法は、リモート・サーバーおよびフェデレーテッド・サーバーのかなりの量のリソースを使用する可能性があります。
- SQL UPDATE ステートメントを発行し、SYSSTAT.COLUMNS ビューの HIGH2KEY および LOW2KEY 列を更新する方法。この場合、HIGH2KEY および LOW2KEY の正しい値を手動で判別する必要があります。

DB2 ツール・カタログの作成

ニックネームの統計情報を更新する場合、DB2 ツール・カタログを使用して、それ以降のニックネーム統計情報の更新をスケジューリングできます。 DB2 ツール・カタログがない場合は、カタログの作成を促すプロンプトが表示されます。コマンド行プロンプトから DB2 ツール・カタログを作成できます。

始める前に

DB2 管理サーバーがインストールされていなければなりません。

手順

CREATE TOOLS カタログ・コマンドを発行します。
DB2 ツール・カタログ用のデータベースを作成するシステムを指定します。

次のタスク

データベースは、メタデータ・ストレージを持たないカタログ済みのシステム上になければなりません。望んでいるシステムがカタログされていない場合は、 DB2 ツール・カタログ用のデータベースを作成する前に、そのシステムをカタログする必要があります。

ニックネーム統計情報の更新状況の表示

ニックネームの統計情報の更新を要求した後に、更新状況を表示できます。

このタスクについて

手順

DB2 コマンド行からニックネーム統計情報の更新の状況を表示するには、SYSPROC.FED_STATS 表を見ます。

以下の例は、SYSPROC.FED_STATS 表の記述を示しています。(例を単純化するため、列の長さを実際より短くしています。)

```
db2 describe table sysproc.fed_stats

Column Type Type
name                           schema    name                Length   Scale Nulls
------------------------------ --------- ------------------ -------- ----- ------
SERVER                         SYSIBM    VARCHAR              128     0     Yes
SCHEMA                         SYSIBM    VARCHAR              128     0     Yes
NICKNAME                       SYSIBM    VARCHAR              128     0     Yes
STATS_UPDATE_TIME              SYSIBM    TIMESTAMP             10     0     No
LOG_FILE_PATH                  SYSIBM    VARCHAR             1000     0     Yes
SQLCODE                        SYSIBM    INTEGER                4     0     Yes
SQLSTATE                       SYSIBM    CHARACTER              5     0     Yes
STATUS                         SYSIBM    VARCHAR             1000     0     Yes
8 record(s) selected.


db2 "select * from sysproc.fed_stats"

SERVER  SCHEMA  NICKNAME STATS_UPDATE_TIME           LOG_FILE_PATH  SQLCODE
------  ------- -------- -------------------------  -------------  ----------
ORA8    HAROLDL NICK1    2006-05-02-12.03.24.927112  -             1791 42704

SQLSTATE  STATUS
--------  ------------------------------------------------------------------------
SQL1791N  The specified server definition, schema, or nickname does not exist.

1 record(s) selected.

.
```

SYSPROC.NNSTAT ストアード・プロシージャー

1 つ以上のニックネームについての現在入手可能な統計情報を取り出します。統計情報は、フェデレーテッド・データベースにあるシステム・カタログに保管されます。

許可

SYSPROC.NNSTAT は fenced プロシージャーです。フェデレーテッド・インスタンスの fenced ユーザー ID の特権には、指定された場所にログ・ファイルを作成する特権が含まれていなければなりません。

構文

```
CALL SYSPROC.NNSTAT(
                    SERVER          VARCHAR(128)
                    SCHEMA          VARCHAR(128)
                    NICKNAME        VARCHAR(128)
```

```
COLNAMES          CLOB(2M)
INDEXNAMES        CLOB(2M)
METHOD            SMALLINT
LOG_FILE_PATH     VARCHAR(1000)
OUT_TRACE         VARCHAR(2000)
UPDATE_METHOD     SMALLINT
)
```

パラメーター

Server フェデレーテッド・サーバーがニックネーム統計情報を収集するサーバー。このサーバーは、ユーザーがフェデレーテッド・データベース内にデータ・ソースを定義するために登録するサーバーです。 1 つのニックネームを指定する場合、このパラメーターには NULL を指定することができます。

Schema

NULL が指定されている場合、フェデレーテッド・サーバーは、指定されたサーバーの下のすべてのニックネームを取り出します。 Server パラメーターが NULL の場合は、フェデレーテッド・サーバーは、指定されたスキーマの下のニックネームの統計情報を取り出します。 Schema パラメーターと Nickname パラメーターが NULL で、サーバーを指定している場合、フェデレーテッド・サーバーは、指定されたサーバーの統計情報を取り出します。

Nickname

ニックネームの名前。ニックネームを指定する場合は、スキーマも指定する必要があります。

Colnames

列名 ID として指定される列の名前。

単一ニックネームのみにこのパラメーターを指定することができます。列名を指定する場合は、スキーマおよびニックネームも指定する必要があります。

NULL を指定すると、すべての列の統計情報が収集されます。NULL がデフォルトです。

指定する列のみが処理されます。空ストリング (") を指定する場合、列は処理されません。

Indexnames

索引名 ID として指定される索引の名前。

単一ニックネームのみにこのパラメーターを指定することができます。索引名を指定する場合は、スキーマおよびニックネームも指定する必要があります。指定する索引のみが処理されます。

NULL を指定すると、すべての索引の統計情報が収集されます。NULL がデフォルトです。

空ストリング (") を指定する場合、索引は処理されません。

Method

データ・ソースから統計情報を収集する方法。

0 または NULL

カタログ・ベースの方法が最初に使用されます。この方法が失敗した場合、次にデータ・ベースの方法が使用されます。これはデフォルトです。

1

カタログ・ベースの統計情報の収集。カタログ・ベースの方法は、リモート・カタログからニックネームのローカル統計情報に情報をマップします。この方法は、リレーショナル・ニックネームにのみ有効です。

2

データ・ベースの統計情報の収集。データ・ベースの方法は、リモート表からのデータを照会して、ローカル統計情報の値を計算します。この方法は、リレーショナル・ニックネームと非リレーショナル・ニックネームの両方に有効です。

この方法は、特定のニックネームでカタログ・ベースの方法が失敗した場合、リレーショナル・ニックネームのデフォルトとなります。一般に、統計情報が収集できない理由は、ニックネームがリモート・ビューに定義されていることにあります。この場合、統計情報はリモート・ソースで使用できません。

列統計 HIGH2KEY および LOW2KEY に関して、非 Unicode データベースのグラフィック列に対するニックネーム統計の収集はサポートされていません。 DB2 は、非 Unicode データベースにおけるグラフィック・データ・タイプから文字データ・タイプへの変換をサポートしていません。そのため、方法 2 を使ってグラフィック値を HIGH2KEY および LOW2KEY VARCHAR2 カタログ列に保管することはできません。

Log_File_Path

ログ・ファイルのパス名とファイル名。フェデレーテッド・サーバーは、サーバー上にログ・ファイルを作成します。ユーザーがパスにリストするディレクトリーは、既存のものでなければなりません。 Windows では、2 つの円記号を使用してログ・パスを指定します。例えば、c:¥¥temp¥¥nnstat.log です。 NULL を指定している場合、フェデレーテッド・サーバーはログを作成しません。

Out_Trace

トレース。

Update_Method

カタログ統計を更新するためのメソッド。

0 または NULL

すべての統計は一緒に更新されます。こちらの方が速いメソッドになります。ただし、1 つのニックネームの統計が無効な場合は、すべてのニックネームの統計に影響し、カタログ統計は正常に更新されません。

1

統計はニックネームごとに更新されます。1 つのニックネームの統計が無効な場合、そのニックネームの統計はデフォルト統計としてカタログに残ります。他のすべてのニックネームの統計は、別々に更新されます。こちらのメソッドの方が遅くなります。

例 1: 次の例では、フェデレーテッド・サーバーは、ADMIN スキーマ内のニックネーム STAFF の統計情報を取り出します。統計情報は列 1、3、4、6、7、および 10 と、索引 1 から 3 について収集されます。データ・ベースの方法を使用して統計情報を収集します。フェデレーテッド・サーバーは、/home/iiuser/reportlogs/log1.txt ファイルにログを書き込みます。統計はニックネームごとに更新されます。

```
CALL SYSPROC.NNSTAT(
    NULL, 'ADMIN','STAFF','COL1, COL3, COL4, COL6, COL7, COL10',
    'IND1, IND2, IND3',2,'/home/iiuser/reportlogs/log1.txt',?,1)
```

例 2: 次の例では、フェデレーテッド・サーバーは、ADMIN スキーマ内の DB2SERV サーバー上のすべてのニックネームの統計情報を取り出します。フェデレーテッド・サーバーは、c:¥¥reports¥¥log1.txt ファイルにログを書き込みます。

```
CALL SYSPROC.NNSTAT(
    'DB2SERV','ADMIN',NULL,NULL,NULL,0,'c:¥¥reports¥¥log1.txt',?)
```

例 3: 次の例では、フェデレーテッド・サーバーは、 DB2SERV サーバー上のすべてのニックネームの統計情報を取り出しますが、ログは作成しません。

```
CALL SYSPROC.NNSTAT(
    'DB2SERV',NULL,NULL,NULL,NULL,0,NULL,?)
```

ニックネーム統計情報の自動更新

自動統計収集は、表とニックネームの最新の統計情報を収集するフィーチャーです。デフォルトでは、このフィーチャーは有効になっています。

始める前に

始める前に

データ・ソースに対して自動統計収集を実行する場合、そのデータ・ソースに接続するには、インスタンス所有者に関するユーザー・マッピングが定義されているか確認してください。

このタスクについて

このタスクについて

自動統計収集は、データベース統計情報を更新する必要がある時点を判別する自動表保守フィーチャーの一部です。ニックネームの場合、自動統計収集は、ニックネーム統計情報 (NNSTAT) ストアード・プロシージャーのカタログ・ベースのメソッドを使用します。カタログ・ベースのメソッドは、リモート・サイトのカタログ情報からニックネーム統計情報を取り出します。

自動統計収集によって考慮される表とニックネームをカスタマイズできます。例えば、自動統計収集フィーチャーによって考慮される表をカスタマイズして、すべてのニックネームや特定のサーバー上のニックネームを明確に組み込んだり除外したりできます。

自動統計収集は、保守ポリシーによって制御されます。自動 RUNSTATS とニックネームに関する統計情報には同じポリシーが使用されます。ポリシーを作成または更新するには、サンプル・ファイルおよび例をカスタマイズして使用することができます。

ニックネーム統計情報を自動収集しない場合は、自動統計収集フィーチャーを無効にするか、ポリシーを指定してデフォルトのポリシーを置換することができます。

手順

表とニックネームに関する自動統計収集のデフォルトの動作を変更するには、以下のようにします。

手順

- オプション: 自動表 RUNSTATS 操作に関するポリシーを作成します。 システムのストアード・プロシージャー SYSPROC.AUTOMAINT_SET_POLICY および SYSPROC.AUTOMAINT_SET_POLICYFILE を使用します。
- オプション: 自動表 RUNSTATS 操作に関する自動保守ポリシー情報を収集します。 システムのストアード・プロシージャー SYSPROC.AUTOMAINT_GET_POLICY および SYSPROC.AUTOMAINT_GET_POLICYFILE を使用します。
- オプション: AUTO_MAINT、AUTO_TBL_MAINT、および AUTO_RUNSTATS 構成パラメーターの値を OFF に設定して、自動統計収集を無効にします。

第 13 章 フェデレーテッド 3 部構成の名前

フェデレーテッド 3 部構成の名前により、リモート・オブジェクトを直接参照することができます。リモート・オブジェクトを直接参照する場合は、リモート・オブジェクトのニックネームを作成する必要はありません。

フェデレーテッド 3 部構成の名前は、フェデレーテッド・サーバーが大量のリモート表にアクセスする環境で非常に役立ちます。前のリリースでは、サーバーのセットアップに加え、アクセスするすべての表のニックネームを作成する必要がありました。バージョン 10.1 では、フェデレーテッド 3 部構成の名前により、サーバー・セット内でアクセスを許可されているどのデータに対しても SQL ステートメントを発行することができます。フェデレーテッド 3 部構成の名前を使用する場合、ニックネームの作成は必要なくなりました。

フェデレーテッド 3 部構成の名前で参照されるリモート・オブジェクトのメタデータは、実行時に取り出されます。このフィーチャーにより、ローカルに保管されているメタデータとリモート・サーバーとの同期化が改善されます。

フェデレーテッド 3 部構成の表名は、すべてのリレーショナル・データ・ソースでサポートされています。

フェデレーテッド 3 部構成の名前に関するサポートと考慮事項

フェデレーテッド 3 部構成の名前を処理する場合は、サポートされている機能と環境、およびいつフェデレーテッド 3 部構成の名前またはニックネームを使用するかを認識している必要があります。

フェデレーテッド 3 部構成の名前のサポート

フェデレーテッド 3 部構成の名前は、以下の環境のすべてのリレーショナル・データ・ソースでサポートされています。

- DB2 pureScale
- 超並列処理 (MPP)
- シリアル・モード

以下の SQL ステートメントは、リレーショナル・データ・ソースに指定されたフェデレーテッド 3 部構成の表名をサポートしています。

- SELECT
- INSERT、UPDATE、DELETE
- 位置付け更新および位置付け削除
- CREATE VIEW
- MERGE (リモート表をターゲット表にすることはできません)
- UNION、INTERSECT

これらの SQL ステートメントは、エクスポート・ユーティリティーでサポートされます。

考慮事項: いつフェデレーテッド 3 部構成の名前またはニックネームを使用するか

以下の機能の場合は、フェデレーテッド 3 部構成の名前ではなくニックネームを使用する必要があります。

- 非リレーショナル・データ・ソースのリモート・オブジェクトの参照

  フェデレーテッド 3 部構成の名前は、リレーショナル・データ・ソースでのみサポートされています。
- インフォメーショナル制約の定義と索引の作成

  これらの機能は、フェデレーテッド 3 部構成の名前ではサポートされていません。
- オブジェクト・レベルでの許可および特権の制御。

  フェデレーテッド 3 部構成の名前の特権は制御できません。
- 統計情報の更新。

  フェデレーテッド 3 部構成の名前を使用してリモート表にアクセスすると、統計が収集されます。ただし、リモート・オブジェクトのメタデータがフェデレーテッド・データベース・カタログに入っていないため、統計を更新することはできません。

以下の状況の場合、ニックネームまたはフェデレーテッド 3 部構成の名前を使用できます。

- 列データ・タイプをデフォルト・マッピング以外のデータ・タイプに変更する場合は、ニックネームを使用する必要があります。代替方法として、フェデレーテッド 3 部構成の名前にビューを作成するか、DML ステートメントでキャスト関数を使用することができます。

フェデレーテッド 3 部構成の名前のためのフェデレーテッド・キャッシュ

フェデレーテッド・キャッシュは、リモート・オブジェクトのメタデータと統計を保管するカタログ・キャッシュです。

フェデレーテッド 3 部構成の名前によって参照されるリモートの表またはビューに対して SQL ステートメントを発行すると、最初の参照時にそのリモート・オブジェクトのメタデータと統計がフェデレーテッド・キャッシュに保管されます。同一セッションまたは他のセッションで同じ 3 部構成の名前を参照する後続の照会は、フェデレーテッド・キャッシュから直接メタデータを取得します。

キャッシュ項目内のメタデータ情報は、有効期限期間に関連付けられます。有効期限期間は、メタデータがリモート・サーバーからロードされて以降、有効かつ最新であると見なされるインターバルです。有効期限期間のデフォルト値はゼロ (0) です。ゼロの値は、明示的にデータの無効化を選択しない限り、そのメタデータが常に有効であることを意味します。

インターバルの値は、環境変数 FEDCACHE_EXPIRE_INTERVAL を設定することにより変更できます。この値は秒単位で指定します。

推奨: db2dj.ini ファイルに FEDCACHE_EXPIRE_INTERVAL 変数を設定してください。

例

次のコマンドを実行することにより、キャッシュをフラッシュして、キャッシュされている項目を最新状態に保つことができます。

- 特定の 3 部構成の名前のメタデータを無効化するには、次のコマンドを実行してください。

  ```
  FLUSH FEDERATED CACHE FOR rudb.rschema.t1
  ```

- rschema という名前の特定のスキーマ内のすべての 3 部構成の名前のメタデータを無効化するには、次のコマンドを実行してください。

  ```
  FLUSH FEDERATED CACHE FOR rudb.rschema.*
  ```

- rudb という名前の特定のサーバー内のすべての 3 部構成の名前のメタデータを無効化するには、次のコマンドを実行してください。

  ```
  FLUSH FEDERATED CACHE FOR rudb.*.*
  ```

- rudb という名前のサーバーをフラッシュするには、次のコマンドを実行してください。

  ```
  FLUSH FEDERATED CACHE FOR SERVER rudb
  ```

フェデレーテッド 3 部構成の名前の指定

フェデレーテッド 3 部構成の名前は、リモート・オブジェクトを表します。 3 部構成の名前は、サーバー名、リモート・スキーマ名、およびリモート・オブジェクト名で構成されています。

説明

バージョン 10.1 では、フェデレーテッド 3 部構成の名前は、前のリリースでサポートされていた 3 部構成の名前と異なります。前のリリースでは、3 部構成の名前は、*db.schema.table* のフォーマットのデータベース名、スキーマ名、および表 ID からなる表名を参照していました。

フェデレーテッド 3 部構成の名前は、以下のエレメントで構成されています。

server_name
CREATE SERVER ステートメントでデータ・ソース・サーバーに割り当てられるサーバー名。

remote_schema
オブジェクトが属しているリモート・スキーマ名。

object_name
アクセスするリモート・オブジェクトのオブジェクト名。

フェデレーテッド 3 部構成の名前は、次のフォーマットで指定します:
server_name.remote_schema.remote_object

代わりに、以下を指定することもできます。

remote_schema.remote.table@server_name.

この構文を指定するには、DB2_COMPATIBILITY_VECTOR レジストリー変数をビット 0x20000 または ORA に設定する必要があります。この変数が設定されている場合、表名、ビュー名、または列名の「@」が区切り文字として扱われます。

許可

データ・ソースで許可 ID が持つ特権には、フェデレーテッド 3 部構成の名前が表すオブジェクトからデータを選択する特権が含まれている必要があります。

フェデレーテッド 3 部構成の名前で参照されるオブジェクトのリモート・オブジェクト・メタデータは、フェデレーテッド・データベース・カタログではなく、フェデレーテッド・キャッシュに保管されます。SQL ステートメントでフェデレーテッド 3 部構成の名前を参照する場合、以前ニックネームに必要だった特権は必要ありません。

Notes®

ローカル・データベース名と同じ名前でサーバーを作成することができます。これらの名前が同等の場合は、フェデレーテッド 3 部構成の名前でサーバー名が使用されたときに、3 部構成の名前はローカル名として扱われます。このローカル名は、ローカル表を見つけるために使用されます。

- ローカル表が見つからない場合は、エラー SQL0204N が出されます。
- 3 部構成の名前の最初の部分がローカル・データベース名に一致しない場合は、この名前が既存のサーバー名に一致するかどうかを検査するために SYSCAT.SERVERS カタログが確認されます。
- このサーバー名の一致が見つからない場合は、エラー SQL0204N が出されます。

スキーマをサポートしないデータ・ソースの場合は、2 部構成の名前としてリモート・オブジェクトを指定することができます。例えば、*server.table* と指定します。2 部構成の名前 (例えば、P1.TABLE) はまず、*schema_name.object_name* としてローカル・オブジェクトに解決されます。

- 名前の解決の試みが失敗すると、フェデレーテッド・サーバーはその名前を *server_name.object_name* としてフェデレーテッド・オブジェクトに解決しようとします。
- 試行が両方とも失敗すると、エラー SQL0204N が出されます。

推奨: ローカル・スキーマ名またはローカル・データベース名と同じ名前でサーバーを作成しないでください。

例

例 1: この例では、DRDA ラッパーが作成され、REMOTE という名前のリモート・データベースに対して SUDB という名前のサーバーが 作成されました。ユーザーは、EMPLOYEE という名前のリモート表に対して照会を実行します。この表は、REMOTE というデータベース内の RSCHEMA という名前のスキーマ内にあります。EMPLOYEE 表にニックネームを作成する代わりに、フェデレーテッド 3 部構成の名前を使用して直接リモート表を照会することができます。

```
SELECT birthdate FROM sudb.rschema.employee WHERE firstname='SAM'

SELECT sudb.rschema.employee.birthdate FROM sudb.rschema.employee
WHERE sudb.rschema.employee.firstname='SAM'
```

同様に、リモート表に対して UPDATE、INSERT、および DELETE ステートメントを発行することができます。

```
UPDATE sudb.rschema.employee SET firstname='MARY'
INSERT INTO sudb.rschema.employee VALUES  ('Bob')
DELETE FROM sudb.rschema.employee
```

例 2: この例では、ニックネームを含む照会に基づいてビューが作成されます。

```
CREATE VIEW v_tpn AS (SELECT c1, c2 FROM sudb.rschema.employee)
```

例えば、c2 列がドロップされるなどによりリモート・オブジェクトのスキーマが変更された場合、このビューは有効なままになります。ただし、c2 列が参照された場合は、この変更によりエラーが発生します。例えば、c2 がリモート表の Employee からドロップされた後、次の照会が発行されます。

```
select c1 from v_tpn
```

この照会は正常に実行されます。ただし、c2 を参照する次の照会は失敗します。

```
select c1, c2 from v_ptn
```

フェデレーテッド 3 部構成の名前の制約事項

フェデレーテッド 3 部構成の名前は、リレーショナル・データ・ソースでサポートされています。非リレーショナル・データ・ソースではサポートされていません。

フェデレーテッド 3 部構成の名前には、以下の制約事項も適用されます。

- ニックネームをサポートしない機能では、フェデレーテッド 3 部構成の名前はサポートされません。
- フェデレーテッド 3 部構成の名前は、サポートされる SQL ステートメントにのみ指定できます。 181 ページの『フェデレーテッド 3 部構成の名前に関するサポートと考慮事項』を参照してください。
- 静的 SQL はサポートされません
- 統計ビューはサポートされません
- マテリアライズ照会表はサポートされません

以下のステートメントは、フェデレーテッド 3 部構成の名前をサポートしません。

- ALTER TABLE
- COMMENT
- CREATE ALIAS
- CREATE FUNCTION (SQL および外部の両方)
- CREATE INDEX
- CREATE METHOD
- CREATE PROCEDURE (SQL、外部、およびソース)
- CREATE TABLE
- CREATE TRIGGER
- CREATE VARIABLE
- GRANT
- IMPORT (ターゲット表として) LOAD

- LOCK TABLE
- RENAME
- REVOKE
- 拡張標識変数を持つステートメント

第 14 章 リモート XML データの処理

Federation では、DB2 Database for Linux, UNIX, and Windows データベースの XML データにアクセスして操作できるようにする、リモート XML データ・タイプをサポートしています。

フェデレーテッド・システムは、DB2 データベース・システムと同じ XML セマンティクスに従っています。固有の XML データ・ストアは、リレーショナル表の列として階層形式で保管される XML 文書を保管およびアクセスする機能を提供します。 XML 列を XML データ・タイプで定義します。

XML データ・タイプを含むリモート表またはビューに対して、リレーショナル・ニックネームを作成できます。 XML ラッパーを使用して、XML 文書に対する非リレーショナル・ニックネームを作成することもできます。

作成したニックネームは、XQuery 言語や SQL 言語で使用できます。 XQuery 言語とは、XML 文書を照会するための基本的なメカニズムです。 SQL 言語は、XML 列を選択する、XML データを挿入、更新、または削除するなどの基本操作を実行する場合に使用できます。 SQL と XQuery を統合し、SQL/XML 関数と述部、および XQuery 関数を使用して、既存のリレーショナル・データおよび XML データの両方に対する照会を作成することもできます。

リモート XML データのニックネームの作成 - 例

リモート XML データを処理するには、XML データを含むリモート表か、 XML 文書に対してニックネームを作成する必要があります。

例: XMLTABLE1 という名前のリモート DB2 表に対応する、リレーショナル・ニックネームを作成します。表 XMLTABLE1 には、XML データ・タイプで定義された列 XMLCOL が含まれます。

```
CREATE NICKNAME NNXML1 FOR SERVER1.SCHEMA1.XMLTABLE1;
```

例: XML ラッパーを使用して、XML 文書に非リレーショナル・ニックネームを作成します。

```
CREATE NICKNAME NNXML2
   (file_path VARCHAR(64) OPTIONS(DOCUMENT 'FILE'),
  doc XML)
FOR SERVER XML_SERVER;
```

XML データの操作 - 例

ニックネームの作成後、リモート表の XML データや、フェデレーテッド・システムの XML 文書を照会および操作することができます。フェデレーテッド・システムは、DB2 データベース・システムがサポートする XML データに対するすべてのデータ操作をサポートします。

以下の例では、ニックネーム NNXML1 を使用して XML データを処理するために使用できるさまざまな方法を示します。

SQL を使用した XML データの操作

SQL を使用すると、基本的な操作 (選択、挿入、更新、および削除) を実行できます。

例: SELECT ステートメントおよび INSERT ステートメントを使用して、ニックネーム NNXML1 の XML データに対するアクセスおよび挿入操作を行います。

以下の SELECT ステートメントで、フェデレーテッド・サーバー上の XML 文書が選択されます。

```
SELECT xmlcol FROM NNXML1;
```

以下の INSERT ステートメントで、ニックネームの XML 列にストリングが挿入されます。

```
INSERT INTO NNXML1 (xmlcol) VALUES ('<a><b>My data</b></a>');
```

SQL/XML 関数を使用した XML データの操作

SQL/XML 関数を使用すると、XML データの照会、妥当性検査、および公開を含む、一定範囲の操作を実行できます。

例: XMLVALIDATE 関数を使用して、XML インスタンス文書のスキーマ仕様から得た XML スキーマに基づいて、XML データを妥当性検査します。

```
SELECT XMLVALIDATE(xmlcol)FROM NNXML1;
```

XQuery を使用した XML データの操作

xmlcolumn 関数などの XQuery 関数を使用して、XML データを操作できます。

例: XQuery を使用して、ルート a の子であるエレメント b を、XMLCOL という名前の XML 列から取り出します。

```
xquery db2-fn:xmlcolumn('NNXML1.XMLCOL')/a/b;
```

XML スキーマに対する XML 文書の妥当性検査

XML 文書が有効であることを検証できます。 XML の妥当性検査を行うには、XML スキーマを登録し、XMLVALIDATE 関数を明示的に呼び出す必要があります。検証はオプションですが、推奨されます。

XML スキーマの登録

XML 妥当性検査には、スキーマの登録が必要です。また、スキーマの登録は、アノテーション付きスキーマ分解を実行する前に行う必要があります。

このタスクについて

XML スキーマを XML スキーマ・リポジトリー (XSR) に登録します。この登録プロセスにより、XSR オブジェクトが作成されます。

XML スキーマの登録は、フェデレーテッド・サーバーで行われます。

手順

1. ストアード・プロシージャーを使用するか、コマンド行プロセッサーを介してコマンドを使用することで、 XML スキーマ文書を XSR に登録します。
2. オプション: XML スキーマが複数のスキーマ文書から構成されている場合、XSR オブジェクトに含める追加の XML スキーマ文書を指定します。
3. XSR への登録処理を完了します。

XML 文書の妥当性検査

特定の XML 文書を妥当性検査するには、XMLVALIDATE 関数を明示的に呼び出す必要があります。

このタスクについて

通常、XML データは、挿入または更新操作時に XMLVALIDATE 関数を使用して妥当性検査します。 XMLVALIDATE 関数は、XML 値の妥当性検査を、XML スキーマまたはインスタンス文書内のスキーマ仕様から取得した XML スキーマに対して行います。

ニックネーム内の XML 列の挿入または更新を行い、XMLVALIDATE 関数を使用すると、妥当性検査はフェデレーテッド・サーバーで行われます。その後、フェデレーテッド・サーバーはデータを直列化して、データ・ソースに送信します。直列化にあたって、XMLVALIDATE によって追加されたデータ・タイプ・アノテーションは保持されません。

手順

SQL ステートメントの一部として XMLVALIDATE 関数を呼び出します。

タスクの結果

- スキーマ仕様が指定されていない XMLVALIDATE が呼び出されると、インスタンス文書の *schemaLocation* 属性に基づいてスキーマが判別されます。スキーマ情報がない場合は、エラー・メッセージが出されます。
- 登録済みのスキーマまたは特定の URI を指定した XMLVALIDATE が呼び出されると、妥当性検査はその特定のスキーマに対して実行されます。外部スキーマが指定されている場合、その外部スキーマが内部スキーマの仕様を上書きします。

例

例: MYSCHEMA.MYDOCUMENTS という名前の XML スキーマを使用して XML 文書を妥当性検査します。

```
INSERT INTO NNXML1(XMLCOL)
  VALUES (
   XMLVALIDATE(
    ? ACCORDING TO XMLSCHEMA ID MYSCHEMA.MYDOCUMENTS
  )
)
```

XML リポジトリーに、MYSCHEMA.MYDOCUMENTS という SQL ID と関連付けられた XML スキーマがある場合、入力された XML 値は妥当性検査されます。

アノテーション付き XML スキーマを使用した XML 文書のニックネームへの分解

アノテーション付き XML スキーマ分解では、XML スキーマに指定された注釈に基づいて文書を分解できます。

このタスクについて

アノテーション付きスキーマ文書は、XML スキーマ・リポジトリー (XSR) に保管し、登録する必要があります。

XML データに、それ自体の階層形式でではなく、リレーショナル・データとしてアクセスする必要がある場合、その XML データを分解 (断片化) することができます。 XML 文書は、リモート表を参照するニックネームに分解することができます。

手順

1. XML スキーマ分解で、スキーマ文書にアノテーションを付けます。
2. 分解に使用するニックネームを作成します。このニックネームは、アノテーション付きスキーマに指定された値と一致しなければなりません。
3. REGISTER XMLSCHEMA コマンドを使用して、スキーマを登録します。
4. XSR オブジェクトに USAGE 特権を付与して、特定のユーザーが特定の XML スキーマを使用できるようにします。
5. ALTER XSROBJECT ステートメントを使用して、スキーマを分解可能にします。
6. DECOMPOSE XML DOCUMENT コマンドを使用して、XML インスタンス文書を分解します。

リモート XML データのフェデレーテッド処理

XML データは、フェデレーテッド・システムとリモート・データ・ソース・クライアントとの間で、文字フォーマットまたはバイナリー・フォーマットの直列化された XML ストリングとして送受信されます。

フェデレーテッド・サーバーが XML データを送受信する方法に影響を与えるプロセスは、直列化、構文解析、およびコード・ページ規則です。

リモート XML データのフェデレーテッド構文解析

フェデレーテッド・システムでは、DB2 XML パーサーを使用してリモート XML データを処理します。

パーサーは、データ・エンコード規則に準拠した整形式の XML データを必要とします。パーサーは、スキーマの妥当性検査は実行しません。

- データの取り出しの際に XML データが整形式でないと、パーサーはエラー・メッセージを出します。
- データの挿入の際に、XML データが整形式でないと、構文解析の行われる場所に応じて、フェデレーテッド・サーバーまたはターゲット・データ・ソースがエラー・メッセージを出します。

Federation での空白文字の処理

XML の構文解析の際に、XML 文書内の境界空白文字を保持することも、取り除くこともできます。*境界空白文字* とは、エレメント間に出現する空白文字のことです。

フェデレーテッド・サーバーでは、DB2 データベース・システムと同じ空白文字規則が適用されます。

アプリケーション・ホスト変数およびパラメーター・マーカーについては、フェデレーテッド・バインド操作の際に空白文字を取り除くかどうかを特殊レジスター CURRENT IMPLICIT XMLPARSE OPTION が決定します。データ・ソースが異なる規則に従って空白文字を処理している場合、フェデレーテッド・システムはセマンティクスの相違の補正を試みます。

リモート XML データのコード・ページの問題

特定のコード・ページの問題が、フェデレーテッド・ステートメントに影響を与える可能性があります。

フェデレーテッド・サーバーは、DB2 XML パーサーのエンコード規則に準拠しています。

フェデレーテッド・ステートメントでは、直列化されたリモート XML データの操作中に、以下のような問題が発生する可能性があります。

- バイナリー・フォーマットで直列化されたリモート XML データの場合:
  - リモート・クライアントからフェデレーテッド・ラッパーへのデータの送信時に、データ損失は起こりません。
  - 有効な IBM エンコード・スキームではない内部エンコード方式がある場合、ラッパーはそのエンコード属性を有効な IBM エンコード・スキームと置き換えるか、XML 宣言からそのエンコード属性を除去して DB2 XML パーサーで値をデコードします。
- 文字フォーマットで直列化されたリモート XML データの場合:
  - XML データのコード・ページは、フェデレーテッド・データベースのコード・ページ内にあります。データ・ソースのクライアントからフェデレーテッド・ラッパーへのデータ送信時には、データ損失が起きる可能性があります。変換の結果として文字が置換される場合、データ・ソースの振る舞いやラッパーの実装に応じて、警告が生成されることがあります。
  - 内部エンコード方式がある場合はラッパーによって取り除かれます。これは、直列化されたストリングがフェデレーテッド・データベースのコード・ページにあるためです。

非リレーショナル・ラッパーの場合、DB2 XML パーサーが文書の内部エンコード方式を使用して XML データをデコードします。

リモート XML データ・タイプに関する制限

フェデレーテッド・システムでは、リモート XML データ・タイプにいくつかの制限があります。

リモート XML データ・タイプは、DB2 for Linux, UNIX, and Windows データ・ソースおよび XML ラッパーでのみ使用できます。

以下の SQL 操作は実行できません。

- ニックネームの XML データ・タイプを変更する。これは、エラー SQL0270N になります。
- xml-index-specification 節を使用して、フェデレーテッド索引 SPECIFICATION ONLY を作成する。これは、エラー SQL0104N になります。
- フェデレーテッド・ストアード・プロシージャーを XML データ・タイプで定義する。これは、エラー SQL1254N になります。
- XML データ・タイプと別のデータ・タイプとの間のタイプ・マッピングを作成する。これは、エラー SQL0604N になります。

以下の制限は、XML ラッパーの XML 列に適用されます。

- XML 列には列オプションを指定できません。
- XML 文書の全コンテンツを参照する単一の XML 列は、ルートのニックネームだけに含めることができます。
- ルートのニックネームは、それぞれ単一の XML 文書に対応していなければなりません。
- 子ニックネーム、ニックネームの XPath オプション、およびニックネームのネーム・スペース・オプションは、関係がありません。

第 15 章 ネストされた表の式におけるエラー耐性

エラー耐性とは、ネストされた表の式における特定の SQL エラーを許容しながら照会実行の継続を許可する機構です。エラー耐性があれば、照会の一部に関するエラーを受け取って照会全体が終了してしまうのではなく、使用可能なデータから最大限の照会結果を得ることができます。

フェデレーテッド・サーバーは、許容可能なエラーを見つけると、エラーを許容し、照会全体を対象にエラーを戻すのではなく、残りの照会の処理を継続します。フェデレーテッド・サーバーが戻す結果セットは、部分的または空の結果となる場合があります。

フェデレーテッド・サーバーがエラーを許容すると、照会がアクセスするデータ・ソースが使用できない場合であっても照会結果を戻します。この機構は、不完全な照会結果であろうと、入手できるなるべく多くの情報を戻す必要がある場合に便利です。例えば、特定のタイプの病状に関するデータを必要とする医師について考えます。いくつかの異なる病院にあるリモート・データ・ソースから情報を収集するために照会を実行します。 1 つ以上の病院のデータベースが使用できない場合、使用可能なデータベースからの結果だけでも医師にとっては非常に価値があります。

UNION ALL ブランチが含まれる照会は、エラー耐性が役立ちます。この機構がないと、照会の 1 つのブランチの処理でエラーが発生した場合、フェデレーテッド・サーバーは照会の処理を停止してしまいます。この機構があれば、照会のその同じブランチで許容するエラーを指定した場合、フェデレーテッド・サーバーはそのエラーを許容し、残りの使用可能なブランチへのナビゲートを継続します。 UNION ALL 操作は、使用可能なあらゆるデータ・ソースからの結果を戻します。

例: 以下の照会は、3 つの異なるサーバーの 3 つのニックネームからデータを選択します。

```
SELECT c1 from nickname1_on_server1
UNION ALL
SELECT c1 from nickname2_on_server2
UNION ALL
SELECT c1 from nickname3_on_server3
```

nickname2_on_server2 が使用できないか、照会処理中にリモート・サーバー server2 が使用できない場合、nickname2 および server2 でのエラーを許容することにより、nickname1_on_server1 および nickname3_on_server3 からの結果セットを取得できます。 3 つの照会ブランチの中の 2 つからの結果セットは、以下の照会を実行することに相当します。

```
SELECT c1 from nickname1_on_server1
UNION ALL
SELECT c1 from nickname3_on_server3
```

照会処理中に、ネストされた表の式で許可する SQL エラーを指定できます。フェデレーテッド・サーバーが許容するエラーのタイプは、リモート接続、許可、および認証でのエラーです。

エラー耐性の対象となるネストされた表の式の指定

RETURN DATA UNTIL 文節を使用して、ネストされた表の式で許容するエラーを指定します。

このタスクについて

RETURN DATA UNTIL 文節を使用する際、許容するエラー・コードを指定する必要があります。以下の表は、*specific-condition-value* 文節で許容されるエラーをリストにしています。許容されるエラー・コードと一致する SQLSTATE (または SQLSTATE と SQLCODE) を指定する必要があります。表にリストされている SQLCODE は必須です。

表 18. specific-condition-value 文節で許可されるエラー

SQLSTATE	エラー・コード	SQLCODE
08001	SQL30080N	-30080
08001	SQL30081N	-30081
08001	SQL30082N	-30082
08001	SQL1336N	-1336
08004	任意	任意
28000	任意	任意
42501	任意	任意
42512	任意	任意
42704	SQL0204N	-204
42720	任意	任意

手順

エラー耐性の対象となるネストされた表の式を指定するには、次のように RETURN DATA UNTIL 文節が含まれた SQL ステートメントを作成します。

```
RETURN DATA UNTIL specific-condition-value
```

RETURN DATA UNTIL
指定した条件が満たされる前に全選択から戻される行は、いずれも全選択から結果セットに戻されます。

specific-condition-value
エラー耐性の条件と値を指定します。

FEDERATED
必須キーワード。指定する特定の条件には、フェデレーテッド・データ・ソースで発生するエラーのみを含める必要があります。

SQLSTATE VALUE *string-constant*
特定の条件を SQLSTATE 値として指定できます。VALUE を指定する場合、ストリング定数の長さは 5 でなければなりません。SQLSTATE 値は、特定の SQLCODE 値に絞ることができます。1 つの *specific-condition-value* で同じ SQLSTATE を共有する、複数の SQLCODE 値を指定することができます。

エラー耐性の対象となるネストされた表の式の例

次の例は、RETURN DATA UNTIL 文節を使用して、1 つ以上のフェデレーテッド・データ・ソースが使用できない場合に照会結果を戻す方法を示しています。

例: 以下の SQL ステートメントは、SQLServer、Oracle、および Sybase の 3 つの異なるサーバーからデータを選択します。

```
SELECT c1 FROM
TABLE RETURN DATA UNTIL
FEDERATED SQLSTATE '08001' SQLCODE -30080, -30082
WITHIN(SELECT c1 FROM n1_from_SQLServer) etq1
UNION ALL
SELECT c1 FROM
TABLE RETURN DATA UNTIL
FEDERATED SQLSTATE '08001' SQLCODE -30080, -30082
WITHIN (SELECT c1 FROM n2_from_Oracle) etq2
UNION ALL
SELECT c1 FROM
TABLE RETURN DATA UNTIL
FEDERATED SQLSTATE '08001' SQLCODE -30080, -30082
WITHIN(SELECT c1 FROM n3_from_Sybase) etq3;
```

シナリオ 1: 1 つのサーバーが使用できない。

このシナリオでは、Oracle サーバーが使用できません。 SQLServer サーバーおよび Sybase サーバーは使用できます。この状態では、UNION ALL 操作の 2 番目のブランチにおける照会は、許容されると指定されている 30080 エラーで空の結果セットを戻します。照会は n1_from_SQLServer および n3_from_Sybase からの結果を戻します。警告 sqlwarn5='E' が出されます。

この結果セットは、以下の照会を実行することに相当します。

```
SELECT c1 FROM n1_from_SQLServer
UNION ALL
SELECT c1 FROM n3_from_Sybase;
```

シナリオ 2: すべてのサーバーが使用できない。

このシナリオでは、すべてのサーバー (SQLServer、Oracle、および Sybase) は使用できません。この状態では、UNION ALL 操作は、空の結果セットを戻します。警告 sqlwarn5='E' が出されます。

シナリオ 3: すべてのサーバーが使用できる。

すべてのサーバーが使用できる場合、照会の結果セットは、同じ照会を RETURN DATA UNTIL 文節を指定せずに実行することに相当します。

エラー耐性の対象となるネストされた表の式におけるデータ・ソース・サポート

エラー耐性は、いくつかのリレーショナル・データ・ソースおよび非リレーショナル・ニックネームでサポートされています。

ネストされた表の式におけるエラー耐性は、以下のリレーショナル・データ・ソースでサポートされています。

- DB2 ファミリー (DRDA)
- Informix
- JDBC
- Microsoft SQL Server
- ODBC
- Oracle (NET8)
- Sybase (CTLIB)
- Teradata

エラー耐性の対象となるネストされた表の式の中では、非リレーショナル・ニックネームを使用することができます。非リレーショナル・ラッパーから許容可能なエラー・コードが戻されると、フェデレーテッド・サーバーは許容可能な接続エラー、認証エラー、または許可エラーを許容できます。

エラー耐性の対象となるネストされた表の式に関する制約事項

エラー耐性のあるネストされた表の式を定義する際にはいくつかの制約事項が適用されます。

RETURN DATA UNTIL 文節が含まれる式を持つ照会またはビューを定義すると、その照会またはビューは読み取り専用になります。RETURN DATA UNTIL 文節を持つ式で宣言されたカーソルは、読み取り専用になります。それらの状態の各々に対してエラーが返されます。

接続障害が発生した場合、初期接続障害は許容されます。初期接続障害は、データ・ソースへの接続試行中に発生します。データ・ソースへの接続試行は、接続の中で、または接続損失後に、データ・ソースに対して初めて照会が発行されたときに、起きます。初期接続障害でない接続損失または接続障害は許容されません。

第 16 章 フェデレーションの高可用性

フェデレーションには、高可用性フィーチャーが備えられています。高可用性災害時リカバリー (HADR) は、ソース・データベースからターゲット・データベースにデータの変更を複製することによってデータ損失を防ぐ、データベース・レプリケーション・フィーチャーです。

ソース・データベースは、*1 次* データベースと呼ばれることもあります。ターゲット・データベースは、*スタンバイ・データベース*と呼ばれることもあります。 1 次データベースが使用できなくなるか失敗すると、スタンバイ・データベースがトランザクション処理をテークオーバーします。これによりスタンバイ・データベースが、新しい 1 次データベースになります。 1 次データベースからスタンバイ・データベースへのこのワークロードの転送を*フェイルオーバー* といいます。フェイルオーバーが発生すると、クライアントのデータベース接続が自動クライアント・リルート・プロセスを介して新しい 1 次データベースに切り替わります。

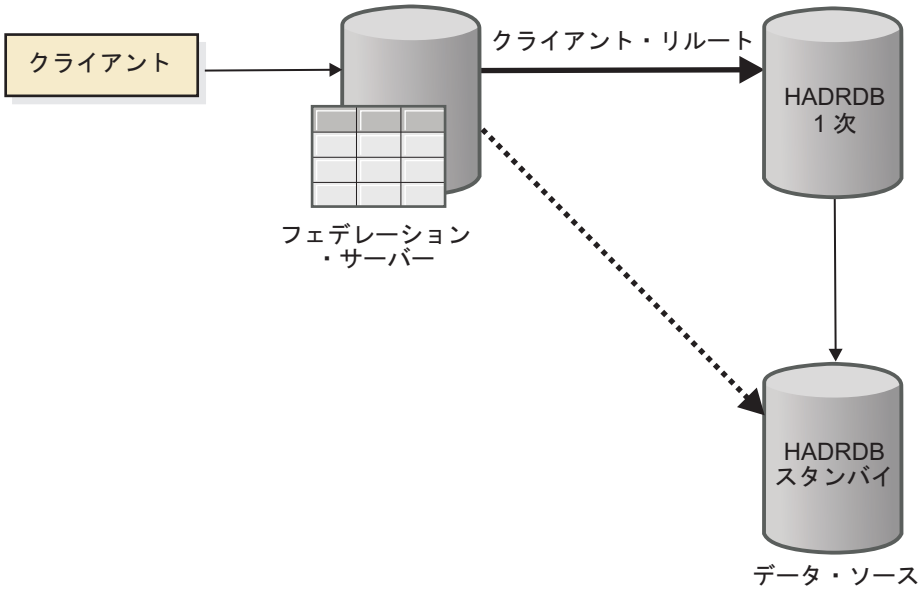
フェデレーテッド・システムでは、HADR データベースはフェデレーテッド・データベースまたはデータ・ソース・データベースのいずれかとして機能することができます。

高可用性災害時リカバリー (HADR) およびフェデレーテッド・データベース

HADR データベースは、フェデレーテッド・データベースとして機能することができます。

HADR データベースがフェデレーテッド・データベースとして機能する場合、1 次フェデレーテッド・データベースが使用できなくなったときに、スタンバイ・データベースがトランザクション処理をテークオーバーします。クライアント・リルート・プロセスがフェデレーテッド・データベースで使用可能になっている場合、クライアントはデータベース接続を自動的に新しい 1 次データベースに切り替えることができます。

この構成では、データ・ソースはフェデレーションによってサポートされるどのデータ・ソースであっても構いません。以下の図は、この構成を示しています。



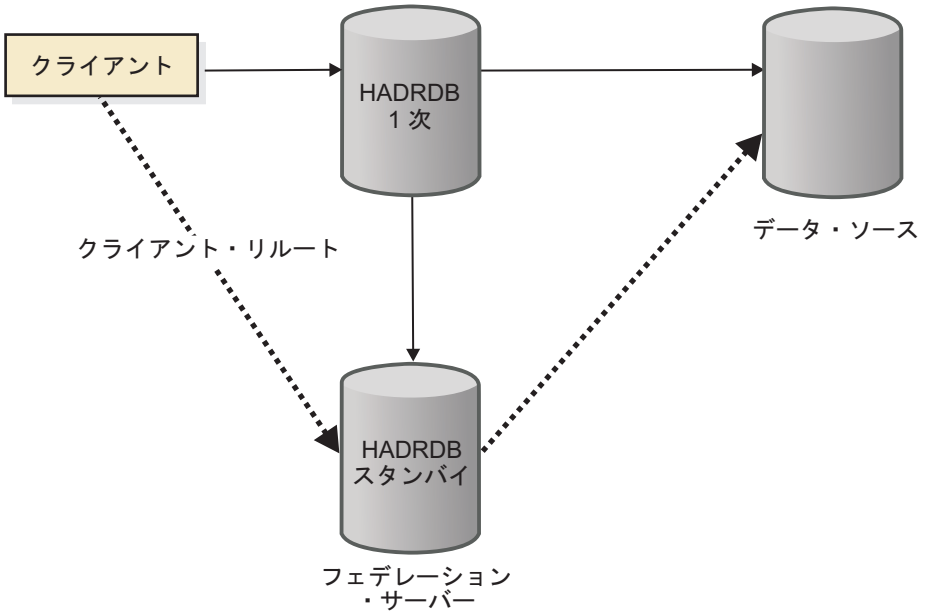
高可用性およびデータ・ソース・データベース

フェデレーションは、高可用性フィーチャーを持つデータ・ソースを処理できます。

データ・ソースが DB2 Database for Linux, UNIX, and Windows の場合、データ・ソース・データベースに対して高可用性災害時リカバリー (HADR) および自動クライアント・リルート (ACR) を構成できます。フェデレーション・サーバーは、データ・ソースのクライアントです。データ・ソースで 1 次データベースが使用できなくなった場合、スタンバイ・データベースがトランザクション処理をテークオーバーします。自動クライアント・リルート (ACR) を使用することで、フェデレーション・サーバーは自動的に新規 1 次データ・ソース・データベースに接続でき、フェデレーションがサポートする main 関数の処理を続行します。

高可用性フィーチャーを持つその他のデータ・ソースについては、フェイルオーバーの後フェデレーテッド・サーバーが自動的に新規アクティブ・データ・ソース・データベースに接続できるかどうかは、データ・ソースによって異なります。

以下の図は、この構成を示しています。



フェデレーションの高可用性災害時リカバリー (HADR) の構成

データ・ソース・データベースに対して高可用性フィーチャーを構成したり (DB2 Database for Linux, UNIX, and Windows データ・ソースの HADR など)、IBM InfoSphere Federation Server に対して高可用性災害時リカバリーを構成したりすることができます。

フェデレーテッド・データベースの HADR の構成

フェデレーテッド・データベースの HADR を構成するには、必要なステップを実行して、スタンバイ・データベースと 1 次データベースをセットアップする必要があります。

このタスクについて

フェデレーテッド・データベースで HADR を構成する際、1 次データベースとスタンバイ・データベースの間の整合性を保つことが重要です。構成中にフェデレーテッド・データベースに対する変更 (つまり、DDL または DML) を行わないでください。

高可用性データベースとして機能するフェデレーテッド・データベースでは、自動クライアント・リルートを構成することをお勧めします。

手順

1. 既存のフェデレーテッド・データベースを選択するか、フェデレーテッド・データベースを作成する手順に従ってフェデレーテッド・データベースを作成し、1 次データベース上に以下のオブジェクトをそれぞれ作成します。
   a. ラッパー。CREATE WRAPPER ステートメントを使って作成します。
   b. サーバー。CREATE SERVER ステートメントを使って作成します。

c. ユーザー。CREATE USER MAPPING ステートメントを使って作成します。

d. ニックネーム。CREATE NICKNAME ステートメントを使って作成します。

2. 1 次データベース・サーバー上でデータベースをバックアップし、スタンバイ・データベース・サーバー上でデータベースをリストアします。 HADR データベースには `logretain on` を指定する必要があります。以下に例を示します。

   1 次サーバー:

   ```
   db2 update db cfg for feddb using logretain on

   db2 backup db feddb to /directory
   ```

   スタンバイ・サーバー:

   ```
   db2 restore db feddb from /directory
   ```

3. スタンバイ・データベースの HADR を構成します。 以下に例を示します。

   ```
   db2 UPDATE DB CFG FOR feddb USING HADR_LOCAL_HOST hostnamest;
   db2 UPDATE DB CFG FOR feddb USING HADR_REMOTE_HOST hostnampr;
   db2 START HADR ON DB feddb AS STANDBY;
   ```

4. 1 次データベースの HADR を構成します。 以下に例を示します。

   ```
   db2 UPDATE DB CFG FOR feddb USING HADR_LOCAL_HOST hostnamepr;
   db2 UPDATE DB CFG FOR feddb USING HADR_REMOTE_HOST hostnamest;
   db2 START HADR ON DB feddb AS PRIMARY;
   ```

   構成の追加情報については、高可用性災害時リカバリー (HADR) 用のデータベース構成を参照してください。

次に行うこと

フェデレーテッド・データベースの HADR が構成された後、フェデレーテッド DDL または DML などの操作を発行できるようになります。以下に例を示します。

- 別のニックネームの作成
- ニックネームを使用した行の更新

1 次データベースが失敗する場合、スタンバイ・データベースに対して TAKEOVER HADR コマンドを発行することができます。これによりスタンバイ・データベースが、トランザクション処理をテークオーバーできる新しい 1 次データベースになります。

データ・ソース・データベースの高可用性の構成

高可用性データベースをデータ・ソースとして構成するには、必要なステップを実行して、スタンバイ・データベースと 1 次データベースをセットアップする必要があります。

このタスクについて

データ・ソース・データベースで高可用性を構成する際、1 次データベースとスタンバイ・データベースの間の整合性を保つことが重要です。構成中にフェデレーテッド・データベースまたはデータ・ソース・データベースに対する変更 (つまり、DDL または DML) を行わないでください。

データ・ソースが DB2 Database for Linux, UNIX, and Windows の場合、高可用性データベースとして機能するデータ・ソース・データベースに対して自動クライアント・リルートを構成する必要があります。フェデレーテッド・サーバーでは、テークオーバー後に新しい 1 次データベースへの接続を自動的に確立するために自動クライアント・リルートが必要です。

以下の手順では、DB2 Database for Linux, UNIX, and Windows データ・ソースの構成について説明しています。

手順

1. 既存のデータ・ソース・データベースを選択するか、CREATE DATABASE ステートメントを使用することにより 1 次データベース・サーバー上にデータ・ソース・データベースを作成します。 以下に例を示します。

   ```
   db2 create db ds
   create table t11 (i int, c char(20));
   insert into t11 values (11, 'dsds');
   ```

2. 1 次データベース・サーバー上でデータベースをバックアップし、スタンバイ・データベース・サーバー上でデータベースをリストアします。 高可用性データベースには `logretain on` を指定する必要があります。以下に例を示します。

   1 次サーバー:

   ```
   db2 update db cfg for ds using logretain on

   db2 backup db ds to /directory
   ```

   スタンバイ・サーバー:

   ```
   db2 restore db ds from /directory
   ```

3. スタンバイ・データベースの高可用性および自動クライアント・リルートを構成します。 以下に例を示します。

   ```
   db2 UPDATE DB CFG FOR ds USING HADR_LOCAL_HOST hostnamest;
   db2 UPDATE DB CFG FOR ds USING HADR_REMOTE_HOST hostnampr;
   db2 START HADR ON DB ds AS STANDBY;
   db2 UPDATE ALTERNATE SERVER FOR DATABASE ds USING HOSTNAME hostnamepr PORT port1
   ```

4. 1 次データベースの高可用性および自動クライアント・リルートを構成します。以下に例を示します。

   ```
   db2 UPDATE DB CFG FOR feddb USING HADR_LOCAL_HOST hostnamepr;
   db2 UPDATE DB CFG FOR feddb USING HADR_REMOTE_HOST hostnamest;
   db2 START HADR ON DB ds AS PRIMARY;
   db2 UPDATE ALTERNATE SERVER FOR DATABASE ds USING HOSTNAME hostnamest PORT port2
   ```

   高可用性の構成の追加情報については、高可用性災害時リカバリー (HADR) 用のデータベース構成を参照してください。自動クライアント・リルートの構成に関する追加情報は、自動クライアント・リルートおよび高可用性災害時リカバリー (HADR) の構成を参照してください。

次に行うこと

データ・ソース・データベースの HADR が構成された後、フェデレーテッド・データベースを作成する手順に従ってフェデレーテッド・データベースを作成

し、高可用性を持つデータ・ソース・データベースにアクセスするためにフェデレーテッド・データベース上に以下のオブジェクトをそれぞれ作成することができます。

a. ラッパー。CREATE WRAPPER ステートメントを使って作成します。

b. サーバー。CREATE SERVER ステートメントを使って作成します。

c. ユーザー。CREATE USER MAPPING ステートメントを使って作成します。

d. ニックネーム。CREATE NICKNAME ステートメントを使って作成します。

既存のフェデレーテッド・データベースのニックネームを使って、高可用性を持つデータ・ソース・データベースにアクセスすることもできます。

データ・ソース上で 1 次データベースが失敗する場合、スタンバイ・データベースに対して TAKEOVER HADR コマンドを発行することができます。これによりスタンバイ・データベースが、トランザクション処理をテークオーバーできる新しい 1 次データベースになります。フェデレーション・サーバーは自動的に、接続を新しい 1 次データベースに切り替えます。

フェデレーションの高可用性災害時リカバリー (HADR) に関する制約事項

DB2 Database for Linux, UNIX, and Windows に適用されるのと同じ高可用性に関する制約事項がフェデレーションにも適用されます。一部の制約事項については特に、HADR に関するフェデレーション・サポートに適用されます。

以下の制約事項は、フェデレーテッド・データベースとして構成される HADR データベースに適用されます。

- 透過 DDL はサポートされていません。
- スタンバイ・フィーチャーの読み取りはサポートされていません。
- スタンバイ・データベースには 1 次データベースと同じデータ・ソースのクライアントが入っている必要があります。そうなっていない場合、スタンバイ・データベースのテークオーバー後のニックネームへのアクセスが失敗します。

第 17 章 フェデレーテッド・サーバーおよびニックネームのモニター

ヘルス・モニターおよびシステム・モニター・エレメントを使用して、フェデレーテッド・システムをモニターできます。

フェデレーテッド・ニックネームおよびサーバーのヘルス・インディケーター

ヘルス・インディケーターを使用すると、ご使用のフェデレーテッド・ニックネームおよびサーバーの状況をモニターすることができます。

ニックネームのヘルス・インディケーターは db.fed_nicknames_op_status です。サーバー定義のヘルス・インディケーターは db.fed_servers_op_status です。フェデレーテッド・ヘルス・インディケーターは、ヘルス・モニターをインストールしたときにインストールされます。

ニックネームまたはサーバーの状態が正常でないときには、ヘルス・インディケーターがアラートを出します。コマンド行を使用して、モニターの結果を表示することができます。

AIX、HP-UX、Linux、Microsoft Windows、および Solaris オペレーティング・システムを使用するフェデレーテッド・サーバーは、ヘルス・インディケーターをサポートします。

表 19 で、フェデレーテッド・ニックネームおよびサーバーのヘルス・インディケーターについて説明します。

表 19. ニックネームおよびサーバーのヘルス・インディケーター

ヘルス・インディケーター	説明
db.fed_nicknames_op_status	フェデレーテッド・サーバーのデータベース内で定義されているすべてのリレーショナル・ニックネームの全体的な正常性を示します。 ニックネームが無効である場合にアラートを出します。無効なニックネームに関する詳細、および無効なニックネームを修復するために取ることができる推奨処置が示されます。
db.fed_servers_op_status	フェデレーテッド・サーバーのデータベース内で定義されているすべてのフェデレーテッド・サーバーの全体的な正常性を示します。 サーバーが使用不可である場合にアラートを出します。使用不可のサーバーに関する詳細、およびそのサーバーを使用可能にするために取ることができる推奨処置が示されます。

ヘルス・インディケーターは以下のデータ・ソースを評価できます。

- DB2 ファミリー (DRDA)
- Excel
- Informix
- JDBC
- Microsoft SQL Server
- ODBC
- Oracle (NET8)
- Sybase (CTLIB)
- 表構造ファイル
- Teradata
- XML (ルート・ニックネームのみ)

フェデレーテッド・ヘルス・インディケーターの活動化

ニックネームとサーバーの正常性をモニターするには、フェデレーテッド・ヘルス・インディケーターをアクティブにしなければなりません。ニックネームのヘルス・インディケーターは db.fed_nicknames_op_status です。サーバー定義のヘルス・インディケーターは db.fed_servers_op_status です。

手順

フェデレーテッド・ヘルス・インディケーターをアクティブにするには、コマンド・ライン・プロセッサーを使用できます。

フェデレーテッド・ニックネームおよびサーバーの正常性のモニター

ニックネームとサーバーの状況をモニターすると、フェデレーテッド・システムの問題を判別して解決するのに役立ちます。

始める前に

- ニックネームに対する SELECT 特権がフェデレーテッド・サーバー上に定義されていることが必要です。
- データベース・マネージャー構成パラメーター FEDERATED を YES に設定します。
- データ・ソースに認証が必要である場合は、データ・ソースにヘルス・モニターの ID からのユーザー・マッピングがなければなりません。

このタスクについて

コマンド行を使用して、モニターの結果を表示することができます。コマンド・ライン・プロセッサーを使用して、ヘルス・インディケーターにより特定される問題を解決してください。

制約事項

203 ページの『フェデレーテッド・ニックネームおよびサーバーのヘルス・インディケーター』 は、ヘルス・インディケーターが評価できるデータ・ソースをリストしたものです。

手順

この作業をコマンド行から行うには、GET HEALTH SNAPSHOT ステートメントを発行します。

フェデレーテッド・ニックネームおよびサーバーの正常性のモニター例

このトピックでは、データベースのヘルス・スナップショットの例を示します。

フェデレーテッド・ヘルス・インディケーター名は db.fed_nicknames_op_status と db.fed_servers_op_status です。 CLP で以下のコマンドを使用して、これらのヘルス・インディケーターを有効にする必要があります。

```
db2 update alert cfg for databases using db.fed_nicknames_op_status set
 THRESHOLDSCHECKED YES
db2 update alert cfg for databases using db.fed_servers_op_status set
 THRESHOLDSCHECKED YES
```

以下のコマンドは、フェデレーテッド・ヘルス・インディケーターを含む (有効にされている場合)、データベース・ヘルス・スナップショットを検索します。

```
db2 get health snapshot for database on <database_name>
```

この例では、データベース名は fedhi です。このコマンドの出力は、両方のヘルス・インディケーターがいずれも「正常」(normal) の状態であることを示しています。「正常」とは、ニックネームとサーバーが有効であるという意味です。

```
            Database Health Snapshot

Snapshot timestamp                          = 02/10/2006 12:10:55.063004

Database name                               = FEDHI
Database path                               = C:¥DB2¥NODE0000¥SQL00006¥
Input database alias                        = FEDHI
Operating system running at database server= NT
Location of the database                    = Local
Database highest severity alert state       = Attention

Health Indicators:

    Indicator Name                            = db.fed_servers_op_status
       Value                                  = 0
       Evaluation timestamp                   = 02/10/2006 12:09:10.961000
       Alert state                            = Normal

    Indicator Name                            = db.fed_nicknames_op_status
       Value                                  = 0
       Evaluation timestamp                   = 02/10/2006 12:09:10.961000
       Alert state                            = Normal

    Indicator Name                            = db.db_op_status
       Value                                  = 0
       Evaluation timestamp                   = 02/10/2006 12:08:10.774000
       Alert state                            = Normal

    Indicator Name                            = db.sort_shrmem_util
       Value                                  = 0
```

```
   Unit                                 = %
   Evaluation timestamp                 = 02/10/2006 12:08:10.774000
   Alert state                          = Normal


Indicator Name                          = db.spilled_sorts
   Value                                = 0
   Unit                                 = %
   Evaluation timestamp                 = 02/10/2006 12:09:10.961000
   Alert state                          = Normal
```

フェデレーテッド・システムのスナップショット・モニターの概説

特定時点のフェデレーテッド・データ・ソースおよび接続中のアプリケーションに関する情報を収集するために、スナップショット・モニターを使用できます。

フェデレーテッド・システムの状況を判別するには、スナップショットは便利です。スナップショットを定期的に取っておくことで、傾向を観察したり、起こりそうな問題を予測したりするのに役立てることができます。

スナップショット・モニターの出力は、以下の形式で使用できます。

- テキスト形式 (スナップショット・モニターのコマンド行プロセッサー・インターフェースを介して)。
- 表関数の出力。この出力は、出力を制限する照会を作成するのに役立ちます。

フェデレーテッド・ワークロードに特に役立つスナップショットには、以下のものがあります。

動的 SQL ステートメントのスナップショット
ステートメント・キャッシュ (フェデレーテッドおよび非フェデレーテッドのステートメントを含む) に現在入っているすべての動的 SQL ステートメントのスナップショットを提供します。

アプリケーション・スナップショット
現在実行中の SQL ステートメントのテキストを含む、特定のアプリケーションに関する情報を提供します。

リモート・データベース・スナップショット
特定のフェデレーテッド・システム・データベースに関する情報を提供します。

すべてのリモート・データベース・スナップショット
アクティブになっている各フェデレーテッド・システム・データベースに関する情報を提供します。

リモート・アプリケーション・スナップショット
アクティブになっている各フェデレーテッド・システム・アプリケーションごとにアプリケーション・レベルの情報を提供します。

フェデレーテッド照会のモニター

照会をモニターすることにより、フェデレーテッド・システムの実行状況を判別することができます。リモート照会のスナップショットを取ると、フェデレーテッド・システムで照会がどのように処理されているかを理解するのに役立ちます。

始める前に

フェデレーテッド・データベースでリモート照会のスナップショット情報を収集するには、STATEMENT モニター・スイッチを ON に設定する必要があります。

このタスクについて

スナップショット・モニターは、フェデレーテッド・サーバーによって処理される各照会の 2 つの側面をトラッキングします。

- アプリケーションによってサブミットされたフェデレーテッド照会全体。これは、ニックネーム、ローカル表、あるいはその両方を参照します。
- ニックネームが含まれる照会の場合には、1 つ以上のリモート・フラグメント。リモート・フラグメントとは、自動的に生成されるステートメントであり、フェデレーテッド照会の代わりに固有のダイアレクトでリモート・データ・ソースにサブミットされます。

フェデレーテッド照会をモニターする場合には、フェデレーテッド・サーバーでローカルに行われる作業と、リモート照会フラグメントに応じてリモート・サーバーで行われる作業の両方について考慮する必要があります。フェデレーテッド照会はフェデレーテッド・サーバーにサブミットされるため、動的 SQL ステートメントのスナップショットおよび SNAPSHOT_DYN_SQL 表関数には、個別のフェデレーテッド照会に関する情報が含まれます。さらに、フェデレーテッド・サーバーが他のデータ・ソースに送信するリモート照会フラグメントに関する情報も含まれます。

手順

フェデレーテッド・データベースへ接続中に、フェデレーテッド・サーバーで照会をモニターするには以下のメソッドのいずれかを使用します。

- テキスト出力:

  ```
  GET SNAPSHOT FOR DYNAMIC SQL on dbname
  ```

  ここで、*dbname* とは、フェデレーテッド・サーバー・データベースの名前です。

- 表関数:

  ```
  CREATE TABLE table snap AS (SELECT * FROM TABLE(SNAPSHOT_DYN_SQL ('dbname', -1))
   as snaptab) definition only;
  INSERT INTO snap (SELECT * FROM TABLE(SNAPSHOT_DYN_SQL ('dbname', -1))
   as snaptab);
  ```

  こうして、スナップ表に対して照会を作成することができます。これには、照会 (フェデレーテッドあるいは非フェデレーテッド) ごとに 1 つの行、およびサーバーのステートメント・キャッシュ内の照会フラグメントごとに 1 つの行が入ります。

  リモート照会フラグメントの名前は、表関数の stmt_text フィールドでリモート照会テキスト (大括弧で囲まれている) の前に付加された、フラグメントの送信先のサーバーになります。例えば、以下の照会を使用して長時間実行のリモート・フラグメントを検索することができます。

```
SELECT total_exec_time, rows_read, total_usr_cpu_time, num_executions,
 substr(stmt_text,1,30)
FROM TABLE(SNAPSHOT_DYN_SQL ('dbname', -1))AS snaptab
-- remote fragments only
WHERE stmt_text LIKE '[%]%'
ORDER BY total_exec_time;
```

フェデレーテッド・ステートメント全体の実行時間と、そのステートメントの代わりに他のデータ・ソースに送信されたリモート・フラグメントの実行時間とを比較することにより、処理時間の多くがどこで費やされているかを知ることができます。
フェデレーテッド照会の代わりに、どの照会フラグメントをリモート・ソースに送信するかを決定するには、照会に関する EXPLAIN 実行プランを参照してください。

フェデレーテッド照会のスナップショット・モニターの例

このトピックでは、リモート Oracle データ・ソースを含むフェデレーテッド照会の、テキスト・ベースの動的 SQL スナップショットの出力の例を示します。

以下のステートメントは、ステートメント・キャッシュに現在入っているすべてのステートメントのスナップショットを検索します。これには、他のデータ・ソースに送信されたフェデレーテッド・ステートメントおよびリモート・フラグメントが含まれます。

```
GET SNAPSHOT FOR DYNAMIC SQL ON <database_name>
```

データベース名とは、ローカル・フェデレーテッド・データベースの名前のことです。

以下の例における出力は、次のステートメントの結果生成されたものです。

```
GET SNAPSHOT FOR DYNAMIC SQL ON FEDDB
```

この例は、フェデレーテッド・ステートメントおよびそのフェデレーテッド・ステートメントによってプッシュダウンされた 1 つのリモート・フラグメントを示しています。リモート・ステートメント・テキストの前に付加されたリモート・サーバー名 (大括弧で囲まれている) を見つけることによって、リモート・フラグメントを識別することができます。この例では、リモート Oracle サーバーの名前は ORA9 です。最初の項目は、全体の経過時間を含む、ニックネームを参照するフェデレーテッド SQL ステートメントを示しています。 2 番目の項目は、リモート Oracle 表名を参照するソース [ORA9] に送信されるリモート・ステートメントを示しています。

```
  Dynamic SQL Snapshot Result

Database name                       = FEDDB

Number of executions                = 1
Number of compilations              = 1
Worst preparation time (ms)         = 475
Best preparation time (ms)          = 475
Internal rows deleted               = 0
Internal rows inserted              = 0
Rows read                           = 5
Internal rows updated               = 0
Rows written                        = 0
Statement sorts                     = 0
Statement sort overflows            = 0
```

```
Total sort time                    = 0
Buffer pool data logical reads     = Not Collected
Buffer pool data physical reads    = Not Collected
Buffer pool temporary data logical reads   = Not Collected
Buffer pool temporary data physical reads  = Not Collected
Buffer pool index logical reads    = Not Collected
Buffer pool index physical reads   = Not Collected
Buffer pool temporary index logical reads  = Not Collected
Buffer pool temporary index physical reads = Not Collected
Buffer pool xda logical reads      = Not Collected
Buffer pool xda physical reads     = Not Collected
Buffer pool temporary xda logical reads    = Not Collected
Buffer pool temporary xda physical reads   = Not Collected
Total execution time (sec.ms)      = 1.816884
Total user cpu time (sec.ms)       = 0.000000
Total system cpu time (sec.ms)     = 0.020000
Statement text                     = select count(*) from orat.supplier,
         orat.nation where s_nationkey =
         n_nationkey and n_name <> 'FRANCE'


Number of executions               = 1
Number of compilations             = 1
Worst preparation time (ms)        = 0
Best preparation time (ms)         = 0
Internal rows deleted              = 0
Internal rows inserted             = 0
Rows read                          = 1
Internal rows updated              = 0
Rows written                       = 0
Statement sorts                    = 0
Statement sort overflows           = 0
Total sort time                    = 0
Buffer pool data logical reads     = Not Collected
Buffer pool data physical reads    = Not Collected
Buffer pool temporary data logical reads   = Not Collected
Buffer pool temporary data physical reads  = Not Collected
Buffer pool index logical reads    = Not Collected
Buffer pool index physical reads   = Not Collected
Buffer pool temporary index logical reads  = Not Collected
Buffer pool temporary index physical reads = Not Collected
Buffer pool xda logical reads      = Not Collected
Buffer pool xda physical reads     = Not Collected
Buffer pool temporary xda logical reads    = Not Collected
Buffer pool temporary xda physical reads   = Not Collected
Total execution time (sec.ms)      = 1.337672
Total user cpu time (sec.ms)       = 0.000000
Total system cpu time (sec.ms)     = 0.000000
Statement text                     = [ORA9] SELECT COUNT(*) FROM "TPCH"."NATION"
         A0, "TPCH"."SUPPLIER" A1 WHERE
         (A0."N_NAME" <> 'FRANCE                    ') AND
         (A1."S_NATIONKEY" = A0."N_NATIONKEY")
```

フェデレーテッド・ステートメントの合計経過時間は、最初の項目のうち「total execution time」(合計実行時間) によって示されています。その項目には、フェデレーテッド・サーバーで照会によって消費されたユーザーおよびシステム CPU 時間も示されています。リモート・フラグメントの場合、照会はフェデレーテッド・サーバーの外部で実行されるため、ユーザーおよびシステム CPU 時間は利用できません。リモート・フラグメントに関しては、ソート・データおよびバッファー・プール・データも収集されません。フェデレーテッド・サーバーにおいては、ソート・データとバッファー・プール・データの両方が収集されます。しかし、ニックネーム・アクセスではバッファー・プールが使用されないため、フェデレーテッ

ド・サーバーの項目について示されるバッファー・プール・アクティビティーは、カタログまたはローカルの表アクセスによるものであることに注意してください。

フェデレーテッド・データベース・システムのモニター・エレメント

このトピックでは、フェデレーテッド・システムに関する情報を提供するモニター・エレメントについて説明します。

フェデレーテッド・システムは、IBM 製および他社製のリレーショナルおよび非リレーショナルの異なるプラットフォーム上に常駐する様々なデータ・ソースにアクセスします。分散データへのアクセスを統合して、異機種混合環境で単一のデータベース・イメージをユーザーに提供します。

次のエレメントにより、フェデレーテッド・システム内で実行される各アプリケーションからデータ・ソースへのすべてのアクセスに関する情報、およびフェデレーテッド・サーバー・インスタンス内で実行される特定のアプリケーションからデータ・ソースへのアクセスに関する情報が示されます。次のエレメントがあります。

- datasource_name - データ・ソース名 : モニター・エレメント
- disconnects - 切断回数 : モニター・エレメント
- insert_sql_stmts - 挿入回数 : モニター・エレメント
- update_sql_stmts - 更新回数 : モニター・エレメント
- delete_sql_stmts - 削除回数 : モニター・エレメント
- dynamic_sql_stmts - 試行された動的 SQL ステートメント : モニター・エレメント
- create_nickname - ニックネーム作成回数 : モニター・エレメント
- passthrus - パススルー数 : モニター・エレメント
- stored_procs - ストアード・プロシージャー数 : モニター・エレメント
- remote_locks - リモート・ロック数 : モニター・エレメント
- sp_rows_selected - ストアード・プロシージャーによって戻された行数 : モニター・エレメント
- select_time - 照会応答時間 : モニター・エレメント
- insert_time - 挿入応答時間 : モニター・エレメント
- update_time - 更新応答時間 : モニター・エレメント
- delete_time - 削除応答時間 : モニター・エレメント
- create_nickname_time - ニックネーム作成応答時間 : モニター・エレメント
- passthru_time - パススルー時間 : モニター・エレメント
- stored_proc_time - ストアード・プロシージャー時間 : モニター・エレメント
- remote_lock_time - リモート・ロック時間 : モニター・エレメント

以下の例は、dynamic_sql_statement スナップショットを示しています。

```
Statement text = [ORACLE817]SELECT A0.C1,A0.C2 FROM ORA_T A0 WHERE A0.C3 = :H0
```

すべてのリモート・ステートメントでは、ステートメント・テキストはリモート・データ・ソース名 (大括弧で囲まれる) で始まり、リモート・データ・ソースに送信される実際のテキストが後に続きます。

第 18 章 クライアント・アプリケーションとデータ・ソースの対話

クライアント・アプリケーションにとって、フェデレーテッド・システム内のデータ・ソースは、 1 つの集合データベースとして見えます。データ・ソースからデータを入手するために、アプリケーションは DB2 SQL の照会をフェデレーテッド・データベースにサブミットします。するとフェデレーテッド・データベースは、その照会を該当のデータ・ソースに配布し、このデータをアプリケーションに戻すか、または要求されたアクションを実行します。

フェデレーテッド・データベースは、ローカル表からのデータとリモート・データ・ソースからのデータを同じ SQL ステートメントの中で結合することができます。たとえば、ローカルの DB2 表にあるデータ、Informix 表、および Sybase ビューを、1 つの SQL ステートメントで結合することができます。データ・ソースがフェデレーテッド・データベース内の通常のリレーショナル表またはビューであるかのように SQL ステートメントを処理することにより、フェデレーテッド・システムは、リレーショナル・データを非リレーショナル・データと結合することができます。

フェデレーテッド・システムでは、ニックネームを使用してデータ・ソースにアクセスできます。ニックネーム は、アプリケーションがデータ・ソース・オブジェクト (表やビューなど) を参照するために使用するフェデレーテッド・データベース・オブジェクトです。データ・ソースに書き込む (例えば、データ・ソースの表を更新する) ために、アプリケーションは (ニックネームを使用して) DB2 SQL を使用することができます。別の方法として、パススルーと呼ばれる特別なセッションを使用すれば、アプリケーションはデータ・ソース側の SQL ダイアレクトを使用して、 (ニックネームを使用せずに)、データ・ソースを直接アクセスすることができます。

DB2 SQL およびニックネームを使用するアプリケーションは、フェデレーテッド・データベースが認識するすべてのデータ・タイプにアクセスすることができます。

フェデレーテッド・データベースのカタログには、フェデレーテッド・データベース内のオブジェクトの情報と、データ・ソース側のオブジェクトの情報が入っています。カタログにはフェデレーテッド・データベース全体についての情報が含まれるので、*グローバル・カタログ* と呼ばれます。

第 19 章 SQL ステートメントでデータ・ソース・オブジェクトをニックネームで参照する

フェデレーテッド・システムでは、SQL ステートメントの中でデータ・ソース・オブジェクトを記述する場合、そのオブジェクトに対してニックネームを定義し、それを使用します。フェデレーテッド・システムでは、SQL ステートメントの中でデータ・ソース、スキーマ、およびオブジェクトの完全修飾名を使用しても認識されません。

照会でニックネームを指定するには、データ・ソース・オブジェクトのニックネームをフェデレーテッド・データベースに登録しておく必要があります。一般に、SQL ステートメント内の、ローカル表を指定できる場所であれば、ニックネームを指定できます。

例: SELECT、INSERT、UPDATE、および DELETE ステートメントでニックネームを使用する

Informix 表内の PERSON.DEPT と呼ばれる表を表す、ニックネーム NFXDEPT を定義するとします。ここで、

- PERSON はデータ・ソース・スキーマです
- DEPT はデータ・ソース表名です

こうすると、SELECT * FROM NFXDEPT の使用がフェデレーテッド・サーバーから許可されます。ただし、ステートメント SELECT * FROM PERSON.DEPT は許可されません (パススルー・セッションを除く)。フェデレーテッド・サーバーには PERSON.DEPT がニックネームとして登録されていません。

例: CREATE TABLE ステートメントでニックネームを使用する

ニックネームを定義してあるリモート表に基づいて、ローカル表を作成するとします。 CREATE TABLE ステートメントの例を次に示します。

```
CREATE TABLE table_name LIKE nickname
```

DDL ステートメントでのニックネーム

DDL ステートメントでニックネームを指定するには、データ・ソース・オブジェクトのニックネームをフェデレーテッド・データベースに登録しておく必要があります。このトピックでは、フェデレーテッド・システムで使用する DDL ステートメントの例をいくつか示します。

COMMENT ON ステートメントでニックネームを使用する

COMMENT ON ステートメントは、フェデレーテッド・データベースのグローバル・カタログにコメントを追加または置き換えます。 COMMENT ON ステートメントは、ニックネームおよび、ニックネームに定義された列に有効です。このステートメントは、データ・ソースのカタログは更新しません。

GRANT および REVOKE ステートメントでニックネームを使用する

GRANT および REVOKE ステートメントは、ある種の特権に関して、また、すべてのユーザーおよびグループに関してニックネームに有効です。ただし、フェデレーテッド・システムは、そのニックネームが指すデータ・ソース上のオブジェクトに対して、相当する GRANT または REVOKE ステートメントを出すのではありません。

たとえば、ユーザー JON が、索引を持たない Oracle 表にニックネームを作成したとします。ニックネームは ORAREM1 です。後になって、Oracle DBA がこの表に索引を定義しました。ユーザー EILEEN は、表により効率的にアクセスする方法を照会オプティマイザーが工夫できるようにするため、この索引の存在をフェデレーテッド・データベースに知らせたいと考えます。この場合 EILEEN は、ORAREM1 用の「索引の仕様」を作成することにより、新しい索引の存在をフェデレーテッド・データベースに知らせることができます。

索引についての情報は SYSSTAT.INDEXES カタログ・ビューに保管されています。 GRANT ステートメントを使用して EILEEN にこのニックネームに対する索引特権を与えると、 EILEEN は「索引の仕様」を作成できるようになります。

```
GRANT INDEX ON NICKNAME ORAREM1 TO USER EILEEN
```

ニックネーム ORAREM1 に関して索引の仕様を作成するという EILEEN の特権を取り消すには、次のような REVOKE ステートメントを使用します。

```
REVOKE INDEX ON ORAREM1 FROM USER EILEEN
```

アプリケーションに影響を与えるデータ・ソース統計情報

データ・ソース・オブジェクトにニックネームを作成すると、フェデレーテッド・データベースのグローバル・カタログは、そのオブジェクトについての情報を使用して更新されます。照会オプティマイザーはこの情報を使用して、オブジェクトからデータをどのように検索するかの計画を立てます。

データ・ソースの情報が最新のものであることを確認することが重要です。フェデレーテッド・データベースは、データ・ソース・オブジェクトの変更を自動的に検出することはありません。

グローバル・カタログに保管されるデータベース・オブジェクト統計情報

データ・ソース・オブジェクトについてグローバル・カタログに保管される情報は、オブジェクトのタイプにより異なります。データベース表およびビューの場合、オブジェクトの名前、列名および属性がグローバル・カタログに保管されます。

表またはニックネームの場合は、次の情報も含まれます。

- 統計情報。たとえば、行数および、行が存在するページの数。フェデレーテッド・データベースが最新の統計情報を入手できるようにするため、ニックネームを作成する前に、データ・ソース側で RUNSTATS コマンドに相当するものを表に対して実行してください。
- 索引記述。表が索引を持たない場合、通常、索引定義に入っているメタデータをカタログに提供することができます。例えば、あるリモート表のニックネームを

作成した後、データ・ソース側でその表に対して索引が作成されたとします。フェデレーテッド・サーバー側では、そのリモート索引を表す索引の仕様を作成できます。索引の仕様は、CREATE INDEX ステートメントの発行と、表のニックネームの参照によって作成できます。 CREATE INDEX ステートメントで SPECIFICATION ONLY 文節を指定すると、索引の仕様のみ作成されます。索引の仕様により、リモート索引が存在することがフェデレーテッド・オプティマイザーに通知されます。しかし、生成されるのはメタデータだけです。フェデレーテッド・サーバー上に実際の索引が作成されるわけではありません。さらに、グローバル・カタログに統計情報が提供されることもありません。索引の仕様のシグニチャーを、リモート索引と正確に同じにした場合には (つまり同じ名前、および同じ順にある同じ列)、SYSPROC.NNSTAT を使用することによって、そのニックネームおよび索引の仕様に関する統計情報を更新できます。

どのようなデータ・ソース情報がグローバル・カタログに保管されているかを知るには、SYSCAT.TABLES および SYSCAT.COLUMNS カタログ・ビューを照会してください。どのようなデータ・ソース索引情報がカタログに保管されているか、または特定の索引の仕様に何が含まれているかを知るには、SYSCAT.INDEXES カタログ・ビューを照会してください。

SYSCAT ビューではなく SYSSTAT ビューを使用した統計情報の更新

SYSCAT ビューは、SYSCAT スキーマに含まれる読み取り専用カタログ・ビューです。 SYSSTAT ビューは、オプティマイザーで使用される統計情報が含まれている更新可能カタログ・ビューです。 SYSSTAT ビューは、SYSSTAT スキーマに含まれています。

SYSCAT スキーマ内のビューに対して UPDATE または INSERT 操作を発行すると、失敗します。ニックネームに関する統計情報を手動で変更するには、SYSSTAT スキーマの更新可能カタログ・ビューを使用してください。

ニックネームの列オプションの定義

列オプションは CREATE NICKNAME および ALTER NICKNAME ステートメントのパラメーターです。最初にニックネームを作成する時に、または既存のニックネームを変更することにより、列オプションを指定することができます。

列オプションを通して提供する情報は、グローバル・カタログに保管されます。

非リレーショナル・データ・ソース

列オプションは、それぞれの非リレーショナル・ラッパーにユニークです。通常、このオプションは、CREATE NICKNAME ステートメントを発行する場合に設定します。

リレーショナル・データ・ソース

リレーショナル・データ・ソースに使用できる列オプションには、NUMERIC_STRING と VARCHAR_NO_TRAILING_BLANKS の 2 つがあります。

NUMERIC_STRING 列オプションの設定

データ・ソースのストリング列に数字だけが入っており、(ブランクも含めて) 他の文字が入っていない場合は、NUMERIC_STRING 列オプションを Y に設定します。

NUMERIC_STRING 列オプションを Y に設定すると、この列を使用する照会のソート操作および比較演算が最適化されます。以下に例を示します。

```
ALTER NICKNAME nickname
   ALTER COLUMN local_column_name
   OPTIONS (SET NUMERIC_STRING 'Y')
```

VARCHAR_NO_TRAILING_BLANKS 列オプションの設定

データ・ソースのストリング列に末尾ブランクが含まれない場合には、VARCHAR_NO_TRAILING_BLANKS 列オプションを Y に設定します。

ある種のデータ・ソース (Oracle など) は、フェデレーテッド・データベースで使用されているものと同じブランク埋め込みストリング比較論理を使用しません。これは、VARCHAR および VARCHAR2 などのデータ・タイプが該当します。その結果、これらのデータ・タイプを含む述部は、照会結果を一貫性のあるものとするために、照会オプティマイザーによって書き直される必要があります。

照会ステートメントを書き直すと、パフォーマンスに影響を与える場合があります。このオプションを特定の列に設定すると、これらの列についての情報が照会オプティマイザーに提供され、より効率の良い SQL ステートメントの生成が可能になります。

例:

```
ALTER NICKNAME nickname
   ALTER COLUMN local_column_name
   OPTIONS (SET VARCHAR_NO_TRAILING_BLANKS 'Y')
```

第 20 章 フェデレーテッド・ビューの作成および使用

全選択の中でニックネームへの参照が含まれるビューは、フェデレーテッド・ビュー です。基本表は、フェデレーテッド・ビューではデータ・ソースの表名ではなく、ニックネームで参照されます。

このタスクについて

制約事項

複数のデータ・ソース・オブジェクトから作成されたフェデレーテッド・ビューは、読み取り専用ビューであり、更新することはできません。

1 つのみのデータ・ソース・オブジェクトから作成されたフェデレーテッド・ビューは、読み取り専用ビューである場合もそうでない場合もあります。

- 単一の非リレーショナルのデータ・ソースから作成されたフェデレーテッド・ビューは、読み取り専用です。
- 1 つのリレーショナル・データ・ソースから作成されたフェデレーテッド・ビューは、 CREATE VIEW ステートメントに何が含まれているかにより、更新が可能な場合があります。

このタスクについて

フェデレーテッド・ビューを使用する利点は、中央リレーショナル・データベース・マネージャー内のローカル表に関して定義されているビューを使用する利点と似ており、次のとおりです。

- ビューがデータを統合して表示する
- ビューから機密データや重要なデータを含む表列を除外できる

手順

ニックネームを持つデータ・ソース・オブジェクトからのフェデレーテッド・ビューを作成します。データ・ソース・データのフェデレーテッド・データベース・ビューを作成するアクションは、「ニックネームのビューの作成」と呼ばれることがあります。このような言い方をする理由は、作成されるフェデレーテッド・ビューにとって、CREATE VIEW ステートメントの全選択は、フェデレーテッド・ビューに入るはずの、それぞれのデータ・ソース表およびビューの、ニックネームを指す必要があるからです。

フェデレーテッド・ビューの作成 - 例

このトピックでは、フェデレーテッド・ビューの作成例を示します。

例: いくつかのデータ・ソース・オブジェクトから類似データをマージするフェデレーテッド・ビューの作成

ヨーロッパ、アジア、および南アメリカの 3 つの別個のサーバー上に顧客データがあるとします。ヨーロッパの顧客データは Oracle 表内にあります。その表のニックネームは ORA_EU_CUST です。アジアの顧客データは Sybase 表内にあります。その表のニックネームは SYB_AS_CUST です。南アメリカの顧客データは Informix 表内にあります。その表のニックネームは INFMX_SA_CUST です。各表には、顧客番号 (CUST_NO)、顧客名 (CUST_NAME)、製品番号 (PROD_NO)、および注文数量 (QUANTITY) を含む列があります。これら 3 つのニックネームから、この顧客データをマージするビューを作成するための構文は、次のとおりです。

```
CREATE VIEW FV1
   AS SELECT * FROM ORA_EU_CUST
   UNION
   SELECT * FROM SYB_AS_CUST
   UNION
   SELECT * FROM INFMX_SA_CUST
```

例: フェデレーテッド・ビューを作成するためのデータの結合

顧客データと販売データが別々のサーバー上にあるとします。顧客データは Oracle 表内にあります。その表のニックネームは ORA_EU_CUST です。販売データは Sybase 表内にあります。その表のニックネームは SYB_SALES です。顧客情報と顧客の購入品とを対応させたいとします。各表には、顧客番号 (CUST_NO) を含む列があります。これら 2 つのニックネームからこのデータを結合するフェデレーテッド・ビューを作成するための構文は、次のとおりです。

```
CREATE VIEW FV4
   AS SELECT A.CUST_NO, A.CUST_NAME, B.PROD_NO, B.QUANTITY
   FROM ORA_EU_CUST A, SYB_SALES B
   WHERE A.CUST_NO=B.CUST_NO
```

第 21 章 分離レベルによるデータ保全性の維持

分離レベルとは、あるアプリケーション・プロセスが、並行稼動している他のアプリケーション・プロセスから分離している度合いを定義するものです。

データ・ソース表のデータ保全性は、表の行を特定の分離レベルでロックすることを要求することにより、維持できます。

ロッキングは、データ・ソース側の基本表の行に対して行われます。ただし、データベース・マネージャーは、複数の行ロックを 1 つの表ロックで置き換えることができます。このアクションは、*ロック・エスカレーション* と呼ばれます。アプリケーション・プロセスには、少なくとも要求した最低限のロック・レベルが保証されます。

フェデレーテッド・データベースの分離レベルは、以下のとおりです。

RR 反復可能読み取り (Repeatable read)

RS 読み取り固定 (Read stability)

CS カーソル固定 (Cursor stability) (デフォルト)

UR 非コミット読み取り (Uncommitted read)

分離のタイプは、ステートメント・レベルの分離と接続レベルの分離です。

分離は、以下のアクションを実行する際に設定できます。

- アプリケーションをプリコンパイルまたはバインドする際。分離レベルは、アプリケーションを準備またはバインドする時点で指定できます。 BIND および PREP コマンドに指定された分離レベルは、フェデレーテッド・サーバーがリモート・データ・ソースに接続する際のデフォルトの分離レベルになります。
- SQL ステートメントの中で WITH 文節を使用する際。このアクションは、ステートメント・レベルの分離と呼ばれます。 WITH 文節は、SELECT、UPDATE、INSERT、および DELETE ステートメントの中で使用できます。

フェデレーテッド・サーバーがあるステートメントの分離レベルを検出できなかった場合は、フェデレーテッド・サーバーがデータ・ソースに接続する際に確立された分離レベルを使用します。

以下の表に、接続レベルの分離を使用するデータ・ソース、使用される分離レベル、およびフェデレーテッド・サーバー上でそれに相当する分離レベルのリストを示します。

表 20. データ・ソースと分離レベル

データ・ソース	制限の最も大きい分離レベル	制限のやや大きい分離レベル	制限のやや小さい分離レベル	制限の最も小さい分離レベル
フェデレーテッド・データベース	反復可能読み取り (Repeatable read)	読み取り固定 (Read stability)	カーソル固定 (Cursor stability)	非コミット読み取り (Uncommitted read)
DB2 製品ファミリー	反復可能読み取り (Repeatable read)	読み取り固定 (Read stability)*	カーソル固定 (Cursor stability)	非コミット読み取り (Uncommitted read)
Informix	反復可能読み取り (Repeatable read)	反復可能読み取り (Repeatable read)	カーソル固定 (Cursor stability)	ダーティー読み取り (Dirty read)
JDBC	シリアライズ可能 (Serializable)	反復可能読み取り (Repeatable read)	読み取りコミット (Read committed)	読み取り非コミット (Read Uncommitted)
Microsoft SQL Server	シリアライズ可能 (Serializable)	反復可能読み取り (Repeatable read)	読み取りコミット (Read committed)	読み取り非コミット (Read Uncommitted)
ODBC	シリアライズ可能 (Serializable)	反復可能読み取り (Repeatable read)	読み取りコミット (Read committed)	読み取り非コミット (Read Uncommitted)
Oracle	シリアライズ可能 (Serializable)	シリアライズ可能 (Serializable)	読み取りコミット (Read committed)	読み取りコミット (Read committed)
Sybase	レベル 3	レベル 3	レベル 1	レベル 0

*DB2 for VM and VSE Server データ・ソースの場合、分離レベルは反復可能読み取りです。

フェデレーテッド・サーバーがデータ・ソースに接続する際、CURRENT ISOLATION 特殊レジスターは使用されません。

非リレーショナル・データ・ソースには、分離レベルに相当する概念がありません。 OLE DB および Teradata には分離レベルの概念がありますが、フェデレーテッド・サーバーではサポートされていません。フェデレーテッド・データベースの分離レベルと、OLE DB、Teradata、および非リレーショナル・データ・ソースとの間に、分離レベルのマッピングはありません。

フェデレーテッド・システムでのステートメント・レベルの分離

フェデレーテッド・データ・ソースでは、WITH 分離文節を使用することによってステートメントの分離を指定する必要があります。

ステートメント・レベルの分離を使用する場合は、ステートメントの中で WITH 分離文節を使用する必要があります。ステートメント・レベルの分離のためのコール・レベル・インターフェース (CLI) またはその他のアプリケーション API で属性を使用しても、それはステートメント分離には影響しません。

フェデレーテッド・システムの中でステートメント・レベルの分離をサポートするデータ・ソースは、DB2 製品ファミリーと Microsoft SQL Server です。ステートメント分離は、DB2 製品ファミリーと SQL Server のリモート・データ・ソースに送られます。

ステートメント・レベルの分離をオンまたはオフにするには、DB2_STATEMENT_ISOLATION サーバー・オプションを使用します。このオプションは CREATE SERVER および ALTER SERVER ステートメントに指定します。このサーバー・オプションは自動的に Y に設定されます。

WITH 分離文節は、以下のステートメントの中で使用できます。

SELECT

SELECT INTO

検索 DELETE

INSERT

検索 UPDATE

DECLARE CURSOR

ロック要求文節

ロック要求文節は、SELECT または SELECT INTO ステートメントの中で使用できます。ロック要求文節をサポートするフェデレーテッド・データ・ソースは、DB2 for Linux, UNIX, and Windows、および DB2 for z/OS です。

WITH 文節を使用した分離レベル設定に関する制限

ステートメントの分離レベルを指定する際には、以下の条件が適用されます。

- 副照会の中で WITH 文節を使用することはできません。
- UR 分離レベルが適用されるのは、全選択または SELECT INTO ステートメントの結果表が読み取り専用の場合だけです。それ以外の場合、データ・ソースが DB2 ファミリーなら、ステートメントの UR 分離レベルは、UR から CS に変更されます。 SQL サーバー・データ・ソースの場合、UR 分離レベルは読み取りコミットにアップグレードされます。
- 以下のデータ・ソースにおいてロック要求文節を指定した場合、フェデレーテッド・サーバーはその文節を無視します。
  - DB2 for System i
  - DB2 for VM
  - Microsoft SQL server

フェデレーテッド・システムでの接続レベルの分離

フェデレーテッド・サーバーは、指定された分離レベルを、データ・ソース側で対応する分離レベルにマップします。

分離レベルは、データ・ソースとの各接続ごとにラッパーによって決定されます。

フェデレーテッド・サーバーがデータ・ソースに接続する際、リモート・データ・ソース側の分離レベルは、フェデレーテッド・サーバーのレベルに相当するレベル

に設定されます。同等レベルがない場合は、フェデレーテッド・サーバーは分離レベルを、その次の制限レベルに設定します。データ・ソースへの接続が確立されたら、接続中は分離レベルを変更できません。

Teradata 以外のすべてのラッパーでは、接続分離レベルを追跡し続けます。接続のセットアップ時にラッパーは、接続分離レベルを現行分離レベルに相当するレベルに設定します。現行分離レベルは、現在のセクション (データ・ソースに対する最初のフェデレーテッド・ステートメント) の分離レベルです。 Teradata ラッパーでは接続分離レベルを変更する手段がないため、Teradata ラッパーのレベルは、常に READ 分離レベル (デフォルト) です。

第 22 章 フェデレーテッド LOB サポート

フェデレーテッド・データベース・システムを使用すると、リモート・データ・ソース側のラージ・オブジェクト (LOB) にアクセスし、取り扱うことができます。

フェデレーテッド・システムは、DRDA、Informix、Microsoft SQL Server、Oracle、および Sybase データ・ソースで LOB の SELECT 操作をサポートします。例:

```
SELECT empname, picture FROM infmx_emp_table
   WHERE empno = '01192345'
```

ここで、*picture* は LOB 列を表し、*infmx_emp_table* は、従業員データが入っている Informix 表を指すニックネームです。

フェデレーテッド・システムは、DRDA ラッパーを使用することにより、以下のデータ・ソースでの LOB の SELECT、INSERT、UPDATE、および DELETE 操作をサポートします。

- DB2 for z/OS (バージョン 7 以降)
- DB2 for System i (バージョン 5)
- DB2 Database for Linux, UNIX, and Windows (バージョン 7 以降)

DB2 Database for Linux, UNIX, and Windows がサポートする読み取りおよび書き込み操作は、次の表にリストされています。

表 21. LOB に対する読み取りおよび書き込みサポート

データ・ソース	操作のタイプ
DB2 for z/OS、DB2 for System i、DB2 Database for Linux, UNIX, and Windows[1]	読み取りおよび書き込み
BioRS	読み取り専用
Informix	読み取り専用
JDBC	読み取り専用
Microsoft SQL Server	読み取り専用
Oracle (NET8 ラッパー) [2]	読み取りおよび書き込み
ODBC	読み取り専用
Sybase	読み取り専用
Teradata	読み取り専用
Web サービス	読み取り専用、および CLOB の場合のみバインドアウト
XML	読み取り専用

注:

1. LOB サポートには、DB2 for System i (バージョン 5 以降) が必要です。DB2 Information Integrator バージョン 8 は、DB2 Database for Linux, UNIX, and Windows バージョン 7 の LOB データにはアクセスできません。
2. Oracle の LONG 列に対して挿入、更新、および削除の各操作を実行するには、リモート列を LONG から LOB に移行し、ニックネームを再作成することが必要です。

Teradata LOB

Teradata LOB は DB2 LOB と多少異なります。 Teradata には、DB2 for Linux, UNIX, and Windows ソフトウェアでサポートされる LOB と同じような大きさのデータ・タイプはありません。ただし、場合によっては最大長が 64000 バイトの Teradata データ・タイプもあります。これらのデータ・タイプは、CHAR、 VARCHAR、 BYTE、 VARBYTE、 GRAPHIC、および VARGRAPHIC です。 Teradata データ・タイプの長さが対応する DB2 データ・タイプの限界を超える場合、これらの Teradata データ・タイプは、対応する DB2 LOB データ・タイプにマップされます。

LOB の長さ

一部のデータ・ソース (Oracle および Informix など) では、LOB 列の長さはシステム・カタログに保管されません。表に対してニックネームを作成する場合は、データ・ソース・システム・カタログの情報が列の長さも含めて検索されます。 LOB 列については長さが存在しないため、フェデレーテッド・データベースでは、長さは DB2 Database for Linux, UNIX, and Windowsの LOB 列の最大長であると想定されます。フェデレーテッド・データベースは、最大長を、ニックネーム列の長さとしてフェデレーテッド・データベース・カタログに保管します。

LOB ロケーター

アプリケーションは、リモート・データ・ソースに保管された LOB 用の LOB ロケーターを要求することができます。 LOB ロケーター とは、ホスト変数に保管された 4 バイトの値です。アプリケーションは LOB ロケーターを使用して、データベース・システムに保持されている LOB 値 (または LOB 式) を参照することができます。

LOB ロケーターを使用すると、アプリケーションは LOB 値が通常のホスト変数に保管されているかのように LOB 値を扱うことができます。 LOB ロケーターを使用する場合、 LOB 値をデータ・ソース・サーバーからアプリケーションに転送する必要はありません (したがって、また戻す必要もありません)。

フェデレーテッド・データベースは LOB をリモート・データ・ソースから検索し、それをフェデレーテッド・サーバーに保管し、次に、保管された LOB に対して LOB ロケーターを発行します。 LOB ロケーターは次の場合にリリースされます。

- アプリケーションが FREE LOCATOR SQL ステートメントを発行
- アプリケーションが COMMIT ステートメントを発行
- フェデレーテッド・インスタンスが再始動された

LOB の制約事項

フェデレーテッド・システムでは、LOB に関していくつかの制約事項があります。

LOB には以下の制約事項が適用されます。

- フェデレーテッド・データベースは、リモート LOB をファイル参照変数にバインドすることはできません
- LOB は、パススルー・セッションではサポートされません

- LOB は、ストアード・プロシージャーのパラメーターとしてはサポートされません

LOB 処理のパフォーマンスに関する考慮事項

LOB データをフェッチおよび処理するフェデレーテッド・アプリケーションを開発する場合、アプリケーション設計者およびデータベース管理者は、LOB 処理がパフォーマンスにどう影響するかについてよく理解しておく必要があります。

アプリケーションがフェデレーテッド・データ・ソースからデータをフェッチする際、フェデレーテッド・サーバーは、データをアプリケーションに送る前に、フェッチしたデータをそれ自身のアプリケーション・バッファーに入れる必要があります。 LOB はバッファー・プール内では処理されないため、LOB データは、まずフェデレーテッド・サーバーについて定義されている TEMPORARY 表スペースをパススルーすることが必要です。パフォーマンスを改善し、リソース消費量をなるべく少なくするため、アプリケーション設計者は、LOB データのマテリアライズ処理を必要な場合にのみ実行するようにしてください。

同じように、フェデレーテッド・サーバーがリモート LOB データを更新する場合、データはデータ・ソースに渡される前に、そのフェデレーテッド・サーバーに割り当てられている TEMPORARY 表スペースをパススルーする必要があります。

一時 LOB は、フェデレーテッド・サーバーに割り当てられている TEMPORARY 表スペースを使用します。したがって、データベース管理者は、この TEMPORARY 表スペースのサイズを大きくして、LOB を処理するために十分な作業域を確保することが必要になる場合があります。

推奨事項: LOB の処理におけるパフォーマンスを最大にするため、TEMPORARY 表スペースをシステム管理 (SMS) スペースとして定義し、それを入出力処理能力の高いディスク上に配置するようにしてください。

DB2 コール・レベル・インターフェースを使用したフェデレーテッド LOB のアクセス

フェデレーテッド・サーバーでは、LOB データの選択のために 2 つの DB2 CLI API がサポートされています。

- SQLFetch API は、単一の操作でフェデレーテッド・サーバーまたはデータ・ソースからの LOB をフェッチして、アプリケーション・バッファーに入れます。
- SQLGetData API は、一度に一定の大きさの塊で LOB をフェッチします。 LOB 全体をフェッチしてアプリケーション・バッファーに入れるには、この API を繰り返して呼び出すことが必要になる場合があります。

推奨事項: パフォーマンスを最適化するため、フェデレーテッド・サーバーによって LOB をフェッチする際には、SQLGetData API を使用してください。

フェデレーテッド・サーバーでは、LOB データを更新するために SQLExecute および SQLPutData の各 API がサポートされています。 SQLExecute API は単一の操作で LOB データを更新します。一方、SQLPutData API の場合は、アプリケーション・バッファーの LOB データのすべてをサーバーに送るために何度も繰り返して

呼び出すことが必要になる場合があります。フェデレーテッド環境においてこれらの API は、それぞれ同じレベルで実行されます。

トラステッドおよび fenced ラッパー

トラステッドまたは fenced として定義されたラッパー用に作成されたニックネームはいずれも、LOB のフェッチまたは更新時には同等に実行されます。

第 23 章 データ・ソース照会の分散要求

フェデレーテッド・データベースにサブミットされる照会は、単一のデータ・ソースから結果を要求することもできますが、通常は、複数のデータ・ソースを含む複数の要求です。典型的な照会は複数のデータ・ソースに分散して配布されるので、*分散要求* と呼ばれます。

一般的に、分散要求では、副照会、セット演算子、および結合副選択の 3 つの SQL 用法の中から 1 つまたは複数を使用して、どこからデータを検索するかを指定します。

データ・ソース照会の分散要求 - 例

このトピックに含まれる例では、副照会、セット演算子、および結合操作を使用した分散要求を示します。

以下の例では、フェデレーテッド・サーバーが DB2 for z/OS データ・ソース、DB2 for System i データ・ソース、および Oracle データ・ソースにアクセスするように構成されているとします。それぞれのデータ・ソースには、従業員情報を含む表が保管されています。フェデレーテッド・サーバーは、表の場所を指すニックネームにより、それらの表を参照します。

zOS_EMPLOYEES
DB2 for z/OS データ・ソース上の、従業員情報が入っている表のニックネームです。

SYSTEMi_EMPLOYEES
DB2 for System i データ・ソース上の、従業員情報が入っている表のニックネームです。

ORA_EMPLOYEES
Oracle データ・ソース上の、従業員情報が入っている表のニックネームです。

ORA_REGIONS
Oracle データ・ソース上の、従業員が住む地域の情報が入っている表のニックネームです。

次の例は、それぞれの表に定義されたニックネームを使用した、分散要求で使用される 3 つの SQL 用法を示しています。

例: 副照会を使用した分散要求

SYSTEMi_EMPLOYEES には、アジアに住む従業員の電話番号が入っています。また、それらの電話番号に関連する地域コードも入っていますが、コードが表す地域は入っていません。 ORA_REGIONS は、コードと地域の両方のリストです。次の照会は、副照会を使用して中国 (CHINA) の地域コードを見つけています。次に、その地域コードを使用して、SYSTEMi_EMPLOYEES から中国に電話番号を持つ従業員のリストを戻します。

```
SELECT name, telephone FROM db2admin.SYSTEMi_employees
   WHERE region_code IN
   (SELECT region_code FROM dbadmin.ora_regions
   WHERE region_name = 'CHINA')
```

例: セット演算子を使用した分散要求

フェデレーテッド・サーバーは 3 つのセット演算子 (UNION、EXCEPT、および INTERSECT) をサポートします。

- UNION セット演算子を使用すると、複数の任意の SELECT ステートメントの条件を満たす行を組み合わせることができます。
- EXCEPT セット演算子を使用すると、最初の SELECT ステートメントの条件は満たすが、2 番目の SELECT ステートメントの条件は満たさないという行を検索することができます。
- INTERSECT セット演算子を使用すると、両方の SELECT ステートメントの条件を満たす行を検索することができます。

これらの 3 つのセット演算子はすべて、重複する行があっても結果から除かないように指定する All オペランドを使用できます。これにより、余分なソートが必要なくなります。

次の照会は、SYSTEMi_EMPLOYEES 表と zOS_EMPLOYEES 表にあるすべての従業員名と地域コードを、それらの表がそれぞれ異なるデータ・ソースにあっても検索します。

```
SELECT name, region_code
   FROM as400_employees
INTERSECT
SELECT name, region_code
   FROM zOS_employees
```

例: 結合の分散要求

リレーショナルの結合 (join) は、複数の表から検索した列の組み合わせを含む結果セットを作成します。結果セット内の行のサイズを制限するための条件を指定すべきです。

次の照会は、2 つの表にある地域コードを比較し、従業員名とそれに対応する地域名を組み合わせます。それぞれの表は別のデータ・ソースに存在します。

```
SELECT t1.name, t2.region_name
   FROM dbadmin.SYSTEMi_employees t1, dbadmin.ora_regions t2
   WHERE t1.region_code = t2.region_code
```

サーバー・オプションによる分散要求の最適化

フェデレーテッド・システムでは、サーバー・オプション と呼ばれるパラメーターを使用して、全体として、あるデータ・ソースに適用される情報をグローバル・カタログに提供したり、あるいはフェデレーテッド・データベースがデータ・ソースとどのように対話するかを制御します。

このタスクについて

このタスクについて

サーバー・オプションは、特定のデータ・ソースの機能について記述し、そのデータ・ソースに関してフェデレーテッド・サーバー側で認識されている知識を拡張します。たとえば、以下のサーバー・オプションを使用できます。

- VARCHAR_NO_TRAILING_BLANKS サーバー・オプションは、データ・ソース・サーバーにあるすべての VARCHAR 列が末尾にブランクを持たないことをオプティマイザーに知らせます。このオプションは、サーバー上のニックネームによって参照されるあらゆるオブジェクトのすべての VARCHAR2 列に、末尾ブランクが含まれていないことが確かな場合にのみ使用してください。これ以外の場合は、列オプションを使用して、末尾ブランクが含まれていない、サーバー上の個々のオブジェクトの列を指定します。この列オプションも VARCHAR_NO_TRAILING_BLANKS という名前です。
- PLAN_HINTS サーバー・オプションは、プラン・ヒント と呼ばれるステートメント・フラグメントを Sybase データ・ソースに提供します。プラン・ヒントは、表にアクセスするにはどの索引を使用するか、また、結果セットのためのデータの検索にあたって、どの表結合シーケンスを使用するかをデータ・ソース・オプティマイザーが決めるのに役立ちます。

通常、データベース管理者がフェデレーテッド・システム用のサーバー・オプションを設定します。しかし、プログラマーもオプションを有効利用することができ、照会の最適化に役立ちます。たとえば、データ・ソース SYB1 と SYB2 について、PLAN_HINTS サーバー・オプションがデフォルトの N に設定されているとします (N は、このデータ・ソースにプラン・ヒントを提供しないということです)。また、SYB1 と SYB2 からデータを選択する分散要求を書いたとします。これらのデータ・ソースで、オプティマイザーがプラン・ヒントを使用して、このデータにアクセスするためのストラテジーを改善できると期待できます。この場合、アプリケーションがフェデレーテッド・データベースに接続されている間は、Y (Y は、プラン・ヒントを提供する) に設定してデフォルトをオーバーライドすることができます。データ・ソースへの接続が終了すると、設定は自動的に N に戻ります。

手順

サーバー・オプションを設定するには、次のようにします。

手順

サーバー・オプションを設定または変更するには、 SET SERVER OPTION ステートメントを使用します。確実に設定が有効になるように、SET SERVER OPTION ステートメントは CONNECT ステートメントの直後に指定してください。サーバー・オプションは、フェデレーテッド・データベースへ接続されている間、設定されています。

次のタスク

ステートメントを動的に準備してください。SET SERVER OPTION ステートメントは、動的 SQL ステートメントだけに影響を与えます。

静的 SQL の場合、SET SERVER OPTION ステートメントの使用は、その静的 SQL ステートメントの実行だけに影響します。SET SERVER OPTION ステートメントを使用しても、オプティマイザーが生成するプランには影響がありません。

フェデレーテッド照会のキャンセル

アプリケーションを中断し、実行中の照会を取り消すことができます。このサポートを使用すると、リモート・データ・ソース・アプリケーションでの照会をその完了前に取り消し、その照会の中断と取り消しをリモート・データ・ソースに波及させることができます。

このタスクについて

制約事項

バージョン 9.7 では、DB2 Database for Linux, UNIX, and Windows データ・ソースに限り、アプリケーションを中断し、実行中の照会を取り消すことができます。

キャンセルできるのは、フェデレーテッド SELECT ステートメント、INSERT、UPDATE、および DELETE ステートメント、フェデレーテッド・ストアード・プロシージャー、およびパススルー・セッションにおけるステートメントです。

フェデレーテッド・ステートメントには、複数のデータ・ソースにアクセスするセグメントを複数含めることができます。サポートされていないデータ・ソースにセグメントがアクセスする場合 (例えば、Sybase データ・ソースに対する複数の照会セグメント)、ステートメントを取り消すことはできません。

ATQ (非同期表キュー) が使用可能な場合は、フェデレーテッド照会を取り消すことはできません。

このタスクについて

フェデレーテッド・サーバーがフェデレーテッド SQL ステートメントを実行していて、その結果とリモート・データ・ソースからの応答を待機するためにブロックされている場合、フェデレーテッド・サーバー上のアプリケーションは、「フェデレーテッド要求ペンディング」状態になります。 LIST APPLICATIONS コマンドを発行すると、「フェデレーテッド要求ペンディング」状態にあるアプリケーションを識別できます。

アプリケーションが「フェデレーテッド要求ペンディング」状態にある場合、Ctrl-C を使用してアプリケーションを中断および取り消すこともできますし、FORCE APPLICATION コマンドを使用してリモート・サーバー上のアプリケーションを停止することもできます。

手順

フェデレーテッド照会を取り消すには、以下のいずれかの方法を使用します。

- Ctrl-C を押してアプリケーションを中断します。

  Ctrl-C を押すと、アプリケーションを中断し、リモート・サーバーで実行されている照会を取り消すことができます。フェデレーテッド・サーバー上の現行の SQL ステートメントは中断および取り消しになり、ローカル・データベースの整合性は維持されます。リモート・ステートメントも取り消されます。アウトバウンドおよびインバウンド接続はどちらもアクティブな状態を保ちます。

- FORCE APPLICATION コマンドを発行します。

  FORCE APPLICATION コマンドを発行すると、アプリケーションを停止できます。フェデレーテッド・サーバー上のアプリケーションは停止し、トランザクションの整合性はローカルでもリモート・サーバー上でも維持されます。インバウンドおよびアウトバウンド接続は閉じ、リモート・データ・ソースにおけるアプリケーション実行のリモート部分は取り消されます。

第 24 章 パススルーによりデータ・ソースを直接照会する

パススルー・セッションを使用すると、DB2 SQL/API では行えない操作を実行できます。

このタスクについて

このタスクについて

パススルー・セッションは、次の場合に便利です。

- アプリケーションがデータ・ソース側でオブジェクトを作成したり、INSERT、UPDATE または DELETE 操作を実行する必要がある。
- フェデレーテッド・データベースがユニークなデータ・ソース操作をサポートしない。

手順

パススルーによりデータ・ソースを直接照会する。

手順

- SET PASSTHRU ステートメントは、パススルー・セッションを開始し、サーバーを直接アクセスするために使用します。このステートメントは動的に発行することができます。このステートメントの例を次に示します。 SET PASSTHRU ORACLE1 この SET PASSTHRU は、ORACLE1 というサーバー名を使用して、データ・ソースへのパススルー・セッションをオープンします。 ORACLE1 は、サーバー定義を作成したときにデータ・ソース・サーバーに対して登録した名前です。
- パススルー・セッションがオープンされている場合、パススルー・セッション内でオブジェクトを参照するときは、ニックネームではなく、オブジェクトの本当の名前を使用してください。参照されているデータ・ソースがフェデレーテッド・データベースでない場合は、データ・ソースの SQL ダイアレクトを使用する必要があります。
- パススルー・セッションで静的ステートメントをサブミットすると、フェデレーテッド・サーバーに送信されて処理されます。 SQL ステートメントをデータ・ソースにサブミットして処理させたい場合は、パススルー・セッション内でステートメントを動的に準備し、セッションのオープン中に実行させる必要があります。パススルー・セッションでステートメントを動的に準備するには、次のようにします。
  - SELECT ステートメントをサブミットするには、このステートメントと共に PREPARE ステートメントを使用し、次に OPEN、FETCH、および CLOSE ステートメントを使用して、照会の結果にアクセスします。
  - SELECT 以外にサポートされるステートメントについては、次の 2 つのオプションがあります。 PREPARE ステートメントを使用してサポートされるステートメントを準備してから、EXECUTE ステートメントで実行することができます。あるいは、EXECUTE IMMEDIATE ステートメントを使用して、ステートメントを準備し、実行することもできます。

次のタスク

パススルー・セッション中に COMMIT または ROLLBACK コマンドを発行すると、そのコマンドは現行の作業単位を完了させますが、パススルー・セッションは終了しません。

フェデレーテッド・パススルーの考慮事項および制約事項

このトピックでは、パススルー・セッションを使用するタイミングを識別するために必要な考慮事項および制約事項について説明します。

以下の考慮事項および制約事項はすべてのデータ・ソースに適用されます。

- パススルー・セッション内で準備されたステートメントは、同じパススルー・セッション内で実行する必要があります。パススルー・セッション内で準備したステートメントを、そのパススルー・セッション外で実行すると、そのステートメントは失敗し、SQLSTATE 56098 エラーになります。パススルー・セッション外で準備したステートメントを、そのパススルー・セッション内で実行すると、そのステートメントは SET PASSTHRU ステートメントとして処理されます。
- アプリケーションは複数の SET PASSTHRU ステートメントを出すことができますが、最後のセッションだけがアクティブになります。新しい SET PASSTHRU ステートメントを呼び出すと、直前の SET PASSTHRU ステートメントは終了します。同じパススルー・セッションで複数のデータ・ソースにパススルーすることはできません。
- アプリケーションで複数のパススルー・セッションを使用する場合、別のパススルー・セッションを開く前に、必ず COMMIT を発行してください。これにより、現行セッションの作業単位が終了します。
- パラメーター・マーカーは、パススルー・セッションではサポートされません。パラメーター・マーカーの代わりにホスト変数を使用してください。
- パススルー・セッションで定義されたカーソルで WITH HOLD セマンティクスを使用することができます。ただし、COMMIT の後に WITH HOLD セマンティクスを使用しようとして、データ・ソースが WITH HOLD セマンティクスをサポートしない場合、エラーが発生することがあります。
- パススルー・セッション内の SQL ステートメントに定義されたホスト変数は、:H*n* (H は大文字で、*n* はユニークな整数) の形式にする必要があります。 *n* の値はゼロから始まり、順番に番号を振る必要があります。
- パススルーは LOB をサポートしません。
- パススルーはストアード・プロシージャー呼び出しをサポートしません。
- パススルーは SELECT INTO ステートメントをサポートしません。
- パススルーでは SQL または外部ユーザー定義関数がサポートされていません。
- パススルー・セッション中に、動的 SQL COMMIT または ROLLBACK ステートメントを実行することはできません。
- パススルー・セッション中に更新または削除操作を実行する場合、WHERE CURRENT OF CURSOR 条件は使用できません。
- パススルー・モードにおいて、動的 SQL ステートメントはリモート側で実行されますが、静的 SQL ステートメントはフェデレーテッド・サーバーに送信されて処理されます。 SQL ステートメントをデータ・ソースにサブミットして処理

させたい場合は、パススルー・セッション内でステートメントを動的に準備し、セッションのオープン中に実行する必要があります。

- SQL を本体とするストアード・プロシージャーに静的 SET PASSTHRU ステートメントが含まれているなら、それらは、ストアード・プロシージャー作成時にブロックされ、エラー SQL0104N が出されます。パススルー・モードを開始および終了するには、EXECUTE IMMEDIATE ステートメントを使用します。

例:

```
create procedure stp()
dynamic result sets 1
language sql
modifies sql data
begin
declare stmt varchar(100);

DECLARE cur1 CURSOR WITH RETURN TO CALLER
FOR SELECT * FROM t1 ;

set stmt = 'set passthru mvs7';
execute immediate stmt;

set stmt = 'insert into t1 values (20, ''passthru_insert'')';
execute immediate stmt;
commit;

insert into t1 values (20, 'stp_insert');
commit;

OPEN cur1;

end
DB20000I The SQL command completed successfully.

call stp()


Result set 1
--------------------------------
10 local
20 stp_insert

2 record(s) selected.

Return Status = 0


select * from t1

C1 C2
--------------------------------
10 remote
20 passthru_insert

2 record(s) selected.


set passthru reset
DB20000I The SQL command completed successfully.

select * from t1

C1 C2
--------------------------------
```

```
10 local
20 stp_insert

2 record(s) selected.
```

- ストアード・プロシージャーまたはコンパウンド SQL ステートメントから戻っても、パススルー・モードが自動的に終了するわけではありません。上記の例に示されているように、パススルー・モードを終了するには、明示的に SET PASSTHRU RESET ステートメントを呼び出す必要があります。

Oracle データ・ソースへのパススルー・セッション

このトピックでは、パススルー・セッションの中で Oracle データ・ソースに対して SQL ステートメントをサブミットする際に、事前に考慮しておくべき SQL に関するいくつかの考慮事項について説明します。

- Oracle サーバーに対して発行される DDL ステートメントはすべて解析時に実行され、トランザクション・セマンティクスの対象ではありません。操作は、完了時に、Oracle により自動的にコミットされます。ロールバックが起きても、DDL はロールバックされません。
- ロー・データ・タイプからの SELECT ステートメントを発行する場合は、 16 進値を受け取るために RAWTOHEX 関数を使用してください。ロー・データ・タイプへの INSERT を実行する場合は、16 進表記で指定します。

第 25 章 照会処理のチューニング

フェデレーテッド・システムをチューニングして、照会処理のパフォーマンスを改善することができます。

このタスクについて

このタスクについて

SQL 照会をフェデレーテッド・データベースにサブミットすると、SQL コンパイラーが照会を処理し、照会オプティマイザーがそれを分析してアクセス・プランを作成します。照会オプティマイザーはアクセス・プラン情報をフェデレーテッド・データベース内の Explain 表に保管します。Explain 表フォーマット、db2expln および dynexpln ツールを使用すると、特定の SQL ステートメント用のアクセス・プランを知ることができます。

手順

フェデレーテッド・システムをチューニングして照会処理を改善するには、以下のようにします。

手順

1. チューニングする SQL 照会をフェデレーテッド・データベースにサブミットします。
2. 照会が評価されている場所を分析します。
3. アクセス・プラン内でなされた決定の理由を調べ、プッシュダウンの機会を増やすようにシステムを変更します。

フェデレーテッド・パフォーマンスに関する資料

パフォーマンスのチューニングに関する詳細情報を記載したさまざまな IBM 資料を参照することができます。

- Asynchronous execution of federated queries in WebSphere Federation Server V9.1 (http://www.ibm.com/developerworks/db2/library/techarticle/dm-0611norwood/)
- Maximize the performance of WebSphere Information Integrator with Materialized Query Tables (http://www.ibm.com/developerworks/db2/library/techarticle/dm-0605lin/index.html)
- Performance enhancements in IBM WebSphere Federation Server V9.1, Part 1: Improve performance of federated queries with new WebSphere Federation Server capabilities (http://www.ibm.com/developerworks/db2/library/techarticle/dm-0612englert/)
- Performance enhancements in IBM WebSphere Federation Server V9.1, Part 2: Performance characteristics of new functionality in WebSphere Federation Server (http://www.ibm.com/developerworks/db2/library/techarticle/dm-0612englert2/)

- Using data federation technology in IBM WebSphere Information Integrator: Data federation usage examples and performance tuning (http://www.ibm.com/developerworks/db2/library/techarticle/dm-0507lin/)
- Parallelism in WebSphere Information Integrator V8.2 (http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0502harris/)
- Data Federation with IBM DB2 Information Integrator V8.1 (http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg247052.html?Open)
- "Using the federated database technology of IBM DB2 Information Integrator" (ftp://ftp.software.ibm.com/software/data/pubs/papers/iifed.pdf)

照会分析

照会処理の重要な部分として、最適パフォーマンスを得るために照会を調整する方法を決める分析があります。

データ・ソースからデータを入手するために、クライアント (ユーザーおよびアプリケーション) は、 SQL の照会をフェデレーテッド・データベースにサブミットします。すると、SQL コンパイラーはグローバル・カタログとデータ・ソース・ラッパー内の情報を検討し、照会の処理に役立てます。これには、データ・ソースへの接続についての情報、サーバー属性、マッピング、索引情報、およびニックネーム統計情報が含まれます。

SQL コンパイラー処理の一部として、*照会オプティマイザー* は照会を分析します。コンパイラーは、*アクセス・プラン* と呼ばれる、照会を処理するための代替ストラテジーを作成します。アクセス・プランは、照会について次のような処理方法を要求します。

- 照会をデータ・ソースが処理する
- 照会をフェデレーテッド・サーバーが処理する
- 照会の一部をデータ・ソースが処理し、一部をフェデレーテッド・サーバーが処理する

フェデレーテッド・データベースは、主にデータ・ソースの能力およびデータの属性に関する情報を基にアクセス・プランを評価します。この情報はラッパーとグローバル・カタログにあります。フェデレーテッド・データベースは照会を*照会フラグメント* と呼ばれるセグメントに分解します。通常、照会フラグメントをデータ・ソースにプッシュダウン した方が、より効率的です (データ・ソースがフラグメントを処理できる場合)。しかし、照会オプティマイザーは次のような他の要素も考慮します。

- 処理する必要のあるデータの量。
- データ・ソースの処理速度。
- フラグメントが戻すデータの量。
- 通信の帯域幅。

プッシュダウン分析は、リレーショナル・データ・ソースについてのみ行われます。非リレーショナル・データ・ソースは、request-reply-compensate プロトコルを使用します。

次の図は、SQL コンパイラーが照会を処理する時に実行するステップを示しています。

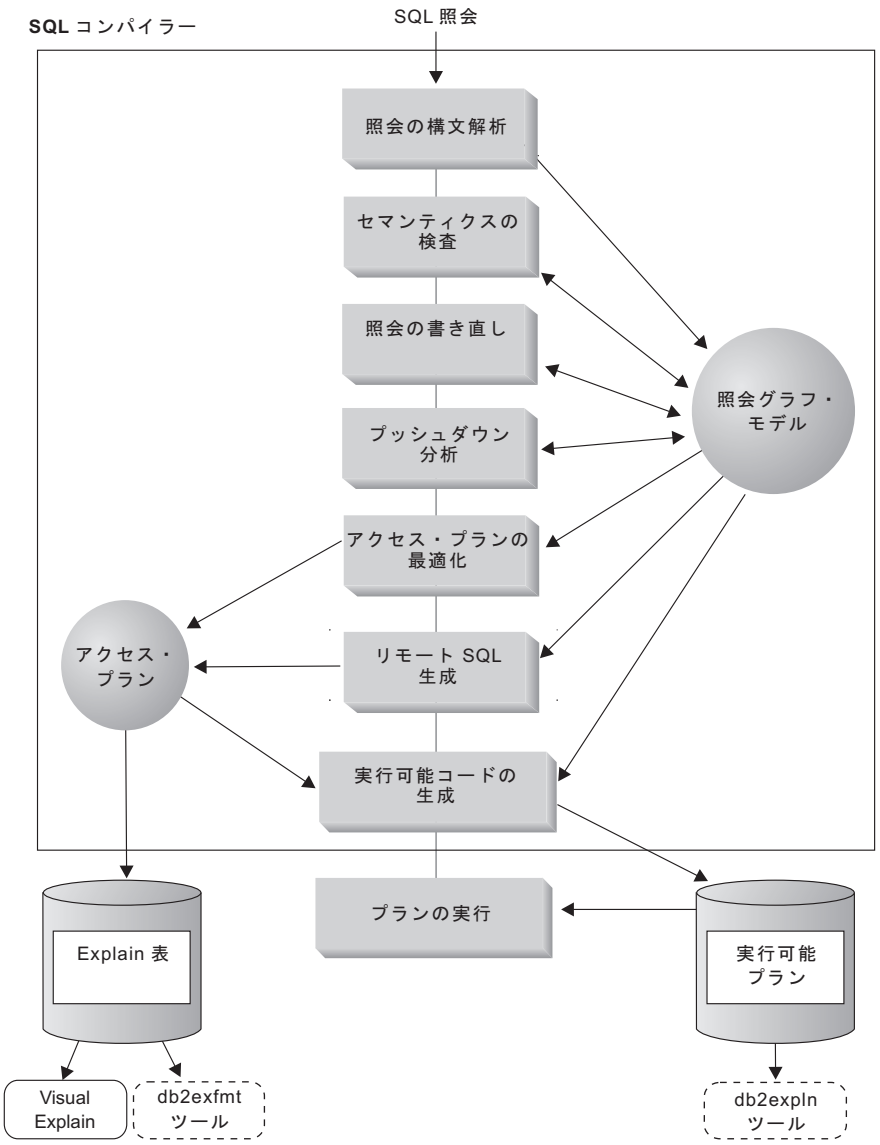


図 9. SQL コンパイラーの照会分析フローチャート

照会オプティマイザーは、照会フラグメントを処理するためのローカルとリモートのアクセス・プランを、リソースのコストに基づいて生成します。それから、フェデレーテッド・データベースは最小のリソース・コストで照会を処理すると思われるプランを選択します。

何らかのフラグメントをデータ・ソースで処理する場合、フェデレーテッド・サーバーはそれらのフラグメントをデータ・ソースにサブミットします。データ・ソースがフラグメントを処理した後、結果が取り出されてフェデレーテッド・サーバーに戻されます。フェデレーテッド・データベースが処理の一部を実行した場合、その処理結果とデータ・ソースから取り出した結果が組み合わされます。それから、フェデレーテッド・サーバーは結果をクライアントに戻します。

プッシュダウン分析の主な作業は、リモートで評価できる操作を判別することです。プッシュダウン分析は、受け取る SQL ステートメント、およびリモート・データ・ソースの能力およびセマンティクスに関する知識を基にこれを行います。照会オプティマイザーはこの分析に基づいて代案を評価し、コストに基づいてアクセス・プランを選択します。オプティマイザーは、コスト効率が悪いという理由から、リモート・データ・ソースで直接操作を実行するという選択をしない可能性もあります。次に行うのは、フェデレーテッド・サーバーとデータ・ソースの間のセマンティクスおよび SQL 操作の差を補正するために照会の再作成を試行するという作業です。これにより照会はさらに最適化されます。

オプティマイザーが選択する最終的なアクセス・プランには、リモート・データ・ソース側で評価される操作が含まれる場合があります。リモート側で実行されるそれらの操作について、SQL コンパイラーは生成段階で、リモート・データ・ソースの SQL ダイアレクト内のフレーズとして、効率的な SQL を作成します。すべてのソースを考慮に入れた最適化照会プランの生成プロセスを*グローバル最適化* と呼びます。

非リレーショナルのソースの場合、ラッパーは request-reply-compensate プロトコルを使用します。

プッシュダウン分析

プッシュダウン分析は、リモート・データ・ソースがある操作を実行できるかどうかを照会オプティマイザーに知らせます。操作 とは 1 つの機能であり、例えばリレーショナル演算子、システム関数やユーザー関数、または SQL 演算子 (GROUP BY、ORDER BY など) です。それから、オプティマイザーは演算子をプッシュダウンするかどうかに関して、コストに基づいた決定を行います。プッシュダウン分析により特定の操作をリモート・ソースで実行できると判断される場合でも、それをフェデレーテッド・サーバーのローカルで実行したほうがリソース消費量が少ないと思われる場合には、オプティマイザーはそうすることにするかもしれません。

プッシュダウン分析は、リレーショナル・データ・ソースについて行われます。非リレーショナル・ソースは、request-reply-compensate プロトコルを使用します。

プッシュダウンできない関数の場合、照会のパフォーマンスはかなり悪化します。例えば、選択述部を、リモート・データ・ソースではなく、ローカルで評価しなければならない場合の影響を考えてみてください。ローカルで評価するとなれば、フェデレーテッド・サーバーはリモート・データ・ソースから表全体を検索し、それから述部を使用して表をローカルでフィルターに掛ける必要が生じます。ネットワークが制約されており、表が大きい場合、照会のパフォーマンスに悪影響がある場合があります。

プッシュダウンできない演算子の場合も、照会のパフォーマンスはかなり悪化します。例えば、リモート・データを集約する GROUP BY 演算子があり、これをローカルで行うとなると、ここでもフェデレーテッド・サーバーはリモート・データ・ソースから表全体を検索する必要が生じます。

例えば、ニックネーム EMP が表 EMPLOYEE を指すとします。この表には 10,000 の行があります。 1 つの列には郵便番号が入っており、別の列には各従業員の給与

が入っています。給与額が 50,000 を超え、かつ特定の郵便番号の範囲に住む、都市ごとの従業員の数を計算するための、次のような照会がフェデレーテッド・サーバーに送られます。

```
SELECT CITY, COUNT(*) FROM EMP
 WHERE ZIP BETWEEN 'CA1' AND 'CA5' AND SALARY > 50000
 GROUP BY CITY;
```

SQL コンパイラーはこのステートメントを受け取ると、次のようないくつかの可能性を検討します。

- データ・ソースとフェデレーテッド・サーバーの照合シーケンスが同じである。どちらの述部もデータ・ソースからフェデレーテッド・サーバーに送られる中間結果セットのサイズを小さくする可能性があるため、両方がプッシュダウンされることもあり得ます。通常、表全体をフェデレーテッド・サーバーにコピーして、操作をローカルで実行するよりも、データ・ソース側でフィルターに掛け、結果をグループ化した方がより効率的です。プッシュダウン分析は、操作をデータ・ソース側で実行できるかどうかを判断します。照合シーケンスが同じならば、述部および GROUP BY 操作はデータ・ソース側で行うことができます。
- 照合シーケンスは同じであるが、照会オプティマイザーはフェデレーテッド・サーバーが非常に高速であることを知っている。 GROUP BY 操作をローカルで実行した方がよい (最低コスト) と、照会オプティマイザーが判断することはありえます。述部はデータ・ソースにプッシュダウンして評価されます。これは、グローバルな最適化と組み合わされたプッシュダウン分析の例です。
- 照合シーケンスが同じではない。 SALARY 述部がデータ・ソースにプッシュダウンされるのはほぼ間違いないでしょう。照合シーケンスに関係なく数値列は同じようにソートされるからです。しかし、ZIP の述部は文字列の順序に依存するため、プッシュダウンされません。 ZIP と SALARY の両方の述部がプッシュダウンされるのでない限り、GROUP BY もプッシュダウンされません。

SQL コンパイラーは使用できるアクセス・プランを検討し、最も効率的なプランを選択します。

要約すれば、目的は、関数および演算子をデータ・ソースにプッシュダウンして評価することを、照会オプティマイザーが考慮するようにすることです。関数または SQL 演算子をリモート・データ・ソース側で評価できるかどうかについては、多くの要因が影響します。照会オプティマイザーに影響を与える重要な要因は、サーバーの特性、ニックネームの特性、および照会の特性です。

プッシュダウンの可否に影響を与えるサーバー特性

プッシュダウンに影響を与えるサーバー特性には、SQL サポート、照合シーケンス、フェデレーテッド・サーバー・オプション、タイプおよび関数のマッピングが含まれます。

非リレーショナル・データ・ソースの場合、プッシュダウンの可否に影響を与える要因は、リレーショナル・データ・ソースのプッシュダウンの可否に影響を与える要因とは異なります。ほとんどの非リレーショナル・データ・ソースの場合、SQL ダイアレクトは要因となりません。非リレーショナル・データ・ソースは SQL を使用しないからです。

以下のトピックでは、プッシュダウンの可否に影響を与える、データ・ソース特有の要因について説明します。

SQL の相違

プッシュダウンに影響を与える SQL 特性には、SQL の機能、制約事項、限界、およびサーバー固有の SQL があります。

- SQL の機能。それぞれのデータ・ソースは、SQL ダイアレクトのバリエーションをサポートし、いろいろなレベルの機能をサポートします。例えば、GROUP BY リストを考えてみましょう。ほとんどのデータ・ソースは GROUP BY 演算子をサポートします。しかし、ある種のデータ・ソースでは、 GROUP BY リスト上の項目の数に制限があったり、または GROUP BY リストに式を許すかどうかにも制約があります。リモート・データ・ソース側で制約がある場合、フェデレーテッド・サーバーは GROUP BY の操作をローカルに実行する可能性があります。
- SQL の制約事項。それぞれのデータ・ソースに、いろいろな SQL 制約事項がある場合があります。例えば、ある種のデータ・ソースでは、リモート SQL ステートメントに値をバインドするために、パラメーター・マーカーを必要とするものがあります。したがって、各データ・ソースがこのようなバインド・メカニズムをサポートできるかを確認するため、パラメーター・マーカー制約をチェックする必要があります。機能のために値をバインドするよい方法をフェデレーテッド・サーバーが見つけられない場合、この機能はローカルで評価しなければなりません。
- SQL の限界。フェデレーテッド・サーバーは、リモート・データ・ソースよりも大きな整数を使用できる場合があります。しかし、データ・ソースに送信されるステートメントには、限界を超える値は組み込めません。したがって、この定数を操作する関数または演算子は、ローカルで評価する必要があります。
- サーバーの特性。いくつかの要因がこのカテゴリーに含まれます。 1 つの例は、NULL 値のソートです (配列次第で最高か、最低か)。例えば、データ・ソース側で NULL 値がフェデレーテッド・サーバーとは異なるソートをされる場合、NULL 可能な式の ORDER BY 操作はリモートでは評価できません。

フェデレーテッド・システムの VARCHAR2 データ・タイプ

照会処理を調整するには、アクセス・プランは VARCHAR2 データのブランク埋め込みを処理しなければなりません。

VARCHAR2_COMPAT サーバー・オプションを使用して、VARCHAR2 互換データ・ソースのサポートを有効にします。

VARCHAR2 互換セマンティクスは、フェデレーテッド・サーバーおよびデータ・ソースが、その一方、または両方が VARCHAR2 互換であるかどうかに基づいて、VARCHAR2 データ・タイプを処理する方法を決定します。システムは、次のように構成することが可能です。

- フェデレーテッド・サーバーは VARCHAR2 互換であるが、VARCHAR2 互換ではないデータ・ソースに接続している。
- フェデレーテッド・サーバーは VARCHAR2 互換ではないが、VARCHAR2 互換であるデータ・ソースに接続している。

- フェデレーテッド・サーバーとデータ・ソースはいずれも VARCHAR2 互換である。

フェデレーテッド・サーバーまたは DB2 データ・ソースが VARCHAR2 互換である場合、VARCHAR2 データ・タイプは VARCHAR データ・タイプと同義として処理されます。

ヒント: パフォーマンスを確実に改善するには、VARCHAR2 互換のデータ・ソースのみに接続するシステムでは、フェデレーテッド・サーバーも VARCHAR2 互換にします。

以下のシナリオでは、フェデレーテッド・サーバーは、VARCHAR2 互換セマンティクスに基づいて、空ストリング値を処理し、演算をデータ・ソースにプッシュダウンします。

シナリオ 1

フェデレーテッド・サーバーは VARCHAR2 互換ではないが、VARCHAR2 互換であるデータ・ソースに接続しています。空ストリングは、フェデレーテッド・サーバーでは許可されますが、データ・ソースでは許可されません。したがって、データ・ソースにプッシュダウンされるすべての空ストリングは、NULL 値に変換されます。

フェデレーテッド・サーバーのデフォルトの振る舞いでは、結果をデータ・ソースにプッシュダウンする前に、CHAR データ・タイプに対してブランク埋め込みセマンティクスを使用します。しかし、フェデレーテッド・サーバーは、空ストリングだけを指定する演算に対してはブランク埋め込みを使用しません。

例

- 以下の、空ストリングのみを含む INSERT および UPDATE ステートメントは、ブランク埋め込みされません。ステートメントは空ストリング値をプッシュダウンし、データ・ソースはこの値を NULL 値に変換します。

  例えば、以下のステートメントはフェデレーテッド・サーバーによってブランク埋め込みされません。

  ```
  INSERT INTO n1 (c1) VALUES ('')
  UPDATE n1 SET c1=''
  INSERT INTO n1 (c1) VALUES (''), ('')
  ```

- 以下の INSERT ステートメントは、空ストリングの代わりに 10 個のブランクをデータ・ソースに送信します。

  ```
  INSERT INTO n1 (col_char) VALUES (''), ('ibm')
  ```

シナリオ 2

フェデレーテッド・サーバーは VARCHAR2 互換であるが、VARCHAR2 互換ではないデータ・ソースに接続しています。フェデレーテッド・サーバーの VARCHAR2 互換セマンティクスは、空ストリングを許可せず、ブランク埋め込みセマンティクスも使用しません。しかし、データ・ソースのセマンティクスは、空ストリングを許可し、ブランク埋め込みセマンティクスを使用します。したがっ

て、フェデレーテッド・サーバーは、そのセマンティクスおよびデータ整合性が保持されている場合を除いて、空ストリングを含む演算がプッシュダウンされることを制限します。

例

- 以下の INSERT ステートメントは、空ストリングが NULL 値に変換されて、ローカルでは処理できないため、データ・ソースにプッシュダウンされます。

  ```
  INSERT INTO n1 (c1) VALUES ('')
  ```

- 以下の INSERT ステートメントは、2 つの別個のステートメントに分割すれば、データ・ソースにプッシュダウンされます。

  ```
  INSERT INTO n1 SELECT * FROM n2
  ```

  VARCHAR2 互換ではないデータ・ソースの表をターゲットまたはソースとしている場合、以下の例のように 2 つのステートメントを使用する必要があります。

  ```
  SELECT * FROM n2
  INSERT INTO n1 VALUES (:H0)
  ```

- 以下の例のように式の内部にネストされている関数は、フェデレーテッド・サーバーでローカルに実行されます。

  ```
  SELECT LENGTH(TRIM(c1)) FROM n1
  ```

- データ・ソースで比較演算が実行されると結果に矛盾が生じる可能性があるため、すべての比較演算はフェデレーテッド・サーバーでローカルに実行されます。

シナリオ 3

フェデレーテッド・サーバーとデータ・ソースはいずれも VARCHAR2 互換です。フェデレーテッド・サーバーもデータ・ソースも空ストリングを許可せず、ブランク埋め込みセマンティクスを使用しません。したがって、すべての空ストリングを含む演算は、セマンティクスが保持されているため、データ・ソースにプッシュダウンされます。

照合シーケンス

COLLATING_SEQUENCE サーバー・オプションを 'Y' にすると、データ・ソースの照合シーケンスがフェデレーテッド・サーバーの照合シーケンスと一致することをフェデレーテッド・データベースに知らせることになります。この設定値により、オプティマイザーは順序に依存する処理のデータ・ソース側へのプッシュダウンを検討することができるようになり、パフォーマンスが向上する可能性があります。

データ・ソースの照合シーケンスがフェデレーテッド・データベースの照合シーケンスと同じでなく、COLLATING_SEQUENCE サーバー・オプションを「Y」に設定した場合は、誤った結果を受け取る可能性があります。例えば、プランがマージ結合を使用する場合、オプティマイザーは順序付けの操作をデータ・ソース側にプッシュダウンする場合があります。データ・ソースの照合シーケンスが同じでない場合、結合の結果は正しい結果セットになりません。データ・ソースの照合シーケン

スがフェデレーテッド・データベースの照合シーケンスと一致するかどうか確かでない場合は、COLLATING_SEQUENCE サーバー・オプションを「N」にしてください。

あるいは別の方法として、データ・ソースが使用する照合シーケンスと同じ照合シーケンスをフェデレーテッド・データベースが使用するように構成することができます。その後、COLLATING_SEQUENCE サーバー・オプションを 'Y' にします。これにより、オプティマイザーは、文字列での順序に依存する操作をプッシュダウンすることを検討できるようになります。

データ・ソースとフェデレーテッド・データベースの照合シーケンスが同じかどうかを判断するには、以下の要因を検討してください。

- 各国語サポート

  照合シーケンスは、サーバーでサポートされる言語に関係しています。ご使用のオペレーティング・システムのフェデレーテッド・データベース NLS 情報とデータ・ソース側の NLS 情報を比較してください。
- 言語認識照合

  フェデレーテッド・データベースまたはデータ・ソースで言語認識照合を使用するかどうかを確認します。別の照合を使用する場合は、COLLATING_SEQENCE を Y に設定しないでください。
- データ・ソースの特性

  データ・ソースによっては、大/小文字を区別しない照合シーケンスを使用して作成されており、その場合、順序に関係する操作では、フェデレーテッド・データベースとは異なる結果になることがあります。
- カスタマイズ

  あるデータ・ソースでは、照合シーケンスに複数のオプションを提供したり、照合シーケンスをカスタマイズできたりします。

フェデレーテッド・サーバーからの照会フラグメントがソートを必要とする場合、どこでソートを処理するかは、いくつかの要因によって決まります。フェデレーテッド・データベースの照合シーケンスがデータ・ソース側の照合シーケンスと同じ場合、ソートはデータ・ソース側またはフェデレーテッド・サーバー側で行うことができます。照会オプティマイザーは、ローカル・ソートまたはリモート・ソートのどちらが照会を実行する最も効率的な方法であるかを判別することができます。

通常、数値比較は、照合シーケンスが異なる場合でも、どちらのロケーションでも実行できます。ただし、NULL 文字の重み付けがフェデレーテッド・データベースとデータ・ソースで異なる場合は、誤った結果になる場合があります。

同様に、比較ステートメントの場合、大/小文字を区別しないデータ・ソースにステートメントをサブミットする場合は注意してください。大/小文字を区別しないデータ・ソースでは、文字 "I" と "i" に割り当てられる重みは同じです。例えば、英語コード・ページの大/小文字を区別しないデータ・ソースでは、 **STEWART**、**SteWArT**、**stewart** はすべて同じと見なされます。フェデレーテッド・データベースは、デフォルトでは大文字小文字を区別し、これらの文字に異なる重みを割り当てます。

フェデレーテッド・データベースとデータ・ソースの照合シーケンスが異なる場合、フェデレーテッド・サーバーはデータをフェデレーテッド・データベースに取り出し、ローカルでソートを行えるようにします。その理由は、ユーザーは、フェデレーテッド・サーバーに定義された照合シーケンスに従って並べられた照会結果を見ることを期待しているからであり、データをローカルで並べ替えることにより、フェデレーテッド・サーバーはこの期待に答えることができるからです。

照会において文字の列に等価述部が含まれている場合は、照合シーケンスが異なっていても ('N' に設定されている)、照会のその部分をプッシュダウンすることができます。例えば、述部 C1 = 'A' は、照合シーケンスに関係なく同じ値を検索するので、フェデレーテッド・サーバーとは異なる照合シーケンスを持つデータ・ソース側にプッシュダウンすることができます。ただしこのような述部は、データ・ソース側の照合シーケンスが大/小文字を区別しない場合は (COLLATING_SEQUENCE='I')、プッシュダウンできません。データ・ソースが大/小文字を区別しない場合、C1= 'A' と C1 = 'a' の結果は同じです。これは DB2 Database for Linux, UNIX, and Windows などの大/小文字を区別する環境では整合性がありません。

管理者は、データ・ソースの照合シーケンスと一致する特定の照合シーケンスのフェデレーテッド・データベースを作成できます。この方法を用いると、すべてのデータ・ソースが同じ照合シーケンスを使用する場合、または、ほとんどあるいはすべての列関数が同じ照合シーケンスを使用するデータ・ソースに向けられている場合は、パフォーマンスが高速になります。例えば DB2 for z/OS では、ORDER BY 文節で定義されたソートは、EBCDIC コード・ページに基づく照合シーケンスで行われます。フェデレーテッド・サーバーを使用して、DB2 for z/OS データを ORDER BY 文節に従ってソートして検索したい場合、フェデレーテッド・データベースが、EBCDIC コード・ページに基づく事前定義の照合シーケンスを使用するように構成することをお勧めします。

フェデレーテッド・データベースとデータ・ソース側の照合シーケンスが異なる場合で、データ・ソース側の順序で並べられたデータを見る必要がある場合は、照会をパススルー・セッションでサブミットするか、またはデータ・ソースのビューで照会を定義します。

フェデレーテッド・サーバー・オプション

設定するサーバー・オプションにより、フェデレーテッド・サーバーがリモート・データ・ソースに関して持つ情報が絞り込まれます。

プッシュダウンの可否に影響を与える、これまでリストした要因は、データベース・サーバーの特性であり、これを変更することはできません。以下のサーバー・オプションを注意深く考慮することにより、照会のパフォーマンスを向上させることができます。

- COLLATING_SEQUENCE。データ・ソースが、フェデレーテッド・データベースの照合シーケンスとは異なる照合シーケンスを持つ場合、文字値に対する順序に依存する操作は、データ・ソース側でリモートで評価することはできません。例えば、照合シーケンスが異なるデータ・ソースにおいて、ニックネーム文字の列に対して MAX 列関数を実行する場合がそうです。 MAX 関数をリモート・データ・ソースで評価すると結果が異なる可能性があるので、フェデレーテッド・データベースは集約操作と MAX 関数をローカルで実行します。

- VARCHAR_NO_TRAILING_BLANKS。このオプションは、末尾ブランクを含まない可変長文字ストリング用です。ある種のデータ・ソース (Oracle など) は、フェデレーテッド・データベースが適用するようなブランク埋め込み比較セマンティクスを適用しません。この埋め込みの違いは、予期しない結果を引き起こす可能性があります。

  例:

  ```
  'HELLO' = 'HELLO         ' in DB2
  'HELLO' <> 'HELLO         ' in Oracle!
  ```

  末尾ブランクが Oracle データ・ソース上の VARCHAR 列に存在している場合、このオプションを N に設定する必要があります (Oracle のデフォルト)。フェデレーテッド・サーバーはセマンティクスにおける違いを補正しながら、整合した結果セットを保証する必要があるため、このオプションはパフォーマンスに影響を与えます。 Oracle データ・ソース列に末尾ブランクが含まれている場合にこの値を Y に設定すると、不整合な結果となる可能性があります。

  データ・ソースにあるすべての VARCHAR および VARCHAR2 の列に末尾ブランクが含まれていないことが確かである場合は、データ・ソースに対してこのサーバー・オプションを設定することを考慮してください。ニックネームを持つ可能性のあるすべてのオブジェクト (ビューを含む) を忘れずに考慮に入れてください。

  推奨: このオプションをそれぞれの列ごとに、VARCHAR_NO_TRAILING_BLANKS 列オプションを使用して設定してください。

- DB2_MAXIMAL_PUSHDOWN。このオプションは、照会オプティマイザーがアクセス・プランを選択するときに使用する 1 次基準を指定します。照会オプティマイザーは、コストを基に、またはリモート・データ・ソースによってできるだけ多くの照会処理を実行できるユーザー要件を基にアクセス・プランを選択できます。 DB2_MAXIMAL_PUSHDOWN を Y に設定すると、ネットワーク・トラフィックの減少が、照会オプティマイザーによってオーバーライドが行われる基準となります。照会オプティマイザーは、データ・ソースへの「送信」が最も少ないアクセス・プランを使用します。このサーバー・オプションを Y に設定すると、フェデレーテッド・サーバーが使用を強制されるアクセス・プランは、最小コストのプランとはならない場合があります。最小コストのプラン以外のアクセス・プランを使用すると、パフォーマンスが低下する可能性があります。DB2_MAXIMAL_PUSHDOWN サーバー・オプションが Y に設定されていると、デカルト積となる照会はリモート・データ・ソースにプッシュダウンされません。デカルト積になる照会は、フェデレーテッド・データベースによって処理されます。フェデレーテッド・サーバーに照会の処理をリモート・データ・ソースにプッシュダウンさせるためには、 DB2_MAXIMAL_PUSHDOWN サーバー・オプションを Y に設定することは必須ではありません。このサーバー・オプションを N に設定しても (デフォルト)、照会オプティマイザーはデータ・ソースに照会の処理をプッシュダウンします。しかし、オプションが N に設定されている場合、オプティマイザーは 1 次基準として、ネットワーク・トラフィックではなくコストを使用します。

さらに、 283 ページの『グローバルな最適化に影響を与えるサーバー特性』では、照会のパフォーマンスに影響を与える可能性がある COMM_RATE、CPU_RATIO、および IO_RATIO サーバー・オプションについて説明しています。

ステートメント・レベル最適化ガイドラインの作成

ステートメント・レベルのサーバー・オプションのサポートにより、最適化プロファイルを使用して、単一 SQL ステートメントに影響を与えるサーバー・オプションを設定できます。

このタスクについて

最適化プロファイルとは、1 つ以上の SQL ステートメントについての最適化パラメーターを指定できる XML 文書です。最適化プロファイルのコンテンツを、最適化プロファイル・スキーマに定義します。最適化プロファイルおよびガイドラインの詳細については、最適化プロファイルとガイドライン (Optimization profiles and guidelines) を参照してください。

いずれのフェデレーション・サーバー・オプションも最適化プロファイルに指定できます。最適化プロファイルに定義されたガイドラインは DB2 オプティマイザーに影響を与え、フェデレーション照会処理のパフォーマンスと効率を向上させることができます。

最適化プロファイルに、単一 SQL ステートメントに対するステートメント・レベルのサーバー・オプションを設定できます。

制約事項: ステートメント・レベルのサーバー・オプションは DB2_MAXIMAL_PUSHDOWN オプションと COLLATING_SEQUENCE オプションに限定されています。

手順

SYSTOOLS.OPT_PROFILE 表に、最適化プロファイルを定義する項目を作成します。
SYSTOOLS.OPT_PROFILE 表には、すべての最適化プロファイルが含まれています。

例

SYSTOOLS.OPT_PROFILE に項目を作成します。

```
drop table SYSTOOLS.OPT_PROFILE;
call sysinstallobjects('opt_profiles', 'c', '', '');
```

最適化プロファイルを編集して、ステートメント・プロファイルを作成します。最適化ガイドラインでは、以下に太字で示すように、ステートメント・キーの後にサーバー・オプションを指定します。

```
<xml version="1.0" encoding="UTF-8"?>
<OPTPROFILE VERSION="10.0.1">
<STMTPROFILE ID="QRY1">
<STMTKEY>
<![CDATA[select * from NICK1 as n1, NICK2 as n2 where n1.c2 = n2.c1 and n1.c2 > 'a']]>
</STMTKEY>
<OPTGUIDELINES>
<SERVEROPTIONS>
```

```
 <SERVER NAME="DATASTORE2">
  <OPTION NAME="DB2_MAXIMAL_PUSHDOWN" VALUE="Y">
  <OPTION NAME="COLLATING_SEQUENCE" VALUE="Y"><
  </SERVER>
  <SERVER NAME="DATASTORE1">
  <OPTION NAME="DB2_MAXIMAL_PUSHDOWN" VALUE="N"/>
 </SERVER>
</SERVEROPTIONS>
</OPTGUIDELINES>
</STMTPROFILE>
</OPTPROFILE>
```

最適化プロファイルにはグローバル・ガイドラインも含まれ、プロファイルが有効な間実行される SQL ステートメントすべてに適用されます。

タイプ・マッピングと関数マッピングの要因

デフォルトのデータ・タイプ・マッピングとデフォルトの関数マッピングは、データ・ソース・ラッパー内に組み込まれています。データ・タイプ・マッピングは、データ・ソースのデータ・タイプとフェデレーテッド・サーバーのデータ・タイプとの間の関係を記述します。デフォルトのデータ・タイプ・マッピングからカスタマイズすることができます。関数マッピングは、データ・ソース関数と、フェデレーテッド・サーバー側の意味的に同等の関数との間の関係を記述します。場合によっては、フェデレーテッド・データベースは、データ・ソース側でサポートされない関数マッピングを補います。

これらのデフォルトのデータ・タイプ・マッピングは、実行時のバッファー・オーバーフローおよび切り捨てを避けるため、それぞれのデータ・ソース・データ・タイプに十分なバッファー・スペースが与えられるように設計されています。特定のアプリケーションに合わせ、場合によってはパフォーマンスを向上させるため、特定のデータ・ソース用または特定のニックネーム用にタイプ・マッピングをカスタマイズすることができます。例えば、Oracle の DATE タイプには日付とタイム・スタンプ部分を含めることができ、デフォルトで DB2 TIMESTAMP にマップされます。 Oracle 日付列にアクセスする場合、その列に日付部分しか含まれていない (タイム・スタンプはない) ことが分かっており、date_compat サーバー・オプションがオフになっていれば、ALTER NICKNAME ステートメントを使用して、ニックネームのローカル・データ・タイプを TIMESTAMP から DATE に変更できます。SalesDate=DATE('2009-01-04') のように、日付だけに基づいて述部を評価する場合、この変更により、列で保持されている日付とタイム・スタンプの情報から SCALAR 関数を使用して日付を抽出する工程がバイパスされるので、パフォーマンスが向上する可能性があります。

フェデレーテッド・データベースは、データ・ソース側でサポートされない関数を補います。関数を補うことには、通常は必要なデータをデータ・ソースから検索し、関数をローカルに適用することが関係しますが、これは多くの場合パフォーマンスに影響を与えます。関数が補われるのは、以下のいくつかの場合です。

- 単に関数がデータ・ソース側に存在しない。例えば、SYSFUN 関数のいくつかは、DB2 for z/OS データ・ソースには存在しないので、ローカルで補う必要があります。

- データ・ソース側に関数があるが、オペランドの特性が関数の制限に違反する。その例は、IS NULL リレーショナル演算子です。ほとんどのデータ・ソースはこれをサポートしますが、あるものは、IS NULL 演算子の左側にのみ列名を許すといった制約があります。
- 関数をリモート側で評価すると異なる結果を戻す可能性がある。この例は、'>' (より大) 演算子です。照合シーケンスが異なるデータ・ソースの場合、「より大」演算子は、フェデレーテッド・データベースでローカルに評価した結果とは異なる結果を戻す可能性があります。

プッシュダウンの可否に影響を与えるニックネーム特性

プッシュダウンに影響を与えるニックネーム特性には、ニックネーム列のローカル・データ・タイプ、フェデレーテッド列オプション、マテリアライズ照会表が含まれます。

プッシュダウンの可否に影響を与える、いくつかのニックネーム特有の要因があります。ニックネーム列のローカルのデータ・タイプは、オプティマイザーが評価する結合シーケンスの可能性の数に影響します。ニックネームには、列に末尾ブランクを含まないことを示す列オプションを指定することができます。これにより SQL コンパイラーは、データ・ソースに送信される SQL ステートメントの述部をより効率的な形で生成することが可能になります。

ニックネーム列のローカル・データ・タイプ

列のローカル・データ・タイプにより、述部をデータ・ソース側で評価することが妨げられないようにしてください。

デフォルトのデータ・タイプ・マッピングは、オーバーフローを防ぐために提供されています。ただし、異なるデータ・タイプまたは異なる長さの 2 つの列の間で述部を結合すると、ハッシュ結合技法はオプティマイザーにより考慮されなくなります。オプティマイザーがハッシュ結合を考慮するには、結合列のデータ・タイプと長さの両方が正確に一致している必要があります。例えば、整数値のみを保持するように設計された Oracle データ・ソース列は、Oracle データベース内ではよく NUMBER として作成され、このデフォルトは NUMBER (38) になります。この Oracle データ・タイプのニックネーム列には、ローカル・データ・タイプ FLOAT が与えられますが、その理由は、DB2 整数の範囲はおよそ NUMBER(9) と等しいからです。この場合、DB2 整数列と NUMBER (ただし整数値のみを保持する) として定義される Oracle 整数列との結合は、Oracle 列が FLOAT タイプとしてマップされるので、ハッシュ結合技法を使用できません。ただし、この Oracle NUMBER 列の定義域を DB2 INTEGER データ・タイプに収容できるならば、ALTER NICKNAME ステートメントを使用して、そのローカル・データ・タイプを変更することができます。次いでオプティマイザーはハッシュ結合技法を考慮でき、これによりパフォーマンスが向上する場合があります。

フェデレーテッド列オプション

アクセス・プランを開発するために照会オプティマイザーが使用する、フェデレーテッド列オプションを定義することができます。

列オプションは、列内のデータを通常とは異なる扱いをするようにラッパーに指示します。 SQL コンパイラーと照会オプティマイザーは、メタデータを使用して、データにアクセスするためのよりよいプランを作成します。フェデレーテッド・データベースは、ニックネームが指すオブジェクトをあたかも表のように扱います。したがって、ニックネームを作成する任意のデータ・ソース・オブジェクトに対して、列オプションをセットすることができます。

ALTER NICKNAME ステートメントを使用して、ニックネームに列オプションを追加、または変更することができます。次の 2 つの列オプションがあります。

- NUMERIC_STRING。この列オプションは、文字タイプの列 (CHAR および VARCHAR) に適用されます。フェデレーテッド・データベースの照合シーケンスとは異なる照合シーケンスを持つデータ・ソースがあるとします。フェデレーテッド・サーバーは、文字データを含む列をデータ・ソース側でソートすることはありません。データはフェデレーテッド・データベースに戻され、ローカルにソートが行われます。しかしここで、列が文字データ・タイプであり、数字 ('0'、'1'、...、'9') だけが入っているとします。これは、NUMERIC_STRING 列オプションを 'Y' にすれば示すことができます。これにより、数値が文字ストリングとして表される場合でも、照合シーケンスに関係なく必ず同じようにソートされるので、照会オプティマイザーは、オプションでデータ・ソース側でソートを実行できるようになります。ソートをリモート側で実行できれば、データをフェデレーテッド・サーバーに持ってきて、ソートをローカルで実行するというオーバーヘッドが避けられます。
- VARCHAR_NO_TRAILING_BLANKS。同じ名前のサーバー・オプションとは異なり、この列オプションは、末尾ブランクを含まない特定の Oracle 列を識別するために使用できます。 SQL コンパイラーのプッシュダウン分析ステップは、この設定値を持つ列に対して実行されるすべての操作をチェックする際に、この情報を考慮に入れます。 VARCHAR_NO_TRAILING_BLANKS 設定値に基づき、SQL コンパイラーは、データ・ソースに送信されるリモート SQL ステートメント内で使用される述部に、意味的に同等ではあるが異なる形の述部を生成する場合があります。値 'Y' を指定すると、照会のパフォーマンスを向上させることができるリモート索引 (選択可能な場合) が使用可能になります。

プッシュダウンの可否に影響を与える照会の特性

複数のデータ・ソースを参照する SQL 演算子はプッシュダウンに影響を与えます。

照会は、複数のデータ・ソースからのニックネームを含む SQL 演算子を参照することができます。フェデレーテッド・サーバーが、1 つの演算子を使用して 2 つの参照されたデータ・ソースからの結果を結合する場合、その操作はフェデレーテッド・サーバーで行う必要があります。この例は UNION のようなセット演算子です。 この演算子は、リモート・データ・ソースで直接評価することはできません。

照会を評価する場所の分析

詳細な照会オプティマイザーの情報は、実際のアクセス・プラン自体とは別に Explain 表に保持されます。この情報により、アクセス・プランを詳細に分析できます。フェデレーテッド・アクセス・プランの SHIP 演算子を調べることにより、どの SQL 演算子がデータ・ソースにプッシュダウンされ、どの演算子がフェデレーテッド・サーバーで実行されたかを判別することができます。

Explain 表は、サポートされるすべてのオペレーティング・システムでアクセスでき、静的と動的の両方の SQL ステートメントの情報を含んでいます。通常、Explain 表からアクセス・プラン情報を入手するために次のツールを使用します。

- Explain 表フォーマット・ツール。db2exfmt ツールを使用して、 Explain 表からの情報を事前定義されたフォーマットで表示します。
- db2expln および dynexpln ツール。これらのツールを使用すると、特定の SQL ステートメント用に選択されたアクセス・プランを知ることができます。動的 SQL ステートメントと静的 SQL ステートメントはどちらも Explain 機能を使って説明できます。 Explain ツールとの唯一の違いは、 Visual Explain では Explain 情報がグラフィック形式で表示されるという点です。それ以外は、2 つの方式で提供される詳細レベルはどちらも同じです。 db2expln と dynexpln の出力を最大限活用するには、以下の点を理解しておく必要があります。
  - サポートされている別の SQL ステートメント、およびそのステートメントに関連した用語 (SELECT ステートメントの述部など)
  - パッケージ (アクセス・プラン) の目的
  - システム・カタログ表の目的と内容
  - 一般的なアプリケーションのチューニングの概念

SQL ステートメントを使用して Explain 表にアクセスすることもできます。これにより、異なる照会を比較したり、同じ照会を長期にわたって比較する場合、出力の操作を簡単に行うことができます。

照会を DB2_MAXIMAL_PUSHDOWN サーバー・オプションを使用して評価する分析

DB2_MAXIMAL_PUSHDOWN サーバー・オプションを Explain ユーティリティーと共に使用して、コストに基づくオプティマイザー決定、またはプッシュダウンが不可能というプッシュダウン分析の判断により、特定の演算子がデータ・ソース側で実行するようにプッシュダウンされていなかったかどうかを判別します。

このタスクについて

手順

DB2_MAXIMAL_PUSHDOWN サーバー・オプションを使用して照会に対して Explain ツールを実行するには、次のようにします。

手順

1. DB2_MAXIMAL_PUSHDOWN サーバー・オプションを 'N' に設定します。これは、このオプションのデフォルト設定です。プッシュダウン分析は、プッシュダウンできる SQL の部分を判別します。その後、照会オプティマイザーは、プッ

シュダウン分析によって設定される基準を違反しないすべての代替プランを生成します。照会オプティマイザーは各プランのコストを見積もり、最も見積もりコストの低いプランを選択します。適切な SHIP 演算子の詳細を表示して、データ・ソースにプッシュダウンされた演算子を分析することができます。プッシュダウンされていると予期した演算子がプッシュダウンされていなかった場合は、ステップ 2 に進みます。

2. DB2_MAXIMAL_PUSHDOWN サーバー・オプションを 'Y' に設定します。Explain ツールを使用して、SQL ステートメントを再度分析します。 Explain 出力に表示されるプランは、データ・ソースにプッシュダウンできるすべての SQL 操作を表示します。
   - オプションを 'Y' にリセットした後に演算子がプッシュダウンされる場合、オプティマイザーは、演算子をリモートよりもローカルに実行するほうがコスト効率が良いと判断しています。オプションを 'Y' にリセットした後に演算子がプッシュダウンされない場合、プッシュダウン分析が演算子をリモートで実行することを許可しなかった可能性があります。
   - オプティマイザーが演算子をプッシュダウンしないというコストに基づく決定をした場合は、ニックネーム統計をチェックして、それらが正確であるかを確認することを考慮してください。プッシュダウン分析で演算子をプッシュダウンしないという決定がされた場合は、サーバー・オプション、データ・タイプ・マッピング、および関数マッピングのチェックを考慮してください。

アクセス・プランの評価決定の概要

このセクションのトピックでは、典型的なアクセス・プランの分析における質問、およびプッシュダウンの機会を増やすために調べることができる領域をリストしています。

なぜ、この述部はリモート側で評価されないのか ?

この質問は、述部が非常に選択的であり、行をフィルターに掛けてネットワーク・トラフィックを減らすことができる場合に生じます。リモートでの述部の評価は、同じデータ・ソースの 2 つの表の間の結合をリモートで評価できるかどうかにも影響します。

次のような調査領域があります。

- サーバー・オプション。サーバー・オプション COLLATING_SEQUENCE および VARCHAR_NO_TRAILING_BLANKS の設定は、どこで述部が評価されるかにどのように影響を与えるか ?
- 副照会の述部。この述部には、別のデータ・ソースに関する副照会が含まれているか? この述部には、このデータ・ソースではサポートされない SQL 演算子を含む副照会があるか? すべてのデータ・ソースが述部内のセット演算子をサポートするわけではありません。
- 述部の関数。この述部には、このリモート・データ・ソースでは評価できない関数が含まれているか? リレーショナル演算子は関数として分類されます。
- 述部のバインド要件。この述部をリモートで評価した場合、何らかの値のバインドインが必要か? そうならば、このデータ・ソースの SQL の制約に違反しないか?

- グローバルな最適化。オプティマイザーが、ローカル処理の方がよりコスト効果が高いと判断しました。

なぜ、GROUP BY 演算子はリモート側で評価されないのか ?

GROUP BY 演算子がリモート側で評価されない理由を判別するためにチェックできる、いくつかの領域があります。

チェックできる領域として、以下のものがあります。

- GROUP BY 演算子の入力はリモート側で評価されるか? 答えが「いいえ」の場合、入力を調べてください。
- データ・ソースはこの演算子に関して何らかの制約を持っているか? 例えば、次のような制約があります。
  - GROUP BY 項目の数の制限
  - 結合された GROUP BY 項目のバイト数の制限
  - GROUP BY リスト上のみの列指定
- データ・ソースはこの SQL 演算子をサポートするか?
- グローバルな最適化。オプティマイザーが、ローカル処理の方がよりコスト効果が高いと判断しました。

なぜ、SET 演算子はリモート側で評価されないのか ?

オペランドおよびデータ・ソースの制約事項をチェックして、SET 演算子がリモート側で評価されない理由を判別することができます。

考慮事項:

- そのオペランドは両方とも完全に同じリモート・データ・ソースで評価されるか? 「はい」でなければならないのに答えが「いいえ」の場合は、両方のオペランドを調べます。
- データ・ソースはこの SET 演算子に何らかの制約を持っているか? 例えば、この特定の SET 演算子に対して、ラージ・オブジェクトまたは長いフィールドは有効な入力か?

なぜ、ORDER BY 操作はリモート側で評価されないのか ?

操作への入力と文節の内容をチェックし、データ・ソースの制約事項をチェックして、ORDER BY 演算子がリモート側で評価されない理由を判別することができます。

考慮事項:

- サーバー・オプション。サーバー・オプション COLLATING_SEQUENCE および VARCHAR_NO_TRAILING_BLANKS の設定は、どこで述部が評価されるかにどのように影響を与えるか ?
- ORDER BY 操作の入力はリモート側で評価されるか? 答えが「いいえ」の場合、入力を調べてください。
- ORDER BY 文節は文字式を含むか? 「はい」ならば、リモート・データ・ソースの照合シーケンスはフェデレーテッド・サーバーの照合シーケンスと異なるか?

- データ・ソースはこの演算子に関して何らかの制約を持っているか? 例えば、ORDER BY 項目の数に制限はあるか? データ・ソースでは、 ORDER BY リストの列指定に制約があるか?

なぜ、リモート INSERT の全選択ステートメントは完全にリモート側で評価されないのか ?

副選択のいくつかの要素をチェックして、リモート INSERT の全選択ステートメントが完全にリモート側で評価されない理由を判別することができます。

考慮事項:

- 副選択はリモート・データ・ソース上で完全に評価できるか? 「いいえ」ならば、副選択を調べてください。
- 副選択はセット演算子を含むか? 「はい」ならば、このデータ・ソースは INSERT への入力としてセット演算子をサポートするか?
- 副選択はターゲット表を参照するか? 「はい」ならば、このデータ・ソースはこの構文を許すか?

なぜ、VALUES 文節のあるリモート INSERT ステートメントは完全にリモート側で評価されないのか ?

VALUES 文節および式をチェックして、VALUES 文節のあるリモート INSERT ステートメントが完全にリモート側で評価されない理由を判別することができます。

考慮事項:

- VALUES 文節はリモート・データ・ソース側で完全に評価できるか? 言い換えれば、リモート・データ・ソースではサポートされない関数が式に含まれていないか?
- 式はスカラー副照会を含むか? その構文はサポートされているか?
- 式はターゲット表を参照するか? その構文はサポートされているか?

なぜ、リモートの、検索された UPDATE ステートメントは完全にリモート側で評価されないのか ?

SET 文節および検索条件の要素をチェックして、リモートの、検索された UPDATE ステートメントが完全にリモート側で評価されない理由を判別することができます。

考慮事項:

- SET 文節はリモート・データ・ソース側で完全に評価できるか? 言い換えれば、リモート・データ・ソースではサポートされない関数が update 式に含まれていないか?
- SET 文節はスカラー副照会を含むか? このデータ・ソースはこの構文を許すか?
- 検索条件はリモート・データ・ソース側で完全に評価できるか? 答えが「いいえ」の場合、検索条件を調べてください。
- 検索条件または SET 文節はターゲット表を参照するか? このデータ・ソースはこの構文を許すか?

- 検索条件または SET 文節は相関を持つターゲット表を参照するか? このデータ・ソースはこの構文を許すか?

なぜ、位置指定の UPDATE ステートメントは完全にリモート側で評価されないのか ?

これは、フェデレーテッド・データベースが、UPDATE ステートメントをデータ・ソースに送信する前に、update 式をローカルで評価することを選択した場合に起こります。この方法はあまりパフォーマンスに影響を与えないはずです。

考慮事項:

- SET 文節はリモート・データ・ソース側で完全に評価できるか? 言い換えれば、リモート・データ・ソースではサポートされない関数が update 式に含まれていないか?
- SET 文節はスカラー副照会を含むか? このデータ・ソースはこの構文を許すか?

なぜ、リモートの、検索された DELETE ステートメントは完全にリモート側で評価されないのか ?

検索条件の要素をチェックして、リモートの、検索された DELETE ステートメントが完全にリモート側で評価されない理由を判別することができます。

考慮事項:

- 検索条件はリモート・データ・ソース側で完全に評価できるか? 答えが「いいえ」の場合、検索条件を調べてください。
- 検索条件はターゲット表を参照するか? このデータ・ソースはこの構文を許すか?
- 検索条件は相関を持つターゲット表を参照するか? このデータ・ソースはこの構文を許すか?

データ・ソースのアップグレードとカスタマイズ

データ・ソースをアップグレードまたはカスタマイズするときには、グローバル・カタログ情報を更新する必要があります。

SQL コンパイラーは、グローバル・カタログに保管されている、データ・ソースの SQL 機能についての情報に依存しています。この情報は定期的に更新する必要があります。データ・ソースの SQL 機能は、データ・ソースのバージョンが新しくなれば変わる可能性があります。データ・ソースがアップグレード、またはカスタマイズされた場合は、グローバル・カタログの情報を更新し、SQL コンパイラーが最新の情報を使用できるようにしてください。

カタログを更新するには、CREATE FUNCTION MAPPING および ALTER SERVER などの SQL DDL ステートメントを使用します。

関数テンプレートでの述部のプッシュダウン

フェデレーテッド・システムでは、それぞれのリモート・データ・ソースにはその固有の関数があります。これらの関数のほとんどには、意味的に同等の DB2 関数があり、デフォルトで関連付けられた関数マッピングがあります。ただし、一部のリモート・ソース関数には、フェデレーテッド・サーバー上に同等の関数がない場合があります。したがって、リモート・データ・ソースだけがこれらの関数を実行できます。これらの関数を使用する照会を書くには、フェデレーテッド・サーバー上に関数テンプレートを作成する必要があります。

関数テンプレートは、リモート関数のローカル記述としての役割を果たします。関数テンプレートは、CREATE FUNCTION ステートメントで AS TEMPLATE 文節を使用して作成します。フェデレーテッド・サーバーでは、関数テンプレートと関連付けられた実行可能コードはありません。テンプレートの定義時に、それを使用して、関数テンプレートをそのリモートの対応するものにマップする関数マッピングを作成できます。次いで、フェデレーテッド・サーバーに発行され、データ・ソースで評価される関数のために発行された SQL ステートメント内の関数テンプレートを参照できます。

照会オプティマイザーは、述部をどこで評価できるかを判断する、コストに基づく決定を下すことができます。可能な場合、オプティマイザーは対応するリモート・サーバー側で関数テンプレートにより述部を評価するプランを生成します。場合によっては、オプティマイザーがデータ・ソース側で関数テンプレートを評価するプランを生成できないこともあります。これが生じた場合は、SQL0142N エラーと、以下のエラー・メッセージを受け取ります。

この SQL ステートメントはサポートされていません。

このエラーを避けるには、照会を書き直して、元の照会の意味を維持しながら、プッシュダウンを使用可能にすることができます。

関数テンプレートをプッシュダウンするには、それが DETERMINISTIC および NO EXTERNAL ACTION 文節を指定して定義されている必要があります。

第 26 章 ニックネームを参照する照会の並列処理

ニックネームを含む照会は、次の 3 種類の照会内並列処理に対応しています。

3 種類の照会内並列処理は次のとおりです。

- 単一パーティション/マルチプロセッサー構成でのパーティション内照会並列処理
- 複数パーティション構成でのパーティション間照会並列処理
- パーティション内並列処理およびパーティション間並列処理で構成される混合照会並列処理 (パーティションごとに 1 つの SMP コンピューターで実行される)

ニックネームを参照する照会によるパーティション内並列処理

パーティション内並列処理とは、1 つの照会を複数の部分に分割して並行させ、それらを 1 つのデータベース・パーティション上の複数のプロセスで並列に実行する処理のことです。

フェデレーテッド照会では、ローカル・データを含む照会の部分を並列に実行することは可能ですが、ニックネームを含む部分は 1 つのエージェント・プロセスを使って逐次に実行されます。

照会のローカルの部分で複数のプロセッサーが機能できると、ローカル表およびニックネームを参照する照会のパフォーマンスは向上します。

DFT_DEGREE データベース構成パラメーターと CURRENT DEGREE 特殊レジスターが、パーティション内並列処理の度合いを制御します。

ニックネームを参照する照会によるパーティション内並列処理の使用可能化

マルチプロセッサー環境でローカルの表とニックネームを参照する照会には、パーティション内並列処理を使用可能にすることができます。その結果、フェデレーテッド・サーバーはローカル表を並列して処理できます。

このタスクについて

制約事項

フェデレーテッド・システムが並列して処理できるのは、ローカル表を参照する照会の部分のみです。照会のリモート部分のすべての操作は、コーディネーター・パーティションが照会を順次処理します。

手順

パーティション内並列処理を使用可能にするには、次のようにします。

手順

1. INTRA_PARALLEL データベース・マネージャー構成パラメーターを YES に設定する。

2. MAX_QUERYDEGREE データベース構成パラメーターを 1 より大きい値に設定する。
3. DFT_DEGREE データベース構成パラメーターを 1 より大きい値に設定するか、または特殊レジスター CURRENT DEGREE を設定する。 DFT_DEGREE パラメーターを ANY に設定すると、パーティション内並列処理のデフォルト・レベルはコンピューター上のプロセッサーの数と同じになります。

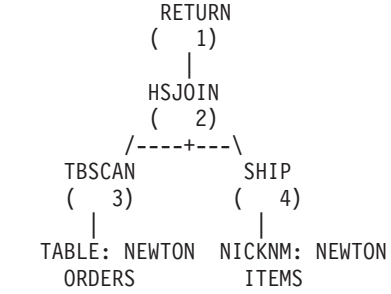
ニックネームを参照する照会によるパーティション内並列処理のアクセス・プランの例

DB2 Explain 機能を使用して、オプティマイザーが照会処理中に使用するアクセス・プランを表示することができます。以下の例は、パーティション内並列処理環境で、オプティマイザーがニックネーム・データにアクセスする方法を示しています。

例 1: 並列処理サポートなし

この例では、フェデレーテッド・サーバーはローカル表 ORDERS とニックネーム ITEMS の結合を逐次処理します。パーティション内並列処理は使用しません。

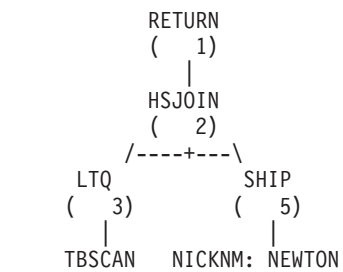
```
SELECT *
FROM ORDERS A, ITEMS B
WHERE A.ID1 = B.ID1 AND B.ITEM = 3
```



例 2: 並列処理サポートあり

この結合の例では、ニックネームと逐次結合する前にローカル表を並列に読み取ることにより、照会を高速に実行することが可能になっています。 LTQ 演算子は、並列処理がプランに導入されている場所を示します。

```
SELECT *
FROM ORDERS A, ITEMS B
WHERE A.ID1 = B.ID1 AND B.ITEM = 3
```



```
     (   4)        ITEMS
        |
   TABLE: NEWTON
    ORDERS
```

例 3: 集約によるパーティション内並列処理

この例では、データベースはパーティション内で並列の関係にあるローカル表データを集約することで、集約の実行を向上させています。ローカル表とニックネームの結合は、逐次に実行されます。

```
SELECT *
FROM ITEMS A
WHERE ID =
   (SELECT MAX(ID)
   FROM ORDERS
   WHERE NUMBER = 10)
```

```
            RETURN
            (   1)
               |
            NLJOIN
            (   2)
           /----+---\
       GRPBY         SHIP
      (   3)        (   7)
         |             |
       LTQ      NICKNM: NEWTON
      (   4)        ITEMS
         |
         1
      GRPBY
      (   5)
         |
       TBSCAN
      (   6)
         |
  TABLE: NEWTON
     ORDERS
```

ニックネームを参照する照会によるパーティション間並列処理

パーティション間並列処理とは、1 つの照会を複数の部分に分割し、それらをパーティション・データベースの異なるパーティションで並列に実行する処理のことです。

ローカル・データとリモート・データを参照する照会では、フェデレーテッド・サーバーがリモート・データを各ローカル・パーティションに分配できます。 262 ページの図 10 および 262 ページの図 11 は、ローカル・データ・ソースとリモート・データ・ソースが関係するパーティション間並列処理の概念を示したものです。

262 ページの図 10 は、パーティション間並列処理を使わずにこのタイプの照会を処理する方法を示しています。リモート・ニックネーム・データとローカル・パーティション化データは、コーディネーター・パーティションで逐次処理されます。ほとんどの処理は 1 つのパーティションで実行されるため、この手法ではデータベース・パーティションの並列処理能力は使用されません。データ・ボリュームが非常に大きい場合にこの手法を使うと、照会の実行時間が長くなる可能性があります。

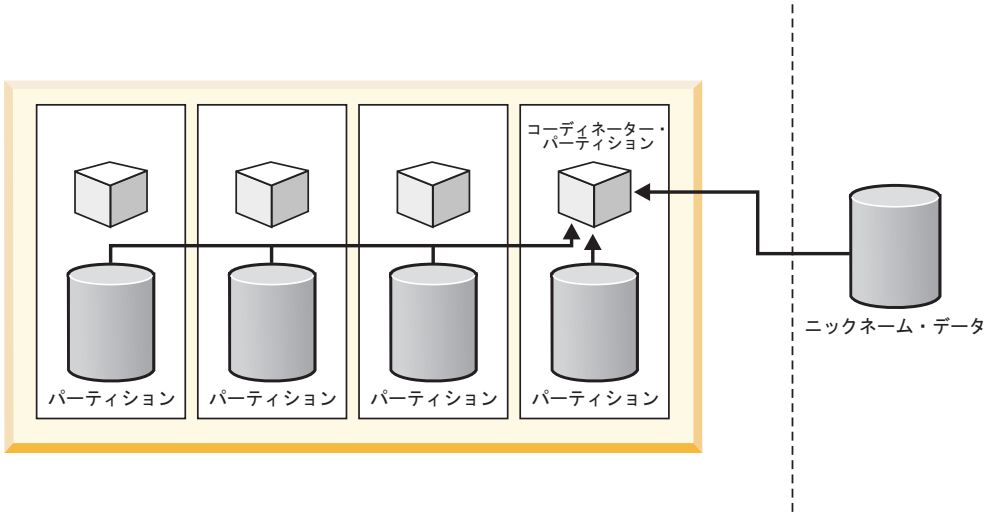


図 10. ローカルおよびリモート・データ・ソースのパーティション間並列処理を使用しない照会

図 11 は、オプティマイザーがパーティションにニックネーム・データを分配するときに、どのように処理が行われるかを示したものです。コーディネーター・パーティションはニックネーム・データを取り出し、並列処理を行うためにデータをデータベース・パーティションに分配します。並列処理が完了すると、アプリケーションに戻される前に、最終処理のために結果がコーディネーター・パーティションに送り戻されます。

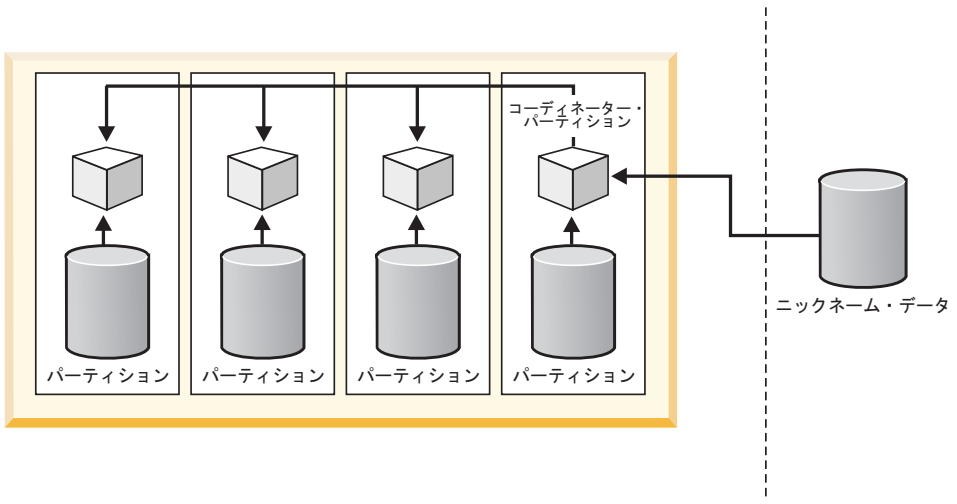


図 11. ローカルおよびリモート・データ・ソースのパーティション間並列処理を使用した照会

263 ページの図 12 および 264 ページの図 13 は、リモート・データ・ソースのみが関係するパーティション間並列処理の概念を示したものです。

図 12 は、コーディネーター・パーティションで実行されるリモート・ニックネーム・データの逐次処理を示しています。コーディネーター・パーティション (フェデレーテッド・サーバーとしての役割も果たす) は、ニックネーム・データを検索し、それを逐次処理します。

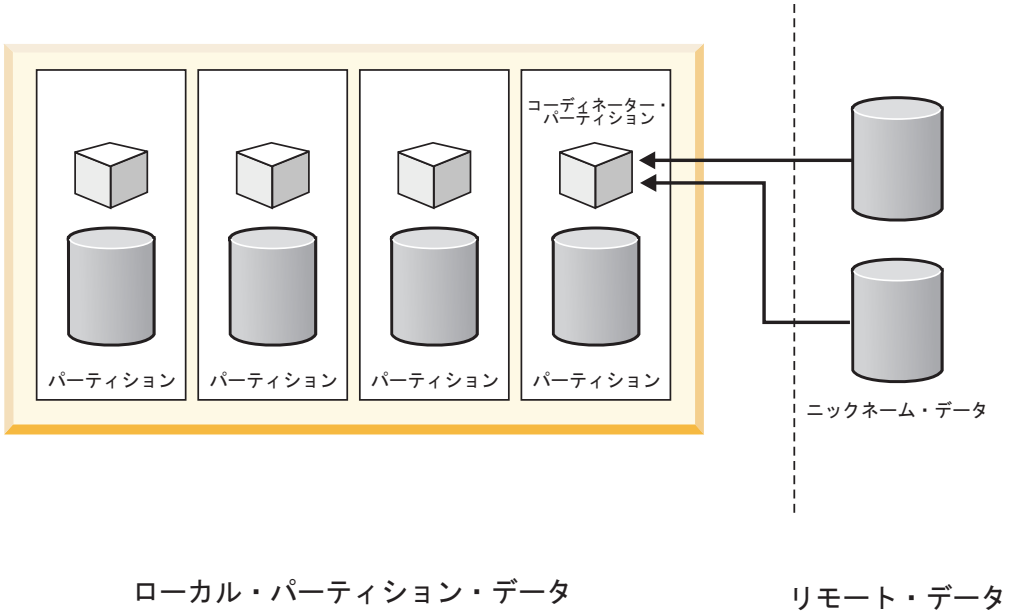


図 12. リモート・データ・ソースのみを参照するパーティション間並列処理を使用しない照会。

264 ページの図 13 は、コーディネーター・パーティションが計算パーティション・グループにデータを分配することを示しています。計算パーティション・グループを使用すると、オプティマイザーは、並列処理のためにパーティション・データベース・サーバーのパーティション全体にニックネーム・データを分配するアクセス・プランを生成することができます。

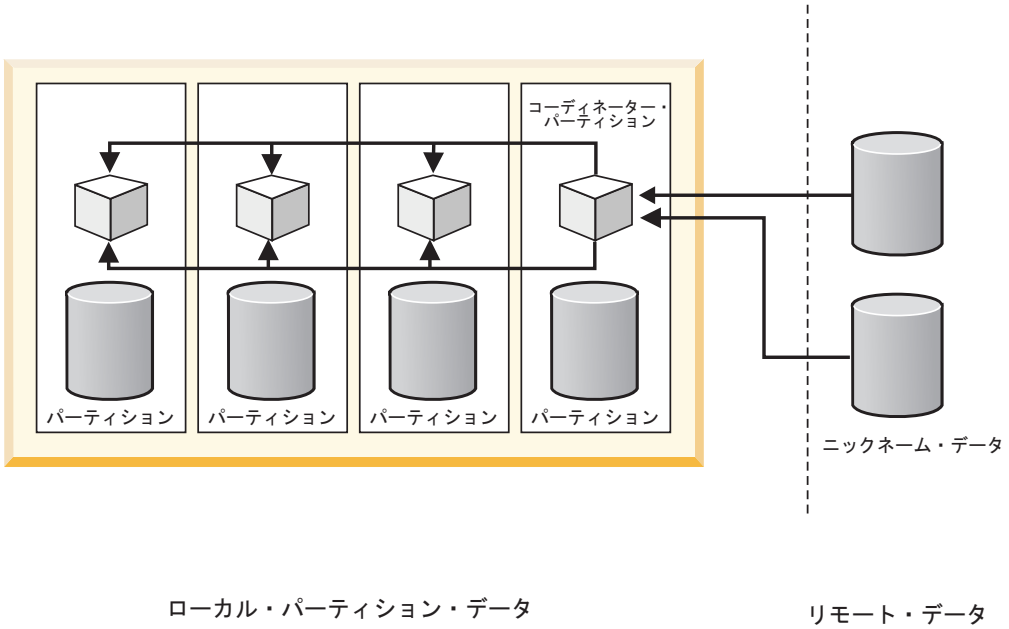


図 13. リモート・データ・ソースのみを参照するパーティション間並列処理を使用する照会。

オプティマイザーが選択するプランに関係なく、ニックネーム・データへのアクセスは常に、コーディネーター・パーティションから逐次行われます。

ニックネームを参照する照会によるパーティション間並列処理の使用可能化

ニックネームに関係する照会が複数のパーティション上で潜在的に並列で実行できるように、パーティション化されたフェデレーテッド・サーバーを構成できます。並列実行により、パーティション化された環境内のフェデレーテッド照会の経過時間が大幅に削減される場合があります。

このタスクについて

制約事項

照会の中の、Fenced ラッパーを使用するニックネームを参照する部分だけが、並列に実行できます。照会の中の、トラステッド・ラッパーを使用するニックネームを参照する部分は、どれも並列に実行できません。

このタスクについて

パーティション化されたデータベース環境では、以下の条件を満たす場合にフェデレーテッド照会を並列に実行できます。

- フェデレーテッド照会に、Fenced ラッパーとローカルにパーティション化された表を使用して定義されたニックネームの組み合わせが関係する。
- フェデレーテッド照会に、Fenced ラッパーを使用して定義されたニックネームが関係し、計算パーティション・グループが定義されている。

パーティション化された環境では、フェデレーテッド照会に使用できるパーティション間並列処理を行うためにデータベース・パラメーターやデータベース構成パラメーターを設定する必要はありません。

手順

パーティション間並列処理を使用可能にするには、次のようにします。

手順

1. CREATE WRAPPER または ALTER WRAPPER ステートメントの DB2_FENCED オプションを Y に設定して発行します。
2. **オプション:** ニックネームだけに関係した照会の部分に対して並列処理を使用可能にしたい場合は、計算パーティション・グループをセットアップします。 照会に、ニックネームとローカル・パーティション表の組み合わせが関係する場合、計算パーティション・グループをセットアップする必要はありません。

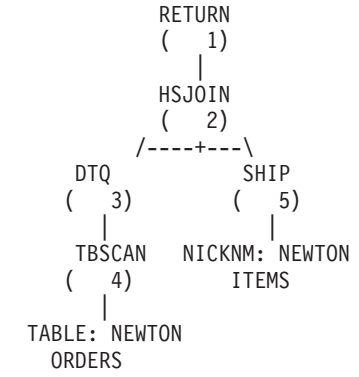
ニックネームを参照する照会によるパーティション間並列処理のアクセス・プランの例

DB2 Explain 機能を使用して、オプティマイザーが照会処理中に使用するアクセス・プランを表示することができます。次の例は、パーティション間並列処理環境で、オプティマイザーがニックネーム・データにアクセスする方法を示しています。

例 1: トラステッド・モード

この例では、ニックネームはトラステッド・ラッパーを使用します。データベースは、コーディネーター・パーティションでローカル表とニックネームの結合を逐次に実行します。データベースは、2 つのパーティションに配布されているローカル・データをコーディネーター・パーティションに移動します。次いで、フェデレーテッド・サーバーはそのローカル・データをニックネーム・データと結合します。データベースは、トラステッド・ラッパーを使用して定義されたニックネームをコーディネーター・パーティションで結合します。データベースは、並列結合を作成する際に、複数のパーティションにまたがってデータを配布することはできません。

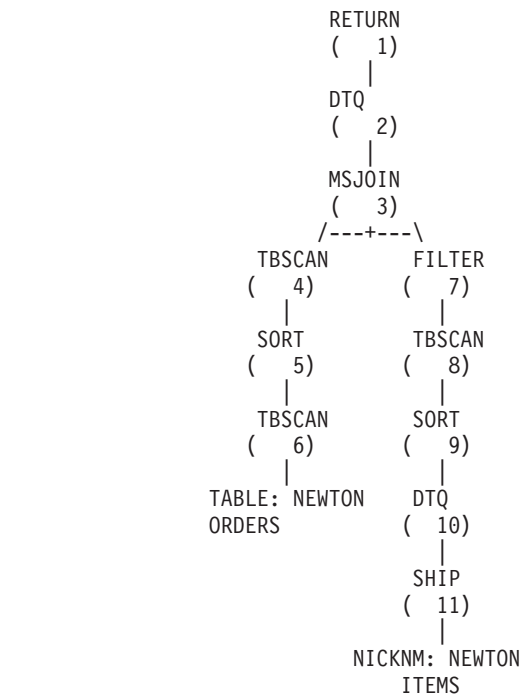
```
SELECT *
FROM ORDERS A, ITEMS B
WHERE A.ID1 = B.ID1 AND B.ITEM = 3
```



例 2: Fenced モード

この例では、ニックネームは Fenced ラッパーを使用します。フェデレーテッド・サーバーはニックネーム・データを他のパーティションに配布し、ローカル・データとの結合を並列して実行します。 SHIP の上の DTQ (分散表キュー) 演算子は、ニックネーム・データがハッシュ・パーティション化を使用してローカル・パーティションに分散され、同じ場所での並列結合を実現することを示します。同じ場所での並列結合では、結合の対応するニックネームとローカル・データは必ず同じパーティション上に配置されるという仕方で、ニックネーム・データがローカル・パーティションに分散されます。

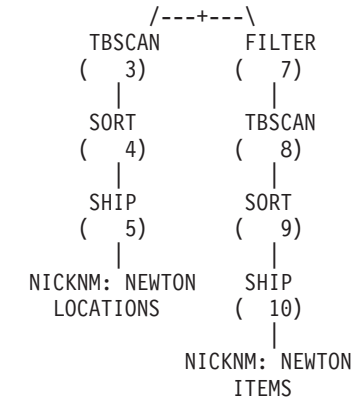
```
SELECT *
FROM ORDERS A, ITEMS B
WHERE A.ID1 = B.ID1 AND B.ITEM = 3
```



例 3: 計算パーティション・グループを使用しない Fenced モード

この例では、2 つのニックネームが Fenced ラッパーを使用し、計算パーティション・グループは定義されていません。フェデレーテッド・サーバーはコーディネーター・パーティションで結合を実行します。フェデレーテッド・サーバーがデータを他のパーティションに配布して処理を実行することはありません。どの SHIP 演算子の上にも TQ 演算子がないということは、ニックネーム・データがパーティション間で分散されないことを示します。

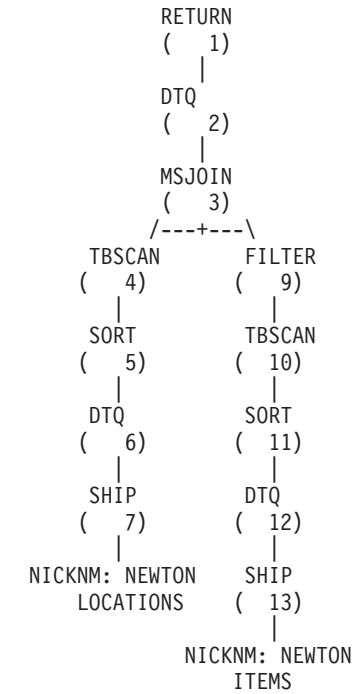
```
SELECT *
FROM ITEMS A, LOCATIONS B
WHERE A.ID1 = B.ID1
```



例 4: 計算パーティション・グループを使用する Fenced モード

この例では、ニックネームが Fenced ラッパーを使用し、計算パーティション・グループが定義されています。この場合、オプティマイザーは、データをコーディネーター・パーティションから計算パーティション・グループ内の他のパーティションに配布するプランを選択します。どちらのニックネームの上の DTQ 演算子も、対応する結合キーが計算パーティション・グループの同じパーティション上に置かれるように、着信リモート・データをハッシュ・パーティション化します。結合は各パーティション上で並列に行われ、それから結果がコーディネーター・パーティションで収集されます。

```
SELECT *
FROM ITEMS A, LOCATIONS B
WHERE A.ID = B.ID
```



計算パーティション・グループ

計算パーティション・グループには、オプティマイザーがニックネーム・データを動的に再配分するために使用できるデータベース・パーティションのセットが定義されます。計算パーティション・グループは、ニックネームのみを含む照会の部分のためのものです。

コーディネーター・パーティションはニックネーム・データを逐次取り出してから、計算パーティション・グループのパーティション全体にデータを再配分します。その時点で、並列処理が行われます。オプティマイザーが計算パーティション・グループを使用するとパフォーマンスが向上することがよくあります。特に、大量のニックネーム・データが関係していたり、照会が複雑な場合などです。

計算パーティション・グループは、データベース・パーティション・グループとしてシステム・カタログ SYSCAT.DBPARTITIONGROUPS に指定されますが、それには IBMCATNODEGROUP は含まれません。

計算パーティション・グループの指定には、DB2_COMPPARTITIONGROUP レジストリー変数を使用します。

計算パーティション・グループの定義

計算パーティション・グループを定義すると、オプティマイザーは、計算パーティション・グループのパーティションにニックネーム・データを配布するプランを使用できるようになります。計算パーティション・グループを定義するのは、ニックネームのみを参照する照会または照会の一部にパーティション間照会並列処理を使用できるようにするためです。

始める前に

始める前に

インスタンス内のすべてのデータベース上で計算パーティション・グループを表すために使用するパーティション・グループは、すべて同じ名前でなければなりません。これらのパーティション・グループの定義をデータベースごとに異なるものにすることはできますが、それらの名前は必ず同じにします。例えば、DB1、DB2、および DB3 と呼ばれる 3 つのデータベースは、異なるノードを含む計算パーティション・グループを定義します。

- DB1: CPG は、ノード 1、2、3、および 4 を含む。
- DB2: CPG は、ノード 49、50、および 53 を含む。
- DB3: CPG は、ノード 78 および 96 を含む。

db2set 変数は、CPG という名前に設定できます。 CPG という名前はすべてのデータベースに共通ですが、CPG の内容はデータベースごとに異なります。

このタスクについて

制約事項

オプティマイザーは計算パーティション・グループを、照会中のニックネームを参照する部分に対してのみ使用しますが、ローカル・データは参照しません。

手順

計算パーティション・グループを定義するには、DB2 コマンド行で次のコマンドを発行します。

```
db2set DB2_COMPPARTITIONGROUP=partitiongroup_name
```

partitiongroup_name は、計算パーティション・グループとして定義するパーティション・グループの名前です。パーティション・グループは定義済みでなければなりません。

次の例は、DB2_COMPPARTITIONGROUP レジストリー変数を使用して計算パーティション・グループ FINANCE3 を定義する方法を示しています。

```
db2set DB2_COMPPARTITIONGROUP=FINANCE3
```

ニックネームを参照する照会によるパーティション間並列処理 - 期待されるパフォーマンス

ローカルにパーティション化された表とニックネームの組み合わせを参照する照会の場合、オプティマイザーは適切なパーティションにニックネーム・データを再配布する実行プランを選択できます。

結合内でニックネーム・データの量がローカル・パーティション化データの量よりも少ない場合は、再配布プランにより照会の実行を高速にすることができます。結合内のニックネーム・データの量がローカル・パーティション化データの量よりも大幅に多い場合は、ニックネーム・データの再配布を含む並列プランを使用することはできません。オプティマイザーが並列プランを選択しない場合は、フェデレーテッド・サーバーがニックネームとローカル表の逐次結合をコーディネーター・パーティションで実行します。

2 つのニックネームの結合の場合は、関係するデータの量が多いときには、計算パーティション・グループのすべてのパーティションにデータを配布する実行プランが便利です。サイズの大きい結合を並列で処理することの利点は、複数のパーティションにデータを再配布する追加コストが埋め合わされることです。ニックネーム・データの量が比較的小さい場合は、複数のパーティションにデータを再配布したとすれば余分にかかるコストを抑えることができるので、結合によるコストは高くありません。一般に、関係するニックネームが大きい場合に、オプティマイザーは計算パーティション・グループのプランを選択し、それ以外の場合は、フェデレーテッド・サーバーはコーディネーター・パーティションでニックネームを逐次結合します。

ニックネームを参照する照会による混合並列処理

パーティション化された環境でローカルの表とニックネームを含む照会には、オプティマイザーはパーティション内並列処理とパーティション間並列処理の両方を使用できます。パーティション間並列処理は、パーティション環境におけるオプティマイザーのオプションです。パーティション内並列処理は、データベース構成またはデータベース・マネージャー構成内で使用可能にされた場合のオプションです。

パーティション間並列処理では、フェデレーテッド・サーバーは、複数のパーティション間でリモート・データを分配し、各パーティション内で並列にデータを処理することができます。

パーティション内並列処理では、1 つのパーティション内の複数のサブエージェント・プロセスを使用してローカル・データを並列処理します。

ニックネームを参照する照会による混合並列処理の使用可能化

パーティション内並列処理およびパーティション間並列処理を使用することで、ローカル・データとリモート・データを参照する照会のパフォーマンスを向上させることができます。

このタスクについて

手順

パーティション化されたフェデレーテッド・サーバー上でパーティション間並列処理を使用可能にするには、次のようにします。

1. CREATE WRAPPER または ALTER WRAPPER ステートメントの DB2_FENCED オプションを Y に設定して発行します。
2. **オプション:** ニックネームのみの結合に対して並列処理を使用可能にする計算パーティション・グループをセットアップします。

フェデレーテッド・サーバー上でパーティション内並列処理を使用可能にするには、次のようにします。

1. MAX_QUERYDEGREE データベース構成パラメーターを 1 より大きい値に設定する。
2. DFT_DEGREE データベース構成パラメーターを 1 より大きい値に設定するか、または特殊レジスター CURRENT DEGREE を設定する必要があります。DFT_DEGREE パラメーターを ANY に設定すると、パーティション内並列処理のデフォルト・レベルはコンピューター上の SMP プロセッサーの数と同じになります。

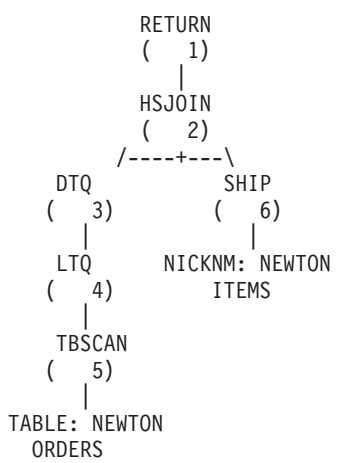
ニックネームを参照する照会による混合並列処理のアクセス・プランの例

DB2 Explain 機能を使用して、オプティマイザーが照会処理中に使用するアクセス・プランを表示することができます。次の例は、パーティション内並列処理環境とパーティション間並列処理環境の両方を使用する環境で、オプティマイザーがニックネーム・データにアクセスする方法を示しています。

例 1: トラステッド・モード

次の例は、トラステッド・モードでのローカル表とニックネームの結合を示しています。フェデレーテッド・サーバーはローカル・データを各パーティションで並列に処理してから、コーディネーター・パーティションでローカル・データをニックネームと結合します。フェデレーテッド・サーバーはニックネーム・データの処理を複数のパーティション、または 1 つのパーティション上の複数のプロセッサーにまたがって並列に行うことはありません。

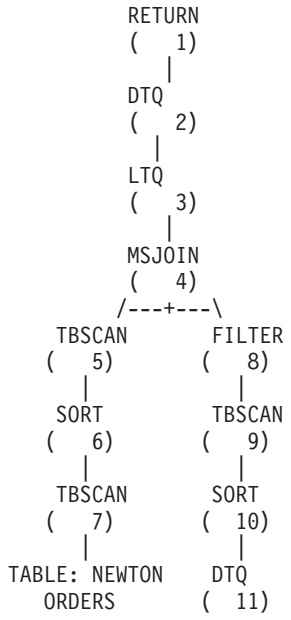
```
SELECT *
FROM ORDERS A, ITEMS B
WHERE A.ID1 = B.ID1 AND B.ITEM = 3
```

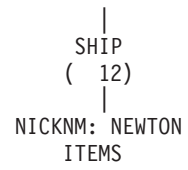


例 2: Fenced モード

次の例は、fenced モードでのローカルにパーティション化された表とニックネームの結合を示しています。フェデレーテッド・サーバーは、ニックネーム・データをコーディネーター・パーティションに逐次に取り出し、このデータをデータベース内の他のパーティションに分散させます。次いでデータは、該当するすべてのデータベース・パーティション間でローカル・データと並列に結合されます。各パーティション内で、複数のサブエージェントはローカル表を読み取り、ニックネーム・データに結合します。この操作はパーティション内並列処理であり、LTQ 演算子によりプラン内で識別されます。結果は最終処理のためにコーディネーター・パーティションに戻され、アプリケーションに戻されます。

```
SELECT *
FROM ORDERS A, ITEMS B
WHERE A.ID1 = B.ID1 AND B.ITEM = 3
```




SHIP
(12)
NICKNM: NEWTON
ITEMS

第 27 章 フェデレーテッド照会の非同期処理

非同期 は、照会パフォーマンスを向上させる 1 つの方法です。アクセス・プランの複数の部分を並行して実行することにより、特定の照会の経過時間を削減します。

フェデレーテッド・システムでは、データは複数のデータ・ソースで複数のシステムに分配され、各システムがその独自のリソースを持ちます。非同期はそれらのリソースを使用する操作をオーバーラップして、複数のシステムが同時にアクティブの状態を保てるようにします。操作をオーバーラップすることで、アクセス・プランの複数の部分を逐次ではなく、並行して実行することができます。

リモート・データ・ソースでの時間のかかる操作を含む複雑な照会の場合、非同期は有効です。非同期により、以下のタイプの操作を同時に行うことが可能になります。

- リモート・データ・ソースでの 2 つ以上の操作
- フェデレーテッド・サーバーと少なくとも 1 つのリモート・データ・ソースでの操作

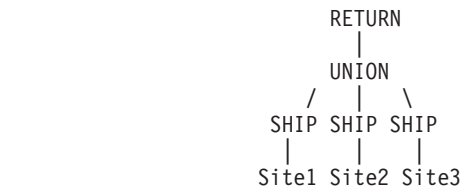
照会操作はリソースを消費するので、非同期はリソースがアイドル状態になっているシステムや、データ・ソースの 1 つ、またはフェデレーテッド・システムだけがいずれかの時点で作業を行っている場合に有効です。

フェデレーテッド照会の非同期処理 - 例

UNION およびマージ結合がある照会の例により、非同期処理がある場合とない場合の照会操作の違いを示します。

例: UNION 操作がある照会

単純照会は、3 つの異なるデータ・ソースからのデータに対して UNION 操作を実行します。各データ・ソース上でデータを生成するために必要な計算は時間がかかります。アクセス・プランは、以下のようになります。



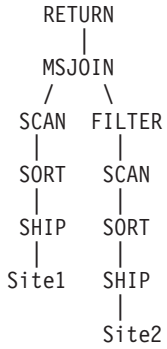
非同期なしの場合、UNION 操作はデータを照会分岐から一度に 1 分岐、左から右に読み取ります。 Site1 サーバーからのデータの読み取り時には、Site2 サーバーと Site3 サーバーはアイドル状態になります。例えば、UNION の各分岐は、1 サイトから結果行を戻すために約 2 時間かかります。 3 つの分岐の合計実行時間は、概算で各分岐の処理にかかる時間の合計になり、この例では約 6 時間になります。

非同期ありの場合、UNION の各分岐は同時に処理を開始し、3 つのリモート・サーバーは同時にアクティブになります。照会の実行時間は、UNION の最も遅い分岐の

実行時間とほぼ等しくなります。この例では、非同期なしの 6 時間と比べて、実行時間は約 2 時間まで削減されます (約 66% 速い)。

例: マージ結合操作がある照会

2 つの異なるデータ・ソースからのデータを結合する照会は、マージ結合操作 (MSJOIN) を使用します。オプティマイザー・アクセス・プランは、以下のようになります。



非同期なしの場合、マージ結合演算子はまず外側 (左) の分岐を処理し、左の分岐が行を戻し始めるまで、内側 (右) の分岐を処理しません。この例の場合、各分岐は複合照会を実行するため、実行に時間がかかります。マージ結合を実行するための概算の合計時間は、各分岐の実行にかかる時間の合計です。

非同期ありの場合、マージ結合の両方の分岐は同時に開始されるため、照会の総実行時間は削減されます。

非同期最適化

照会オプティマイザーは、照会実行プランにおけるリモート操作の非同期処理に関する決定を行います。*非同期最適化* は、オプティマイザーが既存の照会実行プランを分析し、リモート操作を同時実行する機会を探すプロセスです。

非同期なしのアクセス・プラン

実行プランでは、SHIP または RPD 演算子はリモート・データ・ソースで実行されるプランの部分を定義します。

非同期を使用しないと、実行プラン内の SHIP または RPD 演算子の上に位置する他の演算子がそのデータを必要とする場合にのみ、SHIP または RPD 演算子はアクティブになり、リモート処理を開始します。

非同期に合わせて最適化されたアクセス・プラン

オプティマイザーは、SHIP または RPD 演算子が定義するリモート操作を非同期に実行させることができます。

非同期操作では、実行プランの SHIP または RPD 演算子のすぐ上に表キュー (TQ) 演算子が挿入されます。 TQ 演算子はサブプラン と呼ばれるプランの部分を定義します。別のプロセスまたはスレッドがその独自のメモリーを使用してサブプランを実行します。サブプランは照会の開始直後に開始されます。

TQ 演算子は、プラン内の SHIP または RPD 演算子 (データの製作者) とその上の演算子 (データの消費者) の間のパイプと見なすことができます。このパイプはメインのプランからその下のサブプラン内の SHIP の実行を切り離し、2 つのプランのセクション間のデータを非同期交換できるようにします。

プランの SHIP または RPD 演算子のすぐ上に現れる TQ 演算子は、SHIP または RPD 演算子が定義するリモート操作を照会の先頭から開始し、結果を非同期的にフェデレーテッド・サーバーに送信できるようにします。特定のリモート操作で非同期が有効な場合、オプティマイザーはプラン内の対応する SHIP または RPD 演算子のすぐ上に TQ 演算子を置きます。

実行プラン内の TQ 演算子の目的はそれぞれ異なります。通常、TQ 演算子はパーティション・データベースまたはパーティション内並列処理が使用可能なデータベースでの並列操作を表します。サブプランの非同期実行を可能にする別のタイプの TQ 演算子は、非同期 TQ (ATQ) と呼ばれます。

オプティマイザーは次のような場合に特定の SHIP または RPD 演算子を非同期にします。

- それによって照会パフォーマンスが向上する
- ATQ の数がサーバーごとおよび照会ごとの限界より少ない
- 別の最適化手法を使用しているためにまだ演算子が非同期になっていない
- 非同期の制限に関する違反がない
- 照会のセマンティクスが変更されない

アクセス・プラン - 例

アクセス・プランの例では、非同期最適化が行われるプラン実行と行われないプラン実行の違いが例示されています。

最初の 2 つの例は、非同期が使用可能な場合の 273 ページの『フェデレーテッド照会の非同期処理 - 例』 の UNION およびマージ結合プランの外観を示しています。

簡単にするために、例では SHIP 演算子を持つプランのみを示しています。非同期最適化は、RPD 演算子のプランを SHIP 演算子のプランと同じ方法でトランスフォームします。 SHIP 演算子と RPD 演算子は、特に指定のない限り、交換可能です。

例 1a: 非同期なしのプラン

例 1b: 非同期ありのプラン

```
    /   |   \
 SHIP  ATQ  ATQ
         |    |
       SHIP  SHIP
```

例 2a: 非同期なしのプラン

```
        RETURN
          |
        MSJOIN
        /    \
     SCAN   FILTER
       |       |
     SORT    SCAN
       |       |
     SHIP    SORT
             |
            SHIP
```

例 2b: 非同期ありのプラン

```
        RETURN
          |
        MSJOIN
        /    \
     SCAN   FILTER
       |       |
     SORT    SCAN
       |       |
     SHIP    SORT
             |
            ATQ
             |
            SHIP
```

例 3a: 非同期なしのプラン

```
        RETURN
          |
         HSJN
        /    \
     SHIP   SHIP
```

例 3b: 非同期ありのプラン

```
        RETURN
          |
         HSJN
        /    \
      ATQ   SHIP
       |
     SHIP
```

例 4a: 非同期なしのプラン

```
        RETURN
          |
         NLJN
        /    \
     SHIP   SCAN
```

```
        |
       TEMP
        |
       SHIP
```

例 4b: 非同期ありのプラン

```
  RETURN
    |
   NLJN
  /    \
SHIP   SCAN
        |
       TEMP
        |
       ATQ
        |
       SHIP
```

例 5a: 非同期なしのプラン

```
  RETURN
    |
   UNION
  /    \
SHIP   SHIP
        |
       NLJN
      /    \
   SHIP    NICK2
     |
   SHIP
     |
   NICK1
```

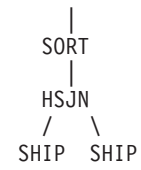
このプランでは、SHIP-SHIP の対の代わりに RPD を使用することはできません。

例 5b: 非同期ありのプラン

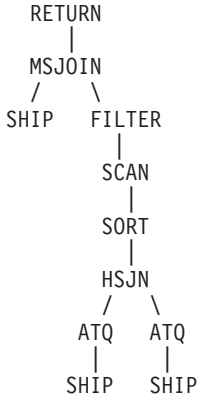
```
  RETURN
    |
   UNION
  /    \
SHIP   SHIP
        |
       NLJN
      /    \
   SHIP    NICK2
     |
    ATQ
     |
   SHIP
     |
   NICK1
```

例 6a: 非同期なしのプラン

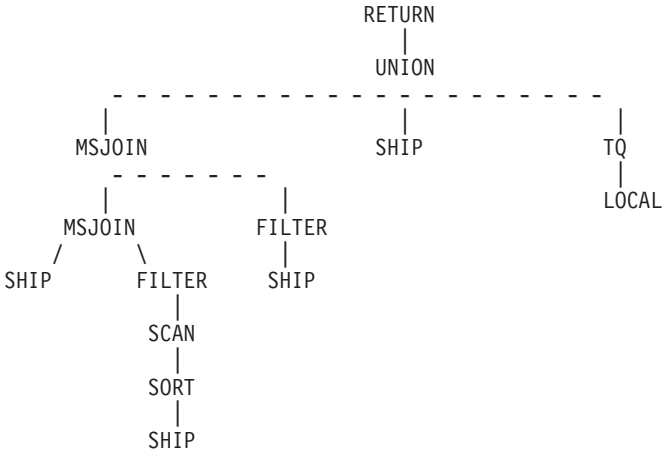
```
  RETURN
    |
  MSJOIN
  /    \
SHIP   FILTER
         |
        SCAN
```



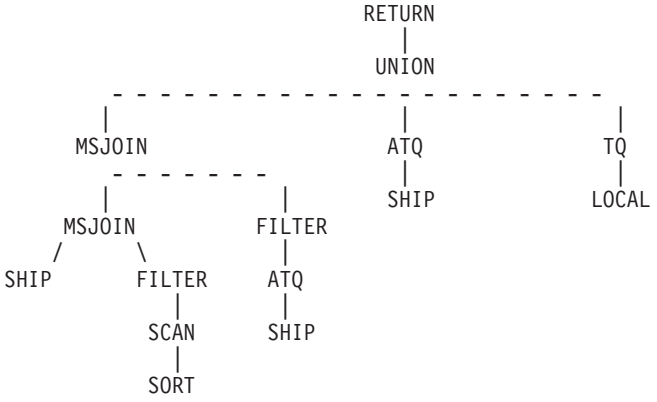
例 6b: 非同期ありのプラン



例 7a: 非同期なしのプラン



例 7b: 非同期ありのプラン



リソース消費量の制御

リモート照会の非同期実行を使用可能にすることに加えて、ATQ 演算子はフェデレーテッド・サーバーとリモート・データ・ソースに影響を与えます。

各 ATQ 演算子は新規のプロセスを作成し、(バッファリングのために) いくらかのメモリーを消費するため、実行プランに多数の ATQ を挿入すると、フェデレーテッド・サーバー上で大量のシステム・リソースが使用されることになる可能性があります。さらに、照会で複数の SHIP または RPD 演算子を特定のリモート・データ・ソースで実行する場合、複数の演算子を非同期にすると複数のカーソルがそのデータ・ソース上で同時に開かれ、許容できないほどの負荷が発生する可能性があります。

フェデレーテッド・システム上、またはデータ・ソース上のリソース消費量を制御するには、構成パラメーターを設定することができます。パラメーターは照会内で許容される ATQ の総数、および照会内の各サーバーで許容される ATQ の数に対する限界を定義します。

非同期最適化の使用可能化

非同期最適化を使用可能にするには、特定の照会に対して非同期 TQ 演算子の数を指定し、データ・ソースにサーバー・オプションを設定します。

このタスクについて

制約事項

非同期最適化には以下が必要になります。

- 複数の論理データベース・パーティションを含むデータベース・パーティション機能 (DPF) を持つフェデレーテッド・システム。
- fenced ラッパーを介したデータ・ソースへのアクセス。

この最適化は以下のオブジェクトをサポートしません。

- トラステッド・ラッパーを介してアクセスされるデータ・ソース上のニックネーム。
- 挿入、更新、または削除操作を行う照会。

手順

非同期処理を使用可能にするには、次のようにします。

手順

1. 以下のパラメーターの 1 つ以上を設定します。
   - FEDERATED_ASYNC データベース・マネージャー構成パラメーターを 0 以上 32 767 以下の値、または ANY に設定します。 MAXAGENTS は、1 つの照会に許容される ATQ の最大数を指定するデータベース構成パラメーター

です。値 ANY を指定すると、オプティマイザーが特定のアクセス・プランの ATQ の数を決定することができます。デフォルトは 0 です。

- オプションで、パッケージ内の静的ステートメントに FEDERATED_ASYNCHRONY バインドおよびプリコンパイル・オプションを設定し、照会の構成パラメーター設定をオーバーライドします。デフォルトは 0 です。
- オプションで、CURRENT FEDERATED ASYNCHRONY 特殊レジスターを設定し、データベース・マネージャー構成パラメーターの設定と照会用のバインド・オプションを動的にオーバーライドします。

これらのパラメーターにより次の階層が形成されます。

a. 特殊レジスター

b. バインド・オプション

c. データベース・マネージャー構成パラメーター

特殊レジスター値 (指定される場合) は、バインド・オプションに優先します。そしてバインド・オプションは、データベース・マネージャー構成パラメーターに優先します。

2. DB2_MAX_ASYNC_REQUESTS_PER_QUERY サーバー・オプションを数値に設定します。このサーバー・オプションは、照会に対してサーバーが許容する非同期要求の最大数を指定します。

   サーバー・オプションの範囲は -1 から 64000 です。

   - デフォルトは 1 です。そのため、サーバーに属する 1 つの SHIP 演算子または 1 つの RPD 演算子は照会内で非同期とみなされます。
   - ODBC データ・ソースの場合のみ、デフォルトは 0 です。非同期は 1 つの接続に対して複数のカーソルを必要とします。データ・ソースが 1 つの接続に対する複数のカーソルをサポートしない場合、ODBC ラッパーのこの値は 0 にすべきです。

非同期最適化のチューニング考慮事項

非同期が有効になっている場合、パフォーマンスに影響を与えるいくつかの要因について考慮する必要があります。

システムに使用可能なプロセス、メモリー、および CPU リソースがある場合、非同期を有効にすると、フェデレーテッド照会のパフォーマンスを向上させることができます。非同期を有効にすると、フェデレーテッド・サーバーによるリモート・ソース・システムの使用が増える可能性があります。これはリモート・ソースが、フェデレーテッド照会の代わりに一度に複数の要求を潜在的に処理できるからです。システムにリソース制約がある場合、非同期であるとパフォーマンスが低下する場合があります。

構成パラメーターをシステムに最適な程度の非同期を実現するように変更することで、非同期用にシステムを調整することができます。

非同期最適化がプランに導入する各非同期 TQ は、追加のサブエージェントを必要とします。十分なサブエージェントがシステムで使用できる場合は、MAXAGENTS データベース・マネージャー構成パラメーターの調整を考慮してください。

非同期の最適化に関する制限

オプティマイザーが特定の照会に非同期最適化を適用すると、照会実行プランで使用できる ATQ の数にいくつかの制限が適用されます。

プラン内で ATQ と結び付けるのに適しており、非同期が有効となる SHIP または RPD の数は、FEDERATED_ASYNC パラメーターによって設定される最大か、または DB2_MAX_ASYNC_REQUESTS_PER_QUERY サーバー・オプションのサーバーごとの限界のいずれかよりも大きくなる可能性があります。この場合、オプティマイザーは ATQ と結び付く SHIP または RPD を、以下の項目が真となるような仕方で選択します。

- プラン内の ATQ の総数が、以下のパラメーターに設定される値以下となる (リスト順)。
  1. FEDERATED ASYNCHRONY 特殊レジスター (指定される場合)
  2. FEDERATED_ASYNCHRONY バインドまたはプリコンパイル・オプション (指定される場合)
  3. FEDERATED_ASYNC パラメーター
- 特定のサーバーの ATQ の総数が、そのサーバーの DB2_MAX_ASYNC_REQUESTS_PER_QUERY サーバー・オプション以下となる。
- 適格な SHIP または RPD が ATQ との結び付きから益を得て、照会パフォーマンスが向上する。

非同期最適化が照会に適用されるかどうかの判別

非同期最適化が特定の照会に適用されるかどうかを判別するために、照会のアクセス・プランおよび特定の演算子を確認する複数の方法のいずれかを使用することができます。

問い合わせ対象の照会について、以下の出力のいずれかを確認することができます。

- db2exfmt 出力
- Visual Explain 出力
- dynexpln 出力

どの出力も、アクセス・プラン内の非同期照会要求は ATQ と示します。 ATQ の記述内の「起点 (origin)」プロパティーは「非同期 (Asynchrony)」を示します。

以下のプランの断片は、ATQ 演算子の使用法と、その詳細プロパティーを示しています。

```
                1.6e+06
                HSJOIN
                (   2)
                4213.74
                  131
            /--------+--------\
         40000                  1000
        HSJOIN                 SHIP
        (   3)                 (  10)
        1122.26                427.733
          117                    14
```

```
          /------+-----\              |
       1000             1000         1000
      ATQ              ATQ         NICKNM: NEWTON
      (   4)           (   7)        S1_NN07
      511.795          532.669
        16               101
        |                |
       1000             1000
      SHIP             SHIP
      (   5)           (   8)
      478.773          499.887
        16               101
        |                |
       1000             1000
 NICKNM: NEWTON    NICKNM: NEWTON
      S2_NN02          S3_NN15

- show the origin of the ATQ as ASYNCHRONY:
 4) TQ    : (Table Queue)
  Cumulative Total Cost:   511.795
  Cumulative CPU Cost:   2.79486e+06
  Cumulative I/O Cost:   16
  Cumulative Re-Total Cost:  68.9489
  Cumulative Re-CPU Cost:  1.72372e+06
  Cumulative Re-I/O Cost:  0
  Cumulative First Row Cost:  30.8308
  Cumulative Comm Cost:  18.2176
  Cumulative First Comm Cost: 0
  Estimated Bufferpool Buffers:  16
  Remote communication cost: 538.297

  Arguments:
  ---------
  JN INPUT: (Join input leg)
   OUTER
  LISTENER: (Listener Table Queue type)
   FALSE
  TQMERGE : (Merging Table Queue flag)
   FALSE
  TQORIGIN: (Table Queue Origin type)
   ASYNCHRONY
  TQREAD  : (Table Queue Read type)
   READ AHEAD
  TQSEND  : (Table Queue Write type)
   DIRECTED
  UNIQUE  : (Uniqueness required flag)
   FALSE
```

照会のアクセス・プラン内に ATQ 演算子が見つからない場合、以下の条件を満たしていることを調べてください。

- システムは、DPF に使用できるフェデレーテッド・システムである。
- 照会は、fenced ラッパーを介してアクセスされるデータ・ソース上のニックネームにアクセスする。
- 非同期が、データベース・マネージャー構成パラメーター FEDERATED_ASYNC、特殊レジスター CURRENT FEDERATED ASYNCHRONY、またはバインド・オプション FEDERATED_ASYNCHRONY を使用して有効になっている。
- サーバー・オプション DB2_MAX_ASYNC_REQUESTS_PER_CONNECTION が、各サーバーでニックネームについて非ゼロ値に設定されている。

第 28 章 グローバルな最適化

SQL コンパイラーは 3 つのフェーズで機能します。これらのフェーズは、リモート・データ・ソースを参照する照会を評価するための最適なアクセス方針を作成する上で助けになります。 3 つのフェーズとは、プッシュダウン分析、グローバル最適化、およびリモート SQL 生成です。

フェデレーテッド・データベースにサブミットされる照会の場合、そのアクセス方針では、元の照会をいくつかの照会フラグメントに分解し、その後、その照会フラグメントの結果を結合する、ということが考えられます。

プッシュダウン分析フェーズの出力を参考にして、照会オプティマイザーは、それぞれの操作をどこで評価するかを決めます。ある操作を、フェデレーテッド・サーバー側でローカルに評価することも、データ・ソース側でリモートで評価することも考えられます。どちらにするかの決定は、オプティマイザーが使用する綿密なコスト・モデルの出力を基に行われます。このモデルは、次のことを判断します。

- その操作を評価するためのコスト
- フェデレーテッド・サーバーとデータ・ソース間でデータやメッセージを伝送するためのコスト

グローバル最適化の目標は、フェデレーテッド・システム全体にわたって (グローバルに)、すべてのデータ・ソースの照会操作を最適化するアクセス・プランを生成することです。グローバルに最適化されたアクセス・プランは、フェデレーテッド・システムでの実行に全体として最もコストのかからないプランです。リモート SQL 生成のフェーズでは、グローバルに最適化されたプランから個々のデータ・ソースによって実行される照会フラグメントへの逆の変換が行われます。

SQL コンパイラーには、サポートされるデータ・ソースとそのデータ・ソースのデータに関するメタデータの特性を含む知識ベースがあります。オプティマイザーは、リモート・データ・ソースが理解できない、または受け入れることができない SQL、照会フラグメント、またはプラン・ヒントを生成しません。

多くの要因がグローバルな最適化の出力に影響するので、照会のパフォーマンスにも影響を与えます。中でも重要な要因は、サーバーの特性とニックネームの特性です。

リレーショナル・ラッパーと非リレーショナル・ラッパーでは、アクセス・プランの作成の仕方は多少異なりますが、概念や最終的な効果は同じです。

グローバルな最適化に影響を与えるサーバー特性

サーバー定義を作成または変更するときに選択できるオプションの中には、照会パフォーマンスに影響を与えるものがあります。

ユーザーは、サーバー・オプション設定値を使用して、データ・ソース・サーバーの特性についての情報を照会オプティマイザーに提供します。サーバー・オプションの設定は、データ・ソース・サーバー定義の一部です。最初にサーバー定義を設

定する時に、CREATE SERVER ステートメントでサーバー・オプションを指定できます。既存のサーバー定義にサーバー・オプションを追加する場合は、 ALTER SERVER ステートメントを使用します。サーバー・オプションの設定値は、フェデレーテッド・データベースのグローバル・カタログに保管されます。

これらのオプションは、ロケーション・オプション (データ・ソース・コンピューターの名前など)、セキュリティー・オプション (認証情報など)、およびパフォーマンス・オプション (CPU 率など) に分類されます。

パフォーマンス・オプションは、データ・ソースで操作の評価を行えるかどうか、またデータ・ソースでの操作の評価により実行が速くなるかどうかをオプティマイザーが判断する上で役に立ちます。パフォーマンスに影響し、チューニングを要する可能性のあるサーバー・オプションは次のものです。

- CPU_RATIO
- IO_RATIO
- COMM_RATE
- COLLATING_SEQUENCE
- PLAN_HINTS
- VARCHAR_NO_TRAILING_BLANKS
- NO_EMPTY_STRING

照会のコスト計算でオーバーフローやアンダーフローが発生すると、予期しないエラーが戻されることがあるので、 CPU_RATIO、IO_RATIO、または COMM_RATE サーバー・オプションをチューニングする際は注意して行ってください。ほとんどの場合、これらのオプションのデフォルト値で十分です。通常、照会で参照されるオブジェクトに関する統計の正確さを確証することは、これらのサーバー・オプションの値を調整することよりも重要です。

CPU 速度の相対比率

この値は、フェデレーテッド・サーバーの CPU 速度と、リモート・データ・ソースが置かれているサーバー上の CPU 速度との間の比率を示します。

この値は、フェデレーテッド・サーバーの CPU 速度を、リモート・データ・ソース用のサーバーの CPU 速度で除算したものとして定義されます。例えば、フェデレーテッド・サーバーの CPU 速度がリモート・サーバーの CPU 速度の 2 倍である場合、CPU_RATIO は 2 に設定する必要があります。フェデレーテッド・サーバーの CPU 速度がリモート・サーバーの CPU 速度の 3 分の 1 である場合、CPU_RATIO は 0.33 に設定する必要があります。

CPU 比率サーバー・オプションを明示的に設定しない場合、フェデレーテッド・オプティマイザーはデフォルト値 1 を使用します。これは、フェデレーテッド CPU 速度とデータ・ソース CPU 速度が等しいことを示します。

比率が低いということは、データ・ソースのサーバー CPU がフェデレーテッド・サーバー CPU よりも速いことを示します。比率が低い場合、オプティマイザーは、 CPU に処理が集中する操作をデータ・ソースにプッシュダウンすることを検討します。低い比率とは、1 未満の値です。

入出力速度の相対比率

この値は、フェデレーテッド・サーバーの入出力速度と、リモート・データ・ソースが置かれているサーバー上の入出力速度との間の比率を示します。

この値は、フェデレーテッド・サーバーの入出力速度を、リモート・サーバーの入出力速度で除算したものと定義されます。例えば、フェデレーテッド・サーバーの入出力速度がリモート・サーバーの入出力速度の 2 倍である場合、IO_RATIO は 2 に設定する必要があります。フェデレーテッド・サーバーの入出力速度がリモート・サーバーの入出力速度の 2 分の 1 である場合、IO_RATIO は 0.5 に設定する必要があります。

入出力比率サーバー・オプションを明示的に設定しない場合、フェデレーテッド・オプティマイザーはデフォルト値 1 を使用します。これは、フェデレーテッド・サーバーとリモート・サーバーの両方の入出力速度が等しいことを示します。

1 未満の低い IO_RATIO は、リモート・サーバーがフェデレーテッド・サーバーより高い入出力速度を持っていることを示します。この場合、オプティマイザーは入出力集中操作をリモート・データ・ソースにプッシュダウンする傾向があります。低い比率とは、1 未満の値です。

フェデレーテッド・サーバーとデータ・ソース間の通信レート

通信レートが低いということは、フェデレーテッド・サーバーとデータ・ソース間のネットワーク通信が遅いことを示します。

COMM_RATE サーバー・オプションの設定値により、通信レートが決定されます。COMM_RATE は、データ・ソース・サーバーとフェデレーテッド・サーバーとの間のネットワーク接続の速度を表します。レートは秒当たりの MB 単位で表されます。デフォルトは 2 MBPS です。

通信レートが低い場合、照会オプティマイザーは、このデータ・ソースと送受信するメッセージの数を減らそうとします。 COMM_RATE サーバー・オプションを非常に小さい数にすると、オプティマイザーは、最小のネットワーク・トラフィックとなる照会を作成します。

データ・ソースの照合シーケンス

照合シーケンスの選択によっては、フェデレーテッド・データベースのパフォーマンスに影響を与える可能性があります。データ・ソースの照合シーケンスがローカル・フェデレーテッド・データベースの照合シーケンスと一致するかどうかを、COLLATING_SEQUENCE サーバー・オプションを使用して示すことができます。

COLLATING_SEQUENCE サーバー・オプションが、データ・ソースとフェデレーテッド・データベースの照合シーケンスが一致することを示す場合、フェデレーテッド・サーバーは、文字データに関係した順序に依存する処理をデータ・ソースにプッシュダウンすることができます。データ・ソースの照合シーケンスがフェデレーテッド・データベースの照合シーケンスと一致しない場合、オプティマイザーはこのデータ・ソースから検索したデータを、順序付けされていないと見なします。フェデレーテッド・データベースは関係するデータを検索し、文字データの順序に依存する処理をすべてローカルで行いますが、その結果、照会の速度は低下し、パフォーマンスに影響を与える場合があります。

リモート・プランのヒント

プラン・ヒントはステートメントの一部分であり、データ・ソース・オプティマイザーに対しての追加情報を提供します。

リモート・プランのヒントを生成するには、PLAN_HINTS サーバー・オプションを使用します。照会のタイプによっては、この情報を利用することで照会のパフォーマンスを改善できます。プラン・ヒントは、データ・ソース・オプティマイザーが索引を使用するかどうか、どの索引を使用するか、またはどの表結合シーケンスを使うかを判別するのに役立ちます。

そのサーバー・オプションが照会のパフォーマンスを向上させるかどうかを判別するために、いくつかのテストを実行する必要があります。

照会に独自のプラン・ヒントをコーディングすることはできません。

プラン・ヒントが使用可能な場合、データ・ソースに送信される照会には追加情報が入っています。例えば、プラン・ヒント付きで Oracle オプティマイザーに送信されるステートメントは次のようなものです。

```
SELECT /*+ INDEX (table1, tlindex)*/
 col1
 FROM table1
```

プラン・ヒントは、ストリング `/*+ INDEX (table1, t1index)*/` です。

グローバルな最適化に影響を与えるニックネーム特性

グローバルな最適化に影響を与えるニックネーム特有の要因がいくつかあり、代表的なものとして、索引情報およびグローバル・カタログ統計情報があります。

SQL コンパイラーが使用できる索引情報とグローバル・カタログ統計データを、常に最新のものにしておくことが重要です。

索引の仕様

SQL コンパイラーは、索引情報を使用して照会を最適化します。

データ・ソース表の索引情報は、その表のニックネームを作成した時にのみ獲得されます。ニックネームが作成された後、データ・ソース表上の索引に加えた変更は、フェデレーテッド・サーバー上では更新されません。リモート索引情報が変更された場合、表のニックネームをドロップし、ニックネームを再作成することによって、フェデレーテッド・サーバー上に保管されている索引情報を更新できます。あるいは、データ・ソース表に新規索引が追加された場合、フェデレーテッド・サーバー上のニックネームの索引の仕様を定義することもできます。

索引を持たないオブジェクト (例えば、ビュー、シノニム、または非リレーショナルのデータ・ソース・オブジェクト) についてのニックネームに関する索引情報は収集されません。

ニックネームが定義されているオブジェクトが索引を持たない場合、このオブジェクトに「索引の仕様」を作成することができます。「索引の仕様」は、グローバル・カタログ内に索引の定義を作成します。「索引の仕様」は実際の索引ではあり

ません。「索引の仕様」を作成するには、CREATE INDEX ステートメントに SPECIFICATION ONLY 文節を指定します。ニックネームに索引の仕様を作成する構文は、ローカル表に索引を作成する構文と似ています。

次の場合は、「索引の仕様」の作成を検討してください。

- 表に新しい索引が作成された。
- 索引を持たないデータ・ソース・オブジェクト (ビューやシノニムなど) にニックネームを作成した場合。

ニックネームに索引の仕様 (SPECIFICATION ONLY) を作成し、索引をユニークと指定した場合、フェデレーテッド・データベースはリモート表の列値がユニークであることを検査しません。リモート列値がユニークでない場合、索引列を含むニックネームに対する照会は、誤ったデータを戻すか、またはエラーになることがあります。

データ・ソース・ビューのニックネームに対して CREATE INDEX...SPECIFICATION ONLY ステートメントを実行する前に、その必要性を検討してください。

- リモート・ビューが、索引を持つデータ・ソース表上の単純な SELECT ステートメントであれば、データ・ソース表の索引と一致するニックネームの索引の仕様を作成することにより、照会のパフォーマンスは飛躍的に向上します。
- 単純な SELECT ステートメントではない (例えば、2 つの表を結合して作成されたビュー) リモート・ビューに索引の仕様を作成すると、照会パフォーマンスは悪化する場合があります。

例えば、2 つの表を結合するリモート・ビューに対して作成された索引の仕様を考慮します。オプティマイザーはこのビューを、ネストされたループ結合内の内部エレメントとして選択する可能性があります。この場合、結合を何回も評価することになり、照会のパフォーマンスは悪くなります。代替案としては、データ・ソース・ビューで参照されているそれぞれの表にニックネームを作成し、この両方のニックネームを参照するフェデレーテッド・ビューを作成することです。

グローバル・カタログ統計情報

フェデレーテッド・サーバー上のグローバル・カタログには、照会の最適化に使用する統計情報が含まれています。

フェデレーテッド・サーバーは、ニックネームに関係した照会を最適化するためにニックネームが定義されているデータ・ソース・オブジェクトの統計に依存しています。これらの統計情報は、CREATE NICKNAME ステートメントを使用してデータ・ソース・オブジェクトにニックネームを作成した時に検索されます。フェデレーテッド・データベースはデータ・ソース側にオブジェクトが存在するかチェックし、次に既存のデータ・ソース統計データを収集します。オプティマイザーにとって役に立つ情報は、データ・ソース・カタログから読み取られ、フェデレーテッド・サーバー上のグローバル・カタログに置かれます。オプティマイザーは、データ・ソースのカタログ情報の一部、またはすべてを使用するので、ニックネームを作成する前に、データ・ソース側で (RUNSTATS と同等のデータ・ソース・コマンドを使用して) 統計情報を更新しておくことをお勧めします。

カタログ統計情報は、表およびビューの全体的なサイズ、および関連する列の値の範囲を記述しています。検索される情報には、以下のものが含まれています (ただし、これらに限定されるわけではありません)。

- ニックネーム・オブジェクト内の行数
- ニックネームが占めるページ数
- 表の各列の個別値の数
- 索引の列の個別値の数
- 列の最高値/最低値

フェデレーテッド・データベースはデータ・ソース側の統計データを検索することはできますが、データ・ソース側の既存の統計データが変更されても、それを自動的に検出することはできません。さらに、フェデレーテッド・データベースは、オブジェクト定義や、データ・ソース側のオブジェクトの構造の変更 (例えば、表に列を追加した場合など) を処理するメカニズムを持っていません。

ニックネームが定義されているリモート・オブジェクトの統計データまたは構造特性が変更された場合、統計情報を更新するには以下の方法があります。

- 手動による統計収集
  - データ・ソース側で RUNSTATS に相当するものを実行します。次に、現行のニックネームをドロップし、ニックネームを再作成します。統計情報を更新する場合は、この方法が推奨されています。

    この方式の利点は、更新された統計に加え、新規索引についての情報またはリモート・オブジェクトの構造の変更が、新規ニックネームに反映されるということです。このメソッドの欠点は、古いニックネームを基にするビューまたはパッケージは無効になるということです。
  - コマンド行プロセッサーから使用できる SYSPROC.NNSTAT() ストアード・プロシージャーを使用します。

    SYSPROC.NNSTAT() は、ニックネーム統計を更新するだけです。構造の変更をリモート・オブジェクトに一致させるようにニックネームを変更することはありません。例えば、リモート・オブジェクトに新規列がある場合、ニックネーム統計を更新しても、列をニックネームに追加しません。
  - SYSSTAT.TABLES カタログ・ビュー内の統計情報を手作業で更新します。この方法は、リモート・データ・ソース上の統計情報が不正または不完全であることが分かっている場合にのみ使用してください。
- 自動統計収集

  このフィーチャーは、デフォルトで実行され、最新の表およびニックネーム統計を自動的に収集することによってパフォーマンスを向上させます。

行の変更を反映する

データ・ソース側で大量の行がオブジェクトに追加されたか、またはオブジェクトから削除された場合、ニックネームのカタログ統計は引き続き古い行数を示すので、フェデレーテッド・データベースはこれらの変更に気が付きません。

ただし、オプティマイザーはすでに正確ではないニックネーム統計情報に基づいて引き続き決定を下すので、パフォーマンスの低下に気付く場合があります。データ・ソース側でリモート・オブジェクトの統計情報を更新した後に、ニックネームの統計情報を更新することで、オプティマイザーは、データ・ソースに対する照会を処理するためのアクセス・プランの生成および選択時に、正確な統計情報を使用することができます。

列の変更時の統計の更新

例えば列が表に加えられるなどの、データ・ソース・オブジェクトの構造の変更がある場合、フェデレーテッド・データベース・カタログ内のそのオブジェクトの統計情報を更新するには、いくつかのステップを実行する必要があります。

このタスクについて

このタスクについて

データ・ソース側で列が追加、削除、または変更された場合、ユーザーは間違った結果やエラー・メッセージを受け取る可能性があります。例えば、ニックネーム *EUROSALES* が Sybase データベース内の *europe* 表を指すとします。この表に *CZECH* と呼ばれる新しい列が追加された場合、フェデレーテッド・データベースは *CZECH* 列に気が付きません。この列を照会で参照すると、エラー・メッセージが出されます。

手順

列の変更時にそのオブジェクトの統計を更新するには、以下のようにします。

手順

1. データ・ソース側で、DB2 RUNSTATS に相当するユーティリティーを実行します。これにより、データ・ソース・カタログ内の統計情報が更新されます。
2. DROP NICKNAME ステートメントを使用して、データ・ソース・オブジェクトの現在のニックネームをドロップします。
3. CREATE NICKNAME ステートメントを使用してニックネームを再作成します。

グローバルな最適化の分析

アクセス・プランに関する詳細情報 (グローバル・オプティマイザーが最適のプランを選択するために使用する一部の情報を含む) は、実際のアクセス・プラン自体とは別に Explain 表に保持されます。

この情報により、アクセス・プランを詳細に分析できます。 Explain 表は、サポートされるすべてのオペレーティング・システムでアクセスでき、静的と動的の両方の SQL ステートメントの情報を含んでいます。 Explain 表は SQL ステートメントを使用してアクセスできます。これにより、異なる照会を比較したり、同じ照会を長期にわたって比較する場合、出力の操作を簡単に行うことができます。

Explain 表からグローバル・アクセス・プラン情報を入手するには、次の複数の方法があります。

- Explain 表フォーマット・ツールの db2exfmt を使用して、Explain 表からの情報を事前定義されたフォーマットで表示します。
- db2expln および dynexpln ツールを使用すると、特定の SQL ステートメント用に選択されたアクセス・プランを知ることができます。

  db2exfmt、Visual Explain、db2expln、または dynexpln の出力を完全に理解するためには、以下を理解していなければなりません。
  - サポートされている別の SQL ステートメント、およびそのステートメントに関連した用語 (SELECT ステートメントの述部など)
  - パッケージ (アクセス・プラン) の目的
  - システム・カタログ表の目的と内容
  - 結合、GROUP BY、集約、およびソートなどの基本的な照会処理演算子。

アクセス・プランの最適化の判断の概要

このセクションでは、典型的な最適化に関する質問、およびパフォーマンスを改善するために調査できるエリアをリストしています。

なぜ、同じデータ・ソースの 2 つのニックネーム間の結合がリモート側で評価されないのか ?

結合操作のエレメント、結合述部、および結果内の行数を検査して、同じデータ・ソースの 2 つのニックネーム間の結合がリモートで評価されない理由を評価することができます。

次のような調査領域があります。

- 結合操作。データ・ソースは結合をサポートできるか?
- 結合述部。結合述部はリモート・データ・ソース側で評価できるか?
- 結合結果の行数。Visual Explain を使用すると行数がわかります。その結合は、2 つのニックネームを結合したよりもはるかに多い行数のセットを作成しないか? その数は実態を反映しているか ? 答えが「いいえ」の場合、SYSPROC.NNSTAT() ストアード・プロシージャーを使用して、ニックネームの統計情報を更新することを検討してください。

なぜ、GROUP BY 演算子はリモート側で評価されないのか ?

次のような調査領域があります。

- 演算子の構文。演算子がリモート・データ・ソース側で評価できるか検証してください。
- 行数。Visual Explain を使用して、GROUP BY 演算子の入力と出力の行数の見積もりをチェックしてください。この 2 つの数は非常に近いですか? 答えが「はい」ならば、オプティマイザーは、この GROUP BY をローカルで評価した方がより効率的と見なします。また、これら 2 つの数は実態を反映していますか ? 答えが「いいえ」の場合、SYSPROC.NNSTAT() ストアード・プロシージャーを使用して、ニックネームの統計情報を更新することを検討してください。

なぜ、そのステートメントはリモート側で完全に評価されないのか？

フェデレーテッド・サーバーは、フェデレーテッド照会で入手した照会セマンティクスおよび結果が、DB2 Database for Linux, UNIX, and Windows で完全に評価された場合とまったく同じになるようにします。照会コンパイラーのプッシュダウン分析フェーズは、リモート・ソースへのプッシュダウン処理が DB2 セマンティクスを維持するかどうかを決定します。したがって、フェデレーテッド照会操作は、リモート・ソースでの対応する操作が同じ意味および結果を持つ場合にのみ、安全にプッシュダウンすることができます。単一のリモート・ソースへの照会の完全なプッシュダウン処理が失敗する最も一般的な理由は、照会内の 1 つ以上の操作について、フェデレーテッド・サーバーとリモート・ソースとの間の機能に、微妙な違いが存在することです。

オプティマイザーはコストに基づく最適化を行います。プッシュダウン分析によって、すべての演算子をリモート・データ・ソースで評価できると示されても、オプティマイザーはさらにコスト見積もりをした上で、グローバルな最適プランを生成します。そのプランの選択の決定に寄与する数多くの要因があります。元の照会にあるすべての操作をリモート・データ・ソースが処理できるとします。しかし、その CPU 速度はフェデレーテッド・サーバーの CPU 速度よりはるかに遅いとします。この場合、フェデレーテッド・サーバーで操作をした方がより有利である可能性があります。望ましいパフォーマンスが得られない場合は、SYSSTAT.SERVEROPTIONS カタログ表内のサーバー統計情報を調べてください。

なぜ、オプティマイザーが生成し、完全にリモート側で評価されるプランが、リモート・データ・ソース側で直接実行される元の照会よりもはるかにパフォーマンスが悪いのか ？

次のような調査領域があります。

- 照会オプティマイザーにより生成されるリモート SQL ステートメント。対応するリモート表名によりニックネームを置き換えることに加え、生成されるリモート SQL ステートメントは、一般に元のフェデレーテッド・ステートメントとは以下のように異なります。
  - 照会内の述部の順序は変更されている場合があります。
  - 元の照会内にある述部は除去され、同等の述部で置き換えられているか、または追加の述部が加えられている場合があります。
  - 副照会が結合として書き直されている場合があります。
  - 変換またはストリング切り捨てを実行する追加の関数が、DB2 のセマンティクスを維持するために追加されている場合があります。

  上記のリストの最後の項目を例外として、これらの変更は通常はパフォーマンスに好ましい影響を与えます。ただし場合によっては、変更によりリモート照会オプティマイザーが、元の照会とは異なる (そして低速の) プランを生成する場合があります。

  よい照会オプティマイザーは、照会の述部の順序付けに影響を受けないはずです。残念ながら、すべての DBMS オプティマイザーが同じというわけではありません。リモート・データ・ソース側のオプティマイザーが、入力の述部の順序

付けに基づいて異なるプランを生成することはありえます。この場合、これはリモートのオプティマイザーに付随する問題です。述部の順序付けを変更するか、またはリモート・データ・ソースのサービス部門に支援を求めてください。

また、述部の置き換えも調べてください。よい照会オプティマイザーは、同等の述部の置き換えに影響を受けないはずです。リモート・データ・ソース側のオプティマイザーが、入力の述部に基づいて異なるプランを生成することはありえます。例えば、あるオプティマイザーは、述部のための推移的閉包 (transitive closure) ステートメントを生成できません。

- 戻される行数。この数は Visual Explain から得られます。照会が大量の行を戻す場合、ネットワーク・トラフィックがボトルネックになる可能性があります。
- 追加の関数。リモート SQL ステートメントに、元の照会にはない、追加の関数が含まれていませんか? データ・タイプを変換するために、余分な関数がいくつか生成される場合があります。これらが必要なものかどうかを確認してください。

第 29 章 パフォーマンスに影響するシステム・モニター・エレメント

フェデレーテッド・データベース・システム・モニターは、データベース・マネージャーの現在の状態に関する統計情報、およびデータベース処理のカウンターやその他の測定などの、アクティビティー情報を収集します。

フェデレーテッド・システムでは、データベース・システム・モニターを使用して、データベース・アクティビティー、システム・パフォーマンス、およびアプリケーション・パフォーマンスについての情報を収集できます。

タイム・スタンプ・モニター・スイッチは、フェデレーテッド・データベースとデータ・ソースとの対話の応答時間をトラッキングするのに使用されます。タイム・スタンプ・スイッチによってトラッキングされるフェデレーテッド・データ・エレメントは、以下のとおりです。

- ニックネーム作成応答時間
- 削除応答時間
- 挿入応答時間
- パススルー時間
- 照会応答時間
- リモート・ロック時間
- ストアード・プロシージャー時間
- 更新応答時間

タイム・スタンプ・モニター・スイッチのデフォルトの設定値は ON です。

推奨: タイム・スタンプ・モニター・スイッチの設定値を、すべてのアプリケーションに対して OFF にすることで、パフォーマンスを向上させることができます。1 つのアプリケーションでタイム・スタンプ・スイッチが ON に設定されていると、システムは応答時間の収集を継続します。そのため、一部のアプリケーションのタイム・スタンプ・スイッチをオフにしただけでは、パフォーマンスを向上させることはできません。

スイッチをオフにすることには、その他の意味があります。

- すべてのアプリケーションでタイム・スタンプ・モニター・スイッチをオフにする場合、変更をインプリメントするためには、DB2 インスタンスを停止し、再始動する必要があります。
- タイム・スタンプ・モニター・スイッチをオフにすると、フェデレーテッドおよび非フェデレーテッド・アプリケーションの両方のタイム・スタンプ情報の収集が使用不可になります。ローカル・データベースも、タイム・スタンプ情報を受け取りません。

ローカルの非フェデレーテッド・アプリケーションのタイム・スタンプ情報が必要な場合は、タイム・スタンプ・モニター・スイッチをオフにしてはいけません。

このコマンドを使用して、すべてのアプリケーションに対してタイム・スタンプ・スイッチを OFF にできます。

```
update dbm cfg using dft_mon_timestamp off
```

その後で、次のコマンドを発行します。

```
db2stop
db2start
```

フェデレーテッド・サーバーの停止と開始を実行することによって、スイッチがすべてのアプリケーションに対してオフになっていることが確認できます。

タイム・スタンプ・スイッチによってトラッキングされる各エレメントについての具体的な情報は、別のトピックで説明します。

第 30 章 マテリアライズ照会表

マテリアライズ照会表は、照会の結果をキャッシュに入れる表です。同じ照会を再びサブミットしたときに、データベース・エンジンは、マテリアライズ照会表からそのデータを戻すことができます。ニックネームを含むマテリアライズ照会表を使用して、照会のパフォーマンスを向上させることができます。

マテリアライズ照会表は、キャッシュ表を作成するときに使用されます。キャッシュ表には、関連するマテリアライズ照会表によって定義されているローカル・データが保管されます。

マテリアライズ照会表とフェデレーテッド・システムの概説

マテリアライズ照会表は、照会の結果をキャッシュに入れる表です。同じ照会を再びサブミットしたときに、データベース・エンジンは、照会の計算を繰り返す代わりに、マテリアライズ照会表からそのデータを戻すことができます。

ニックネームを含むマテリアライズ照会表を使用して、照会のパフォーマンスを向上させたり、論理の一部をカプセル化することができます。マテリアライズ照会表は、キャッシュ表を作成するときに使用されます。

SQL オプティマイザーは、基本表やニックネームよりもマテリアライズ照会表を使用したほうが照会をより効率的に実行できるかどうかを判別します。オプティマイザーは次の要素を考慮して、マテリアライズ照会表を選択します。

- マテリアライズ照会表は照会の全部または一部と一致していなければなりません。
- リフレッシュ経過時間基準と一致していなければなりません。
- マテリアライズ照会表を使用するアクセス・プランは、基本表またはニックネームを使用するアクセス・プランよりコストが低くなければなりません。

以下のデータ・ソースからのオブジェクトのニックネームを含むマテリアライズ照会表がサポートされます。

- リレーショナル・データ・ソース
  - DRDA
  - Informix
  - JDBC
  - ODBC
  - Oracle
  - Sybase
  - Microsoft SQL Server
  - Teradata
- 非リレーショナル・データ・ソース
  - BioRS
  - Excel
  - 表構造ファイル

– Web サービス
– XML

フェデレーテッド・マテリアライズ照会表の作成

マテリアライズ照会表は、データをローカルにキャッシュに入れて、照会のパフォーマンスを向上させるために使用します。リレーショナルおよび非リレーショナルのデータ・ソースからのニックネームを使用して、マテリアライズ照会表を作成することができます。

このタスクについて

制約事項

- 『データ・ソースに固有の、マテリアライズ照会表の制約事項』
- 照会の述部または選択リストに関数テンプレートがある場合、その関数テンプレートはマテリアライズ照会表の一部でなければなりません。
- 297 ページの『ニックネームを含むマテリアライズ照会表の使用に関する制約事項』

手順

マテリアライズ照会表を作成するには、組み込むリモート・データ・ソース・オブジェクトを表すニックネームを参照する CREATE TABLE ステートメントを発行します。

ユーザー保守マテリアライズ照会表には、副選択ステートメントで INSERT ステートメントを使用してデータを追加することができます。例:

```
insert into my_mqt (select ..from n1, n2 where ..)
```

ここで、照会の選択部分は、マテリアライズ照会表定義と一致します。オプティマイザーは、my_mqt を使用して照会の選択部分を置き換える場合があります。この場合、ステートメントは次のようになります。

```
insert into my_mqt (select .. from my_mqt);
```

この場合、マテリアライズ照会表は INSERT 操作のソースになります。これが起きないようにするには、以下のコマンドの 1 つを発行して、マテリアライズ照会表を一時的に使用不可にすることができます。

```
SET CURRENT REFRESH AGE 0
SET CURRENT MAINTAINED TABLE TYPE FOR OPTIMIZATION SYSTEM
```

データ・ソースに固有の、マテリアライズ照会表の制約事項

マテリアライズ照会表を作成する場合、個別のデータ・ソースの制約事項を知っている必要があります。

このトピックでは、次のデータ・ソースについて、マテリアライズ照会表を作成する場合の制約事項を説明します。

- BioRS
- 表構造ファイル
- Web サービス

- XML

BioRS 検索の制約事項

BioRS ラッパーの WHERE 文節には、少なくとも 1 つの述部が必要です。ラッパーの述部要件を持たすマテリアライズ照会表を作成しなければなりません。述部を指定しないと、マテリアライズ照会表のリフレッシュが失敗します。

表構造ファイルの制約事項

表構造ファイルのニックネームに DOCUMENT オプションを指定して定義している場合は、ファイル・パスを指定した述部がマテリアライズ照会表になければなりません。述部を指定しないと、マテリアライズ照会表のリフレッシュが失敗します。

Web サービスの制約事項

マテリアライズ照会表は、ニックネーム階層のフラット・ビューに対してのみ作成することができます。階層のニックネームごとにマテリアライズ照会表を作成することはできません。

XML の制約事項

マテリアライズ照会表は、子テーブル上に作成することはできません。

XML 表のニックネームに DOCUMENT オプションを指定して定義している場合は、マテリアライズ照会表にはファイル・パスを指定した述部が必要です。述部を指定しないと、マテリアライズ照会表のリフレッシュが失敗します。

ニックネームを含むマテリアライズ照会表の使用に関する制約事項

フェデレーテッド・システムの調整時には、ニックネームを参照するマテリアライズ照会表の制約事項を考慮する必要があります。

データ・ソース・オブジェクト用のラベル・ベースのアクセス制御 (LBAC)

ラベル・ベースのアクセス制御または Oracle Label Security を使用するデータ・ソース・オブジェクトのニックネームはキャッシュに入れられず、それらに対するマテリアライズ照会表を作成することはできません。

システム保守のマテリアライズ照会表

フェデレーテッド・システムは、パーティション・データベース環境でニックネームを参照する、システム保守のマテリアライズ照会表をサポートしていません。

フェデレーテッド再書き込み機能によって照会の経路を MQT に指定するには、フェデレーテッド・ラッパーが fenced である必要があります。ラッパーを FENCED として作成するか、ALTER WRAPPER ステートメントを使用してラッパーを変更することができます。例:

```
CREATE WRAPPER <wrapper name>
   LIBRARY <libname>
   OPTIONS (DB2_FENCED  'N');
```

```
ALTER WRAPPER <wrapper name> OPTIONS (SET DB2_FENCED 'Y');
```

データベース構成パラメーター DFT_MTTB_TYPES、または SQL より前に発行された設定済みの現在の特殊レジスターを使用して、最適化のために保守する表タイプを ALL または USER に設定する必要があります。

構成パラメーターの値を USER に変更するには、以下を発行します。

```
update db cfg for <dbalias> using DFT_MTTB_TYPES USER
```

設定済みの現在の特殊レジスターを使用するには、以下を発行します。

```
set current maintained table types for optimization ALL
```

この制約に対処するために、ユーザー保守のマテリアライズ照会表を使用することができます。

例えば、DEPART という名前の非リレーショナル・ニックネームの場合、次のコマンドを発行してシステム保守のマテリアライズ照会表をシミュレートすることができます。

```
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION ALL;

CREATE TABLE AST1(C1, C2)
 AS (SELECT EMPNO, FIRSTNME FROM DEPART WHERE EMPNO>'000000')
 DATA INITIALLY DEFERRED REFRESH DEFERRED
 ENABLE QUERY OPTIMIZATION MAINTAINED BY USER;

SET INTEGRITY FOR AST1 ALL IMMEDIATE UNCHECKED;

INSERT INTO AST1 (SELECT EMPNO, FIRSTNME FROM DEPART WHERE EMPNO>'000000');

SET CURRENT REFRESH AGE ANY;
```

次の SELECT ステートメントを実行すると、上で定義したマテリアライズ照会表からの応答を受け取ることができます。

```
SELECT EMPNO, FIRSTNME FROM DEPART
 WHERE EMPNO > '000000' AND FIRSTNME LIKE 'AN%';
```

第 31 章 キャッシュ表

頻繁にアクセスしてもほとんど変更のないデータを保管する場合は、キャッシュ表を使用します。

キャッシュ表を使用すると、データ・ソースのデータに直接アクセスする代わりに、データをローカルに格納できるので、照会のパフォーマンスが向上します。

以下のデータ・ソースのデータをキャッシュに入れることができます。

- DB2 ファミリー
- Informix
- Microsoft SQL Server
- Oracle
- Sybase

キャッシュ表は次のコンポーネントで構成されています。

- フェデレーテッド・データベース・システム上のニックネーム。ニックネームには、データ・ソース表と同じ列定義とデータ・アクセスがあります。
- ニックネームに定義する 1 つ以上のマテリアライズ照会表。このタイプのマテリアライズ照会表は、FEDERATED_TOOL によって保守されるマテリアライズ照会表です。通常、このマテリアライズ照会表には、データ・ソース表のデータのうち、使用頻度の高いデータのサブセットが含まれています。
- 各マテリアライズ照会表のレプリケーション・スケジュール。レプリケーション・スケジュールによって、ローカルのマテリアライズ照会表をデータ・ソース表と同じ最新の状態に保つことができます。レプリケーション・スケジュールは、ユーザーが定義しなければなりません。

以下は、キャッシュ表を示した図です。

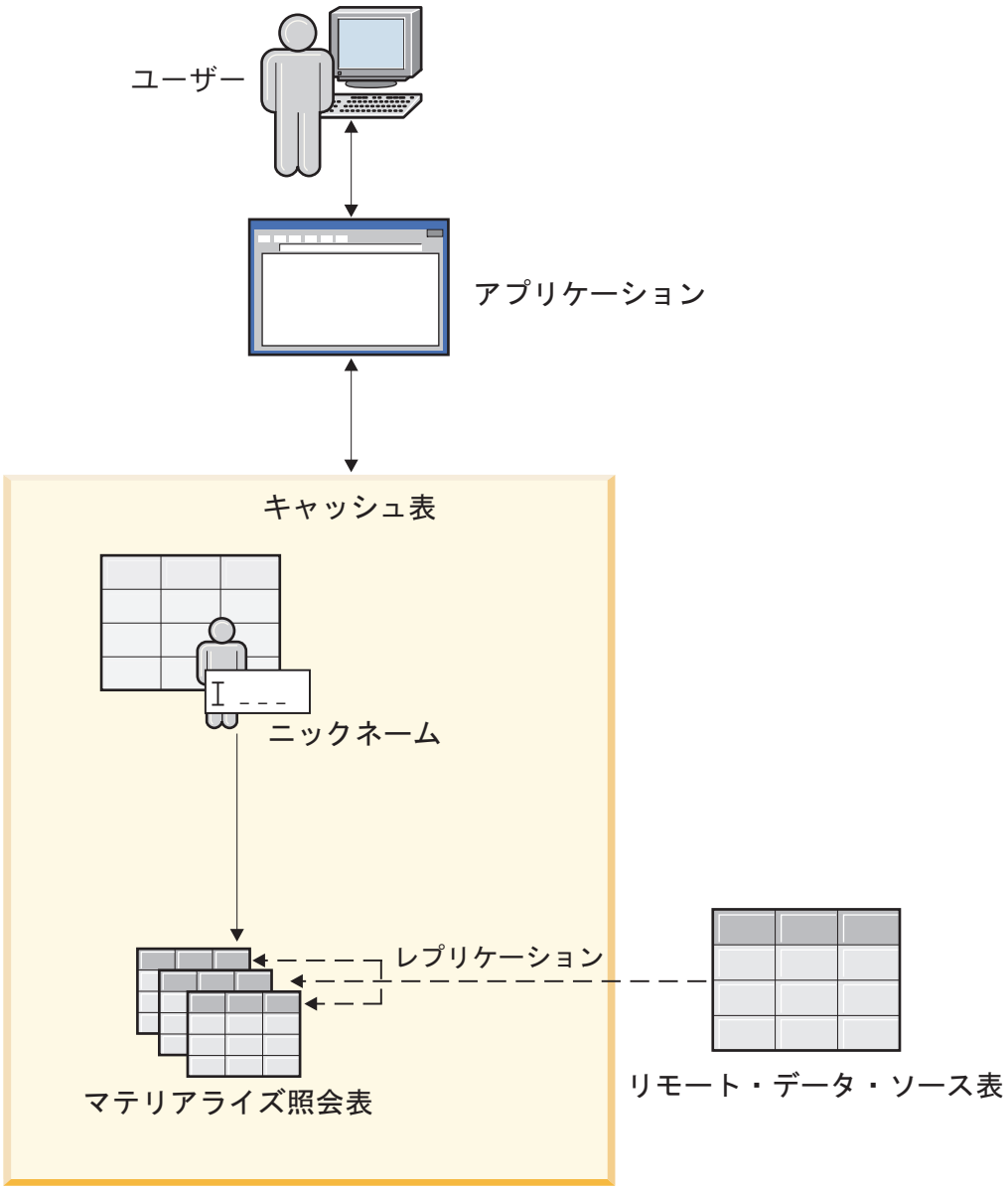


図 14. キャッシュ表。

キャッシュ表には、ニックネームと同じ名前があります。キャッシュ表は、1 つのデータ・ソース表にしか関連付けることができません。

キャッシュ表が使用可能になっている場合に、照会の要求先のデータがマテリアライズ照会表にあれば、照会オプティマイザーは照会をキャッシュ表に送信します。

第 32 章 サンプル・キャッシュ表の作成

サンプルを使用して分散キャッシングをセットアップすることができます。

始める前に

- フェデレーテッド・サーバーで FEDERATED パラメーターを YES に設定します。FEDERATED パラメーターは、データベース・マネージャー構成パラメーターです。
- Informix データ・ソースにアクセスするには、Informix Client Software Development Kit (SDK) をフェデレーテッド・サーバーにインストールして構成する必要があります。
- DB2 Database for Linux, UNIX, and Windows の表からデータをキャッシュに入れるには、DB2データベースでアーカイブ・ログを構成する必要があります。
- フェデレーテッド・データベースまたはソース・データベースは、キャッシュ表を作成するコンピューターになければなりません。フェデレーテッド・データベースまたはソース・データベースがローカルに置かれていない場合は、それらのデータベースのカタログをローカル・コンピューターに作成しておく必要があります。データベースをカタログするときに使用する別名は、データベース名と同じでなければなりません。
- データベース間のユーザー・マッピングで使用するユーザー ID には、ソース・データベースに表を作成する権限が必要です。

このタスクについて

サンプルの DB2cacheTables.zip には、以下のタスクの実行に使用できるコマンド・スクリプトが用意されています。

- フェデレーションを使用可能にし、キャッシュ・データベースのフェデレーテッド・オブジェクトを作成する。
- 1 つまたは複数のマテリアライズ照会表 (MQT) を作成してキャッシングに加わる。
- 更新されたキャッシュ表を維持するためにレプリケーションをセットアップする。

手順

README ファイルの説明に従って、DB2cacheTables.zip サンプルのコマンドを実行します。

第 33 章 バルク挿入

データのバルク挿入を行うと、Netezza データ・ソースへの挿入操作のパフォーマンスを向上させることができます。

フェデレーションは、挿入操作の評価に基づいて、Netezza データ・ソースへのデータのバルク挿入をサポートします。

挿入操作にアクセス・プランの INSERT 演算子のソース表や複数の値が含まれる場合、フェデレーションはバルク挿入を自動的に実行します。データを Netezza ニックネームに挿入する場合には、通常、バルク挿入が選択されます。以下の挿入操作を使用します。

- 以下の例に示すように、副照会の結果セットから Netezza にデータを挿入します。

  ```
  INSERT INTO n1 SELECT c1, c2 FROM local_t1;
  ```

- 以下の例に示すように複数の値を Netezza に挿入します。

  ```
  INSERT INTO n1 VALUES (1, 'a'), (2, 'b');
  ```

バルク挿入を使用可能にするには、ODBC サーバー・オプションの ENABLE_BULK_INSERT を Y に設定します。

第 34 章 フェデレーテッド・サーバーのセキュリティー

フェデレーテッド・サーバーは、データの暗号化に Secure Socket Layer (SSL) を、また特定のデータ・ソースについては HTTP および SOCKS の各プロキシーをサポートしています。

暗号化

暗号化は、名前とパスワードだけによるセキュリティー・レベルより上のレベルのセキュリティーを提供します。エンドポイント間の暗号化に関するインターネット標準として、Secure Socket Layer (SSL) と Transport Layer Security (TLS) があります。SSL は、署名付き証明書を使用してセキュア通信を確保するためのものです。*証明書* は、ユーザーまたはサーバーの身元を保証するデジタル文書です。証明書は VeriSign などの認証局によって署名されますが、自己署名もできます。各通信パートナーは、特定の証明書を信頼のおけるものとして受け入れるかどうかを判別します。

各通信パートナーは、証明書ストア、つまり鍵ストアを持ちます。鍵ストアには、パートナーが受け入れる他のパートナーの証明書の他に、パートナー自身を表す証明書も保持されます。各エンドポイントは、通信を開くときに証明書を送信するかどうか、および証明書を提供しないパートナーからの通信を受け入れるかどうかを決定します。

ラッパーと関数では、以下の SSL 機能が関連します。

- サーバー・サイドでの送信する証明書の識別
- サーバー・サイドでのクライアント証明書の検査
- クライアント・サイドでのサーバー証明書の検査
- クライアント・サイドでの送信する証明書の識別
- クライアントとサーバーの両方での、ローカル鍵ストアに対するサポート・タイプ、ロケーションの特定、およびアクセス

SSL とプロキシー間の相互作用は、プロキシーのタイプによって異なります。一般に SSL 通信は、プロキシーを介してトンネリング (中継) されます。プロキシー・セッションは平文で確立されます。

IBM Global Security Kit (GSKit) は、ラッパーおよびユーザー定義関数のための暗号化サービスを提供します。

プロキシー

さまざまなインターネット・アタックを撃退するために、多くの企業がファイアウォールをインプリメントしています。ファイアウォール とは、通常はハードウェアとソフトウェアの両方で構成され、多くの場合はたとえば企業イントラネットとインターネットの通信境界に配置される、ネットワーク構成のことです。ファイアウォールは、その境界を越えるトラフィックを規制するゲートキーパーの役割を果た

します。ほとんどの場合、ファイアウォールは望ましくない通信がその境界を越えるのを阻止しますが、正当な通信がブロックされることがあります。

正当な通信がすべてファイアウォールをパススルーするようにするには、プロキシーをインプリメントします。*プロキシー* は、ファイアウォールを介して通信する権限があるサーバー・プログラムです。ユーザー・プログラムがリモート・サーバーに接続する必要がある場合、ユーザー・プログラムはプロキシーに対して要求し、プロキシーがリモート・サーバーに接続します。接続を行った後、プロキシーはユーザー・プログラムとリモート・サーバー間のトラフィックを制御します。ユーザー・プログラムはこのプロセスによってファイアウォールを越えることができるようになるので、セキュリティーが強化されます。つまり、リモート・サーバーが認識しているのはプロキシーのアドレスだけであり、ユーザー・プログラムのアドレスは知りません。

SOCKS および HTTP プロキシー・サーバーは、ユーザー・プログラムとリモート・サーバー間のトラフィックを制御します。 SOCKS プロキシーはトランスポート層で作動し、2 つのアドレス間の任意のトランスポート・メッセージ (TCP または UDP) を中継します。フェデレーテッド・サーバーは SOCKS4 と SOCKS5 の両方をサポートします。IPv4 をサポートする SOCKS4 は、ユーザー認証をサポートしていません。したがって、だれでもユーザー資格情報を提供する必要もなく SOCKS4 プロキシーを介して通信できます。 IPv6 をサポートする SOCKS5 は、いくつかのユーザー認証方式をサポートしています。 Internet Engineering Task Force (IETF) は SOCKS5 を標準として承認しました。詳しくは、www.ietf.org で RFC1928、RFC1929、および RFC1961 を参照してください。

SOCKS プロキシーを使用するには、あらかじめトランスポート層を構成してプロキシーを使用できるようにする必要があります。プロキシーへの接続を開いた後、トランスポート層はプロキシーに対して、リモート・サーバーへの接続を開くよう要求します。認証を必要とするように構成されている場合、SOCKS プロキシーはリモート・サーバーへの接続を開く前に、ID とパスワードの提供をプログラムに要求することができます。

HTTP プロキシーは、アプリケーション層プロトコルである HTTP プロトコルと連動します。プロキシーへの TCP/IP 接続を開いた後、ユーザー・プログラムはリモート・サーバーの名前を含む要求を送信します。その後ユーザー・プログラムは、プロキシー・サーバーを介して要求をサブミットし続けます。このプロセスは、リモート・サーバーが認証を必要とする場合は少し変わります。リモート・サーバーが認証を必要とする場合、プロキシー・サーバーはユーザー確認のためのプロキシー認証ヘッダーを含む応答メッセージを送信します。このヘッダーには、実行される認証の種類に関する情報が含まれます。ユーザー・プログラムは要求を再サブミットし、認証ユーザー確認への応答を含むプロキシー認証ヘッダーを組み込みます。

第 35 章 パブリック・ユーザー・マッピング

ご使用のフェデレーテッド・サーバーのセキュリティーを確保するために、パブリック・ユーザー・マッピングを使用したデータ・ソースへのアクセスについて制限事項があります。

InfoSphere Federation Server バージョン 9.7 フィックスパック 2 以降のすべてのデータ・ソースに対して、ローカル・データベース・ユーザーすべてを 1 つの許可 ID およびパスワードにマップできるようにする、パブリック・ユーザー・マッピングを作成できます。パブリック・ユーザー・マッピングを使用中、フェデレーテッド・サーバーのセキュリティーを確保するため、以下の制約事項が存在します。

パブリック・ユーザー・マッピングと非パブリック・ユーザー・マッピングは、同じサーバー定義に共存することはできない

ある特定のサーバー定義に対して許可されるのは、パブリック・ユーザー・マッピングまたは非パブリック・ユーザー・マッピングのどちらか 1 つのタイプのみです。あるタイプのユーザー・マッピングがサーバーに存在している場合に、他のタイプのユーザー・マッピングを作成しようとすると、エラー SQL1515N を受け取ります。例えば、パブリック・ユーザー・マッピングが、あるサーバーに定義されている場合、同じサーバーには、非パブリック・ユーザー・マッピングは作成できません。逆もまた同じです。非パブリック・ユーザー・マッピングが、あるサーバー定義に存在している場合に、そのサーバーにパブリック・ユーザー・マッピングを作成しようとすると、エラー SQL1515N を受け取ります。

パブリックと非パブリックのユーザー・マッピングの共存の制限は、アウトバウンド・フェデレーテッド・トラステッド接続の使用も制限します。
FED_PROXY_USER サーバー・オプションが、あるサーバー定義に存在している場合に、パブリック・ユーザー・マッピングを作成しようとするとエラー SQL1515N を受け取ります。パブリック・ユーザー・マッピングが存在している時に FED_PROXY_USER サーバー・オプションを追加するため ALTER SERVER ステートメントを実行しようとする場合も、エラー SQL1516N を受け取ります。

パブリック・ユーザー・マッピングは外部のユーザー・マッピング・リポジトリーからは取得しない

外部のユーザー・マッピング・リポジトリーでパブリック・ユーザー・マッピングを作成しようとする試み自体は制限されていませんが、フェデレーテッド・サーバーでは、セキュリティーを確保するために、それらマッピングの検索および取得を行いません。

第 36 章 フェデレーテッド・トラステッド・コンテキストおよびトラステッド接続

システム・パフォーマンスを強化して、ユーザー・マッピングの使用および保守を最小化または完全に削減します。

トラステッド・コンテキスト は、例えばアプリケーション・サーバーとフェデレーテッド・サーバーとの間またはフェデレーテッド・サーバーとリモート・データベース・サーバーとの間など、クライアントとデータ・ソースとの間の信頼関係を定義する、 DB2 データベース・オブジェクトです。トラスト・リレーションシップを定義するために、トラステッド・コンテキストはトラスト属性 を指定します。以下の 3 つのタイプのトラスト属性があります。

- 初期データベース接続要求を作成するシステム許可 ID
- 接続元となる IP アドレスまたはドメイン・ネーム
- データベース・サーバーとデータベース・クライアントとの間のデータ通信の暗号化設定

接続要求のすべての属性がサーバー上で定義されるトラステッド・コンテキスト・オブジェクトで指定されたトラスト属性と一致するとき、*トラステッド接続* が確立されます。明示的なトラステッド接続が確立された後、認証ありまたは認証なしで、同じ物理接続に対してユーザーを切り替えることができます。さらに、トラステッド接続内だけで使用できる特権を指定した役割をユーザーに付与できます。

この例は、BOSS のトラステッド・コンテキスト・オブジェクトを作成します。

```
CREATE TRUSTED CONTEXT MYCTX
 BASED UPON CONNECTION USING SYSTEM AUTHID BOSS
 ATTRIBUTES (ADDRESS '9.26.111.111')
 WITH USE FOR MARY WITH AUTHENTICATION ROLE MANAGER,
  PUBLIC WITHOUT AUTHENTICATION
 DEFAULT ROLE AUDITOR
 ENABLE
```

この例では、BOSS だけが IP アドレス 9.26.111.111 からトラステッド接続を開始できます。 Mary は接続を再利用できますが、最初に認証が必要となります。その後、Mary はこのトラステッド接続内で使用可能な特権を指定する MANAGER の追加役割を取得します。 PUBLIC として指定されるその他のユーザーは、接続の再利用が可能で、認証は必要ありません。これらの他のユーザーは、トラステッド接続内で使用可能な特権を指定する AUDITOR の追加役割を取得します。これらの追加特権は、ユーザーがトラステッド接続のアクティブ・ユーザーである間だけ使用可能となります。

トラステッド接続は明示的または暗黙的のどちらかです。接続のタイプによって、その接続を再利用できるかどうか、およびユーザーが追加役割を取得できるかどうかが決まります。

暗黙的なトラステッド接続 は、トラステッド接続が明示的に要求されなくても、接続属性がサーバー上のトラステッド・コンテキスト・オブジェクトのトラスト属性と一致する場合に確立されます。暗黙的なトラステッド接続が確立された後、トラ

ステッド接続の開始元だけが他の場合には使用可能でない役割を継承します。他のユーザーは、暗黙的なトラステッド接続を再利用できません。

明示的なトラステッド接続 は、アプリケーションが API を使用してトラステッド接続を要求するときに確立されます。接続属性がトラステッド・コンテキストのトラスト属性と一致する場合、トラステッド接続が確立されます。それ以外の場合は、通常の接続が確立されます。明示的なトラステッド接続が確立された後、他のユーザーは接続を再利用できます。接続の発信元および接続のユーザーの両方が、他の場合には使用可能でない追加の役割を継承します。

フェデレーテッド・トラステッド接続の利点

複数層のアプリケーション・モデルにおいて、フェデレーテッド・トラステッド接続は、単一の物理接続を再利用して各ユーザーの実際の ID を、層を介してデータベース・サーバーに伝搬します。

フェデレーテッド・トラステッド接続の利点を理解するために、典型的な複数層アプリケーション・モデルによくある問題について考えてみましょう。ある複数層アプリケーション・モデルは、エンタープライズ・ユーザー (層 1)、それらのユーザーが対話するアプリケーションが実行されているアプリケーション・サーバー (層 2)、アプリケーション・サーバーがすべてのデータベース・アクセスを経路指定する際の中継となるフェデレーテッド・サーバー (層 3)、およびフェデレーテッド・サーバーが通信を管理するさまざまなデータベース・サーバー (層 4) で構成されています。このモデルにおいて、アプリケーション・サーバーはユーザーを認証し、フェデレーテッド・サーバーとの対話を管理します。フェデレーテッド・サーバーは、ユーザー要求をデータ・ソース固有の形式に変換し、リモート・データ・ソースとの接続を確立し、要求をそれらのデータ・ソースに送信します。

このモデルでは、アプリケーション・サーバー ID とパスワードを使用してデータベース・サーバーへの接続を作成します。つまり、フェデレーテッド・サーバーは単に ID とパスワードをアプリケーション・サーバーからデータベース・サーバーに渡すだけです。データベース・サーバーは、この ID に関連付けられているデータベース特権を使用して、アプリケーション・サーバーが実行するすべてのトランザクションを許可および監査します。それには、アプリケーション・サーバーがエンタープライズ・ユーザーの代わりに実行するすべてのトランザクションが含まれます。

アプリケーション・サーバー ID を使用することには、以下のような問題があります。

- アプリケーション・サーバーがすべてのトランザクションを実行するため、トランザクションを実行するユーザーの実際の ID が分からない。
- ユーザーのトランザクションを監査できないため、トランザクションについてユーザーに責任を取らせることができない。
- アプリケーション・サーバー ID にはあらゆるユーザーが必要とするすべての特権のスーパーセットが必要なため、特権を最小限にするという原則に違反する。
- アプリケーション・サーバー ID が漏えいした場合はデータにぜい弱性がある。

フェデレーテッド・トラステッド接続では、これらの問題に対応しており、システムのセキュリティーとパフォーマンス向上につながる以下のような利点があります。

ユーザーの ID が分かる
ユーザーが接続上で切り替えられるため、データベースにアクセスするユーザーの実際の ID が分かります。

ユーザーに責任を求められる
フェデレーテッド・データベースおよびリモート・データ・ソース・データベースの監査ログでは、アプリケーション・サーバーがそれ自体の目的で実行したトランザクションと、個々のユーザーが実行したトランザクションが識別されます。そのため、特定のトランザクションに関して特定のユーザーに責任を求めることができます。

特権が制限される
トラステッド・コンテキストを作成すると、デフォルトのデータベース役割をすべてのユーザーに付与し、特定の役割を特定のユーザーに付与することができます。そのトラステッド・コンテキストの定義と一致するトラステッド・データベース接続だけが、その役割に関連付けられた特権を活用することができます。

データのぜい弱性が低くなる
フェデレーテッド・トラステッド接続を使用するシステムでは、アプリケーション・サーバー ID は、あらゆるユーザーに必要なすべての特権のスーパーセットを必要としません。そのため、万が一アプリケーション・サーバー ID が漏えいすることがあっても、ID があらゆるユーザーに必要なすべての特権のスーパーセットを持っている場合に比べて、データのぜい弱性は低く抑えられます。

管理保守が最小限ですむ
ユーザー・マッピングを作成して保守する必要性が大きく削減されます。

パフォーマンスが向上する
明示的なトラステッド接続が確立された後、フェデレーテッド・サーバーは、接続上の現在のユーザー ID を別のユーザー ID に切り替えることができます。その際、ユーザーの認証が必要な場合もあれば、不要な場合もあります。同じ物理接続を別のユーザーに対して再利用すると、パフォーマンスを向上させることができます。

フェデレーテッド・トラステッド接続のタイプ

フェデレーテッド・トラステッド接続は、エンドツーエンド・トラステッド接続またはアウトバウンド・トラステッド接続のどちらかです。どちらのタイプの接続が確立されるかは、システムをどのように構成するか、およびインバウンド接続要求がトラステッドであるかどうかに依存します。

典型的なフェデレーテッド構成は複数層です。つまり、アプリケーション・サーバー、フェデレーテッド・サーバー、およびリモート・データ・ソース・サーバーが含まれます。この構成では、フェデレーテッド・サーバーはアプリケーション・サーバーからインバウンド接続要求を受け取り、データ・ソース・サーバーへのアウトバウンド接続要求を送信します。

エンドツーエンド・フェデレーテッド・トラステッド接続

エンドツーエンド・フェデレーテッド・トラステッド接続は、接続の再利用を提供し、インバウンド接続およびアウトバウンド接続の両方の ID アサーションを提供します。例えば、フェデレーテッド・サーバーへのインバウンド接続が明示的であっても暗黙的であってもトラステッドである場合、フェデレーテッド・サーバーはアウトバウンド・トラステッド接続を自動的に要求します。データ・ソースが ID アサーション機能を提供する場合、トラステッド・アウトバウンド接続が確立されて、ユーザーの ID はリモート・データ・ソースに伝搬されます。インバウンドおよびアウトバウンド接続は別のユーザーがトラステッド接続の再利用を要求するたびに再利用されて、その新しいユーザーの ID はシステム全体に伝搬されます。

ID アサーション機能を提供しないデータ・ソースでは、フェデレーテッド・サーバーはエンドツーエンド ID アサーションを提供しますが、接続の再利用は提供しません。それらの場合、ユーザーが切り替わるごとに、フェデレーテッド・サーバーは直前のユーザー用のアウトバウンド接続をクローズして新しいユーザー用の新しい接続を作成します。そのようにして、アウトバウンド接続が再利用されなくても、ユーザーの ID はシステム全体に伝搬されます。

アウトバウンド・フェデレーテッド・トラステッド接続

アウトバウンド・フェデレーテッド・トラステッド接続は、データ・ソースが認証なしでトラステッド接続を再利用する機能を強化して、データ・ソース・パスワードをユーザー・マッピング内に保管する必要をなくします。インバウンド接続がトラステッドである場合、フェデレーテッド・サーバーは認証なしで再利用されるトラステッド・アウトバウンド接続を自動的に要求します。非トラステッド・インバウンド接続では、アウトバウンド・フェデレーテッド・トラステッド接続が、ユーザーの認証を必要としないで接続を再利用する機能を提供します。

この構成では、サーバー定義内の FED_PROXY_USER オプションを使用して、アウトバウンド接続を最初に確立した許可 ID を指定します。指定する許可 ID には、REMOTE_AUTHID オプションおよび REMOTE_PASSWORD オプションの両方を含むユーザー・マッピングが必要です。

リモート・データ・ソース・サーバー上でトラステッド・コンテキストを構成する方法に応じて、データベース・カタログ内のユーザー・マッピング数を最小で 1 まで削減できます。例えば、PUBLIC が認証なしで接続することを許可する場合、フェデレーテッド・プロキシー・ユーザーだけがユーザー・マッピングを必要とします。ただし、リモート・データ・ソース上のフェデレーテッド・トラステッド・コンテキストが認証を必要とすると指定している場合、またはユーザーがフェデレーテッド・サーバーとリモート・データ・ソースとで同じユーザー ID を使用しない場合は、ユーザーのユーザー・マッピングを作成または変更して、REMOTE_AUTHID オプションだけを使用する、 REMOTE_PASSWORD オプションだけを使用する、または REMOTE_AUTHID オプションと REMOTE_PASSWORD オプションだけを使用するようにする必要があります。さらに、USE_TRUSTED_CONTEXT オプションを Y に設定しなければなりません。

アウトバウンド・フェデレーテッド・トラステッド接続には多大な追加の構成タスクおよび保守タスクが関連しているために、可能な限り、フェデレーテッド・システムがエンドツーエンド・トラステッド接続をインプリメントするように設計して

ください。そして、確かに必要な場合にのみ、アウトバウンド・フェデレーテッド・トラステッド接続を使用してください。

フェデレーテッド・トラステッド接続用の API

トラステッド接続を要求および再使用するための API は、アプリケーションの種類によって異なります。

フェデレーテッド・トラステッド接続を要求するには、アプリケーションでこれらの API を指定する必要があります。

CLI/ODBC アプリケーション
クライアントがトラステッド接続を要求しているかどうかを示すには、属性 SQL_ATTR_USE_TRUSTED_CONTEXT を指定した SQLSetConnectAttr を使用します。その後、SQLConnect を発行します。

XA CLI/ODBC アプリケーション
トラステッド接続を要求するには、xa_open ストリング要求で TCTX 属性を true に設定します。

Java アプリケーション
トラステッド接続を要求するには getDB2TrustedPooledConnection または getDB2TrustedXAConnection を使用します。

(認証を必要とする場合も必要としない場合も) 別のユーザー用に接続を再使用するには、アプリケーションで次のような API を指定する必要があります。

CLI/ODBC アプリケーション
接続の切り替え後のユーザー ID とそのユーザーのパスワードを指定するために、SQLSetConnectAttr を使って SQL_ATTR_TRUSTED_CONTEXT_USERID および SQL_ATTR_TRUSTED_CONTEXT_PASSWORD を設定します。

XA CLI/ODBC アプリケーション
接続の切り替え後のユーザー ID とそのユーザーのパスワードを指定するために、SQLSetConnectAttr を使って SQL_ATTR_TRUSTED_CONTEXT_USERID および SQL_ATTR_TRUSTED_CONTEXT_PASSWORD を設定します。

Java アプリケーション
getDB2Connection および reuseDB2Connection を使用します。

フェデレーテッド・トラステッド接続をインプリメントするシナリオ

フェデレーテッド・システムは、それぞれが固有のものです。そのため、フェデレーテッド・トラステッド・コンテキストをインプリメントするためのステップバイステップの指示はありません。これらのシナリオを使用して、フェデレーテッド・トラステッド接続について十分に理解してください。その後、独自のソリューションを計画してインプリメントしてください。

各シナリオは、ユーザー、アプリケーション・サーバー上で実行するアプリケーション、フェデレーテッド・サーバー、およびリモート DB2 データベース・サーバーを含む、同じ複数層フェデレーテッド・システムを使用します。シナリオは、フ

ェデレーテッド・トラステッド接続を使用するようにリモート・ソースおよびフェデレーテッド・サーバーを構成する方法を示しています。

ユーザー・マッピングを必要としない最初のシナリオは、最も簡単にインプリメントおよび保守できます。実際のところ、これが新しいシステムにトラステッド接続をインプリメントするときに使用が推奨されるシナリオです。

2 番目のシナリオは、ユーザー・マッピングを含むシステムにトラステッド接続をインプリメントする方法を例示しています。 3 番目のシナリオは、フェデレーテッド・トラステッド接続を強化してユーザー・マッピングを除去する方法を例示しています。これらのシナリオは最初のシナリオに比べて複雑であり、構成および保守により多くの労力を必要とすることに注意してください。

シナリオ: エンドツーエンドのフェデレーテッド・トラステッド接続 (ユーザー・マッピングなし)

ユーザー・マッピングを使用すると、かなりの管理保守が必要になります。このシナリオでは、フェデレーテッド・システムがユーザー・マッピングを必要としないようにフェデレーテッド・トラステッド接続を構成する方法を示します。

ユーザー ID およびパスワードの要件

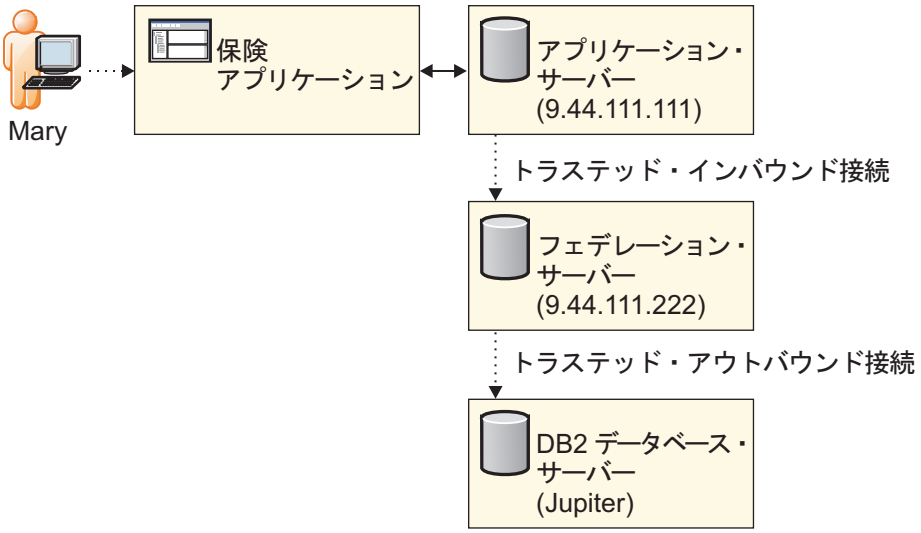
ユーザー・マッピングを使用せずにフェデレーテッド・トラステッド・コンテキストをセットアップするには、ユーザー ID とパスワードが以下の要件を満たしている必要があります。

- 接続の発信元のユーザー ID とパスワードがフェデレーテッド・サーバーで使用可能である必要があります。また、接続の発信元のユーザー ID とパスワードは、フェデレーテッド・サーバー上とデータ・ソース上で同じでなければなりません。資格情報は、接続の発信元が発行する CONNECT ステートメント内、またはアプリケーションが行う API 呼び出しの中で識別されます。CONNECT ステートメントが使用される場合、暗黙的なトラステッド接続が作成されます。この接続は再利用できません。アプリケーションが API 呼び出しを行い、トラステッド接続を明示的に要求する場合、明示的なトラステッド接続が作成されます。この接続は再利用できます。
- 認証なしでの接続の再利用をリモート・トラステッド・コンテキストが許可している場合、接続を再利用する側のユーザー ID は、フェデレーテッド・サーバー上とリモート・データ・ソース上で同じでなければなりません。
- 認証を使用した接続の再利用をリモート・トラステッド・コンテキストが許可している場合、接続を再利用する側のユーザー名およびパスワードは、フェデレーテッド・サーバー上とリモート・データ・ソース上で同じでなければなりません。

シナリオ

以下の図は、エンドツーエンドのフェデレーテッド・トラステッド接続を使用するように構成された典型的な複数層フェデレーテッド・システムを示しています。このシナリオにはアプリケーション・サーバーが含まれていますが、データベース・クライアントがトラステッド接続を確立することも可能です。

接続の発信元: BOSS



このシナリオには、いずれもユーザー・マッピングを必要としない 2 人のユーザーが関係します。

- BOSS は、フェデレーテッド・サーバー上とデータ・ソース上で同一のユーザー ID およびパスワードを持っています。そのため、BOSS はユーザー・マッピングを必要としません。
- Mary は、フェデレーテッド・サーバー上とデータ・ソース上で同一のユーザー ID を持っており、フェデレーテッド・コンテキストによって、認証なしでの接続が許可されています。そのため、Mary はユーザー・マッピングを必要としません。

このシナリオには次の 3 つのサーバーが含まれます。

- アプリケーション・サーバー。これは、保険アプリケーションをホストするもので、IP アドレスは 9.44.111.111 です。
- フェデレーション・サーバー。このサーバーの IP アドレスは 9.44.111.222 です。
- リモート DB2 データベース・サーバー。これは、フェデレーテッド・サーバー上で JUPITER としてカタログされています。

以下のステップは、このシナリオの構成を説明しています。

注: コマンド内で、可変のオブジェクト名はイタリック体で表記されています。トラステッド・コンテキストをインプリメントする際に、ご使用の特定のシステム構成に該当する変数名を指定してください。

1. リモート DB2 データベース・サーバー上で、次のトラステッド・コンテキスト・オブジェクトを作成します。

```
CREATE TRUSTED CONTEXT MY_DB2_TCX
BASED UPON CONNECTION USING
SYSTEM AUTHID BOSS
ATTRIBUTES (ADDRESS '9.44.111.222')
WITH USE FOR PUBLIC WITHOUT AUTHENTICATION
ENABLE
```

このトラステッド・コンテキストは、BOSS がトラステッド接続の発信元であり、接続要求が IP アドレス 9.44.111.222 (フェデレーション・サーバーを識別する) から来る必要があることを指定します。トラステッド・コンテキストは WITH USE FOR PUBLIC WITHOUT AUTHENTICATION を指定します。そのため、トラステッド接続が確立された後、リモート・データ・ソースの正当なユーザーはだれでも、ユーザー ID を提供するだけで接続を再利用できます。

2. フェデレーション・サーバー上で、次のトラステッド・コンテキスト・オブジェクトを作成します。

```
CREATE TRUSTED CONTEXT MY_WFS_TCX
BASED UPON CONNECTION USING
SYSTEM AUTHID BOSS
ATTRIBUTES (ADDRESS '9.44.111.111')
WITH USE FOR PUBLIC WITHOUT AUTHENTICATION
ENABLE
```

このトラステッド・コンテキストは、BOSS がトラステッド接続の発信元であり、接続の要求が IP アドレス 9.44.111.111 (アプリケーション・サーバーを識別する) から来る必要があることを指定します。トラステッド接続が確立された後、フェデレーテッド・サーバーの正当なユーザーはだれでも、ユーザー ID を提供するだけで接続を再利用できます。

3. フェデレーション・サーバー上で、次のサーバー定義を作成します。

```
CREATE SERVER JUPITER TYPE db2/udb
 VERSION 9.5 WRAPPER drda...
 OPTIONS(DBNAME 'remotedb', ...);
```

このサーバー定義には、フェデレーテッド・サーバーが remotedb という名前のリモート DB2 データベースに接続するために必要とする情報が含まれます。

シナリオ、ステップバイステップ

以下は、このシナリオにおいてどのようにトラステッド接続が作成され、ユーザーが切り替えられるかを示す、簡潔なステップバイステップによる説明です。シナリオのコードには、アプリケーションがこれらのタスクを実行する方法を説明するコメントが含まれています。

1. アプリケーション・サーバーが BOSS に関するトラステッド・インバウンド接続を要求します。
2. BOSS がタスクを実行します。すると、フェデレーテッド・サーバーは BOSS について明示的なアウトバウンド・トラステッド接続を確立します。ユーザー ID の BOSS は、アプリケーション・サーバーからフェデレーション・サーバーを介して DB2 データベース・サーバーに伝搬されます。そのサーバーで、BOSS が実行するアクションを監査できます。
3. Mary が自分のラップトップ上の保険アプリケーションにログインします。アプリケーション・サーバーは、フェデレーテッド・サーバーへのインバウンド接続を BOSS から Mary に切り替えます。
4. Mary がアプリケーション内でタスクを実行します。
5. フェデレーション・サーバーはアウトバウンド接続を BOSS から Mary に切り替えます。すると、Mary の ID はフェデレーション・サーバーを介して DB2 サーバーに伝搬されます。そのサーバーで、Mary が実行するアクションを監査できます。

エンドツーエンドのフェデレーテッド・トラステッド接続シナリオのためのサンプル・コード

このサンプル・コードは、エンドツーエンドのフェデレーテッド・トラステッド接続を使用するアプリケーション内で API を使用する方法を示します。

アプリケーションで、明示的なトラステッド・インバウンド接続を要求するため、および接続上でユーザーを切り替えるためには、API を使用する必要があります。このサンプル・コードは、これらのタスクを実行するアプリケーションの一部を例示します。両方のエンドツーエンドのトラステッド・コンテキスト・シナリオで、アプリケーションは同じです。

このアプリケーションからの抜粋では、コマンド行インターフェース API を使用します。 Java または XA ODBC/CLI を使用するアプリケーション用の API も使用可能です。

```
//Set the trusted connection attribute.
SQLSetConnectAttr(h1, SQL_ATTR_USE_TRUSTED_CONTEXT, SQL_TRUE, SQL_IS_INTEGER);

//Establish a trusted inbound connection for BOSS.
SQLConnect(h1, "testdb", SQL_NTS, "BOSS", SQL_NTS,"*****", SQL_NTS);

//Establish a trusted outbound connection for BOSS.
Perform some work under the user ID BOSS.
SQLExecDirect(hstmt, (unsigned char*)"INSERT INTO PATENTS_NN VALUES...", SQL_NTS);
...
//Commit the work.
SQLEndTran(SQL_HANDLE_DBC, h1, SQL_COMMIT);

//At the transaction boundary, switch the inbound user ID on the trusted
connection to Mary.
SQLSetConnectAttr(h1, SQL_ATTR_TRUSTED_CONTEXT_USERID,"Mary", SQL_IS_POINTER);

//Switch the outbound user ID on the trusted connection to Mary. Perform some
work under the user ID Mary.
SQLExecDirect(*hstmt, (unsigned char*)"INSERT INTO PATENTS_NN VALUES...", SQL_NTS)

...
//Commit the work.
SQLEndTran(SQL_HANDLE_DBC, h1, SQL_COMMIT);

//Disconnect from the database.
SQLDisconnect(h1);
```

シナリオ: エンドツーエンドのフェデレーテッド・トラステッド接続 (ユーザー・マッピングあり)

ユーザー・マッピングは、フェデレーテッド・サーバー上およびリモート・データ・ソース上でユーザーが同じユーザー ID およびパスワードを持たない場合に必要となります。このシナリオでは、トラステッド・ユーザー・マッピングを作成して、トラステッド・コンテキストを構成します。

ユーザー・マッピングの要件

フェデレーテッド・サーバーは、インバウンド接続要求を受け取り、リモート・データ・ソースへのアウトバウンド接続要求を出します。ユーザーがフェデレーテッド・サーバー上およびリモート DB2 データベース・サーバー上で同じユーザー ID およびパスワードを持つとき、ユーザー・マッピングは必要ありません。ただし、

ユーザー資格情報が一致しないときは、ユーザー・マッピングが必要です。ユーザー・マッピングは、フェデレーテッド・サーバー上のユーザーのユーザー ID を、リモート・データベース・サーバー上のユーザーのユーザー ID および指定されている場合にはユーザーのパスワードにマップします。

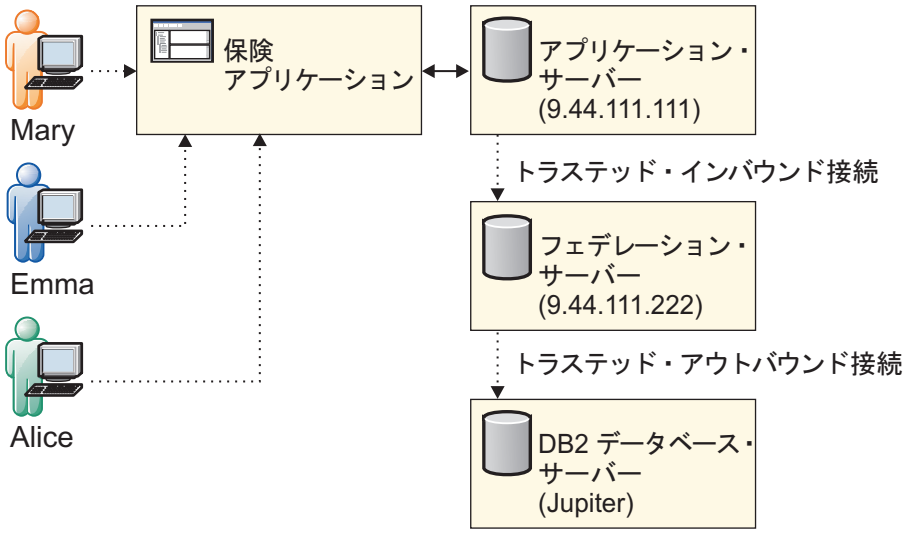
エンドツーエンド・トラステッド・コンテキストを使用するフェデレーテッド・システムで、フェデレーテッド・サーバー上およびリモート DB2 データベース・サーバー上とで名前およびパスワードが一致しないユーザーは、*トラステッド・ユーザー・マッピング* を必要とします。トラステッド・ユーザー・マッピングは、ユーザーにトラステッド・コンテキストを使用する許可があることを指定します。トラステッド・ユーザー・マッピングを作成するか、または既存のユーザー・マッピングを変更するには、 USE_TRUSTED_CONTEXT ユーザー・マッピング・オプションを 'Y' に設定します。

どのユーザーがトラステッド・ユーザー・マッピングを作成および変更できるかは、注意深く制御されます。 SECADM 権限を持つユーザーだけがトラステッド・ユーザー・マッピングを作成またはドロップできます。また、既存のユーザー・マッピングを変更して USE_TRUSTED_CONTEXT ユーザー・マッピング・オプションを追加、設定、またはドロップできるのも SECADM 権限を持つユーザーだけです。トラステッド・ユーザー・マッピングを持つユーザーは、自分のユーザー・マッピングの REMOTE_PASSWORD オプションだけを変更できます。

シナリオ

ユーザー・マッピングを使用する典型的な複数層フェデレーテッド・システムを例示する簡単な図を以下に示します。このシナリオにはアプリケーション・サーバーが含まれていますが、データベース・クライアントがトラステッド接続を確立することも可能です。

接続の発信元: BOSS



このシナリオには、以下の 4 つのユーザーがあります。

- BOSS は接続の発信元であり、フェデレーテッド・サーバー上およびリモート DB2 データベース・サーバー上で同じユーザー ID およびパスワードを持ちます。そのため、BOSS にはユーザー・マッピングがありません。
- Mary にはユーザー・マッピングがありません。彼女はフェデレーション・サーバー上および DB2 データベース・サーバー上で、同じユーザー ID を使用します。トラステッド・コンテキストは PUBLIC が認証なしで接続を再利用できると指定しているので、 Mary のパスワードは必要ありません。
- Alice には、REMOTE_AUTHID オプションを指定するトラステッド・ユーザー・マッピングがあります。 Alice は、フェデレーテッド・サーバー上およびリモート DB2 データベース・サーバー上で異なるユーザー ID を持ちます。トラステッド・コンテキストは誰でも認証なしで接続を再利用できると指定しているので、 Alice のパスワードは必要ありません。
- フェデレーテッド・サーバー上で、Emma はユーザー ID EMMA を使用します。 DB2 データベース・サーバー上で、このユーザー ID はユーザー ID EGREENE およびパスワード MYPASS にマップします。トラステッド・コンテキストは Emma に認証が必要なことを指定しているので、 Emma には REMOTE_AUTHID オプションおよび REMOTE_PASSWORD オプションの両方を指定するトラステッド・ユーザー・マッピングがあります。

このシナリオには次の 3 つのサーバーが含まれます。

- アプリケーション・サーバー。これは、保険アプリケーションをホストするもので、IP アドレスは 9.44.111.111 です。
- フェデレーション・サーバー。このサーバーの IP アドレスは 9.44.111.222 です。
- リモート DB2 データベース・サーバー。これは、フェデレーテッド・サーバー上で JUPITER としてカタログされています。

このシナリオを構成するために、SECADM は以下のステップを完了します。

1. リモート DB2 データベース・サーバー上で、以下のトラステッド・コンテキスト・オブジェクトを作成します。

```
CREATE TRUSTED CONTEXT MY_DB2_TCX
BASED UPON CONNECTION USING
SYSTEM AUTHID BOSS
ATTRIBUTES (ADDRESS '9.44.111.222')
WITH USE FOR EMMA WITH AUTHENTICATION,
PUBLIC WITHOUT AUTHENTICATION
ENABLE
```

   このトラステッド・コンテキストは、 BOSS がトラステッド接続の発信元であり、接続の要求は IP アドレスが 9.44.111.222 のフェデレーテッド・サーバーから出される必要があることを指定します。トラステッド接続が確立した後に、有効なデータベース・ユーザーはユーザー ID を指定するだけで接続を再利用できます。

2. フェデレーテッド・サーバー上で、トラステッド・コンテキスト・オブジェクトを作成します。

```
CREATE TRUSTED CONTEXT MY_WFS_TCX
BASED UPON CONNECTION USING
SYSTEM AUTHID BOSS
```

```
ATTRIBUTES (ADDRESS '9.44.111.111')
WITH USE FOR EMMA WITH AUTHENTICATION,
PUBLIC WITHOUT AUTHENTICATION
ENABLE
```

このトラステッド・コンテキストは、BOSS がトラステッド接続の発信元であり、接続の要求が IP アドレス 9.44.111.111 (アプリケーション・サーバーを識別する) から来る必要があることを指定します。トラステッド接続が確立した後に、 Emma は接続を再利用できますが、認証を受ける必要があります。有効なデータベース・ユーザーは、ユーザー ID を指定するだけで接続を再利用できます。

3. フェデレーション・サーバー上で、次のサーバー定義を作成します。

```
CREATE SERVER JUPITER TYPE db2/udb
VERSION 9.5 WRAPPER drda
OPTIONS(DBNAME 'remotedb', ...);
```

このサーバー定義には、フェデレーテッド・サーバーが remotedb という名前のリモート DB2 データベースに接続するために必要とする情報が含まれます。

4. フェデレーション・サーバー上で、Alice 用にこのトラステッド・ユーザー・マッピングを作成します。

```
CREATE MAPPING FOR USER ALICE
SERVER JUPITER
OPTIONS
(REMOTE_AUTHID 'AJACKSON', USE_TRUSTED_CONTEXT 'Y');
```

このユーザー・マッピングは、トラステッド接続がユーザー ID AJACKSON にマップするユーザー ALICE によって、リモート DB2 データベース・サーバー上で再利用可能であることを指定します。

5. フェデレーション・サーバー上で、Emma 用にこのトラステッド・ユーザー・マッピングを作成します。

```
CREATE MAPPING FOR USER EMMA
SERVER JUPITER
OPTIONS
(REMOTE_AUTHID 'EGREENE', REMOTE_PASSWORD 'MYPASS', USE_TRUSTED_CONTEXT 'Y');
```

このユーザー・マッピングは、トラステッド接続がユーザー ID EGREENE およびパスワード MYPASS にマップするユーザー EMMA によって、リモート DB2 データベース・サーバー上で再利用可能であることを指定します。

シナリオ、ステップバイステップ

1. アプリケーション・サーバーが BOSS に関するトラステッド・インバウンド接続を要求します。
2. BOSS がタスクを実行すると、ユーザー ID BOSS はフェデレーテッド・サーバーを介して DB2 データベース・サーバーに伝搬します。ここで BOSS が実行したアクションを監査できます。
3. Emma はアプリケーション・サーバー上でホスティングされる保険アプリケーションにログインします。アプリケーション・サーバーは、Emma の認証後に、フェデレーテッド・インバウンド接続が BOSS から Emma に切り替わることを要求します。
4. Emma はアプリケーション内でタスクを実行します。

5. フェデレーテッド・サーバーはフェデレーテッド・アウトバウンド接続を BOSS から Emma に切り替えます。ユーザーにマップされた彼女のユーザー ID およびパスワードは、フェデレーテッド・サーバーを介して DB2 サーバーに伝搬されます。ここで EGREENE (Emma のリモート・ユーザー ID) が実行するアクションを監査できます。
6. Alice が保険アプリケーションにログインします。アプリケーション・サーバーは、Alice の認証なしで、フェデレーテッド・インバウンド接続が Emma から Alice に切り替わることを要求します。
7. Alice はアプリケーション内でタスクを実行します。
8. フェデレーテッド・サーバーはフェデレーテッド・アウトバウンド接続を Emma から AJACKSON (Alice のマップされたユーザー ID) に切り替えます。彼女のユーザー ID は、フェデレーテッド・サーバーを介して DB2 データベース・サーバーに伝搬します。ここで AJACKSON が実行するアクションを監査できます。
9. Mary が保険アプリケーションにログインします。 Mary は認証を必要としません。そのためアプリケーション・サーバーは、パスワードを指定しないで、フェデレーテッド・インバウンド・トラステッド接続を Mary に切り替えます。
10. Mary がアプリケーション内でタスクを実行します。
11. フェデレーテッド・サーバーはフェデレーテッド・アウトバウンド接続を AJACKSON から Mary に切り換え、 Mary のユーザー ID は、フェデレーテッド・サーバーを介して DB2 データベース・サーバーに伝搬されます。ここで Mary が実行するアクションを監査できます。

シナリオ: フェデレーテッド・アウトバウンド・トラステッド接続

フェデレーテッド・アウトバウンド・トラステッド接続は、フェデレーテッド・サーバーとリモート DB2 データベース・サーバーとの間にトラステッド環境を確立します。この接続では認証が必要ないので、ユーザー・パスワードを保管および保守する必要がなくなります。

いくつかの構成では、フェデレーション・サーバーは非トラステッドのインバウンド接続を処理する必要があります。以下の記述のいずれかが該当する場合、接続は非トラステッドとなります。

- インバウンド接続要求の属性がフェデレーテッド・サーバー上にあるどのトラステッド・コンテキスト・オブジェクトの属性とも一致しない。
- フェデレーション・サーバーが、接続要求を出したサーバーに対するトラステッド・コンテキストを指定していない。すべてのユーザーは、非トラステッドのインバウンド接続を取得します。

トラステッド・コンテキスト、サーバー定義、およびユーザー・マッピング要件

リモート DB2 データ・ソース・サーバー上にトラステッド・コンテキストを作成するとき、 WITH USE FOR PUBLIC WITHOUT AUTHENTICATION を指定する必要があります。その後、フェデレーテッド・サーバー上および DB2 データベース・サーバー上で同じユーザー ID を使用するすべてのユーザーは、認証なしでアウトバウンド・トラステッド・コンテキストを使用できます。

フェデレーテッド・アウトバウンド・トラステッド接続を作成するには、FED_PROXY_USER オプションをリモート DB2 データ・ソース・サーバーのサーバー定義で指定します。このオプションは、アウトバウンド・トラステッド接続を開始したユーザーの許可 ID を識別します。さらに、フェデレーテッド・プロキシー・ユーザー用のユーザー・マッピングを作成します。データ・ソース上のトラステッド・コンテキストは接続の開始元が認証されることを必要とするので、このユーザー・マッピングには REMOTE_AUTHID および REMOTE_PASSWORD の両方のオプションを指定する必要があります。

SECADM 権限を持つユーザーだけが FED_PROXY_USER オプションの定義されたサーバー定義を作成および変更できます。さらに、SET SERVER OPTION ステートメントは FED_PROXY_USER サーバー・オプションには無効です。

さまざまなプロキシー・ユーザーの間を接続するさまざまなユーザーのセットを構成する必要のある状況が生じることがあります。例えば、異なるユーザー用に異なる役割を構成する場合があります。これを容易にするために、デフォルトのフェデレーテッド・プロキシー・ユーザーを使用しないユーザー・マッピングごとに、FED_PROXY_USER オプションを指定して USE_TRUSTED_CONTEXT オプションを 'Y' に設定します。 FED_PROXY_USER オプションがサーバー定義およびユーザー・マッピングの両方に指定されると、ユーザー・マッピング内の値はサーバー定義内の値をオーバーライドします。

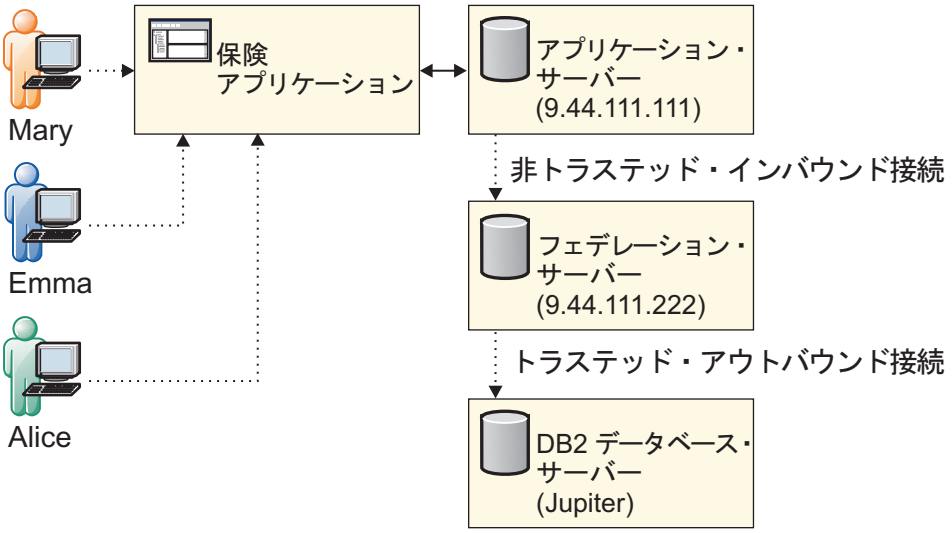
SECADM 権限を持つユーザーだけが FED_PROXY_USER オプションを追加、ドロップ、または設定すること、および FED_PROXY_USER オプションを含むユーザー・マッピングを作成またはドロップすることができます。

重要:
トラステッド・アウトバウンド接続のユーザーがユーザー・マッピングを必要とするのは、ユーザーがフェデレーテッド・サーバーとデータベース・サーバーとで異なるユーザー ID を使用するとき、およびユーザーがデフォルトのフェデレーテッド・プロキシー・ユーザーではないフェデレーテッド・プロキシー・ユーザーを介して接続する必要があるときの 2 つの場合だけです。

シナリオ

次の図は、フェデレーテッド・アウトバウンド・トラステッド接続を必要とするシナリオを示しています。このシナリオにはアプリケーション・サーバーが含まれていますが、データベース・クライアントがトラステッド接続を確立することも可能です。

フェデレーテッド・プロキシー・ユーザー: BOSS および ADM



このシナリオには、以下の 5 つのユーザーがあります。

- BOSS、デフォルトのフェデレーテッド・プロキシー・ユーザーであり、ユーザー・マッピングがあります。
- ADM、フェデレーテッド・プロキシー・ユーザーであり、ユーザー・マッピングがあります。
- Mary、保険アプリケーションを使用するユーザーで、ユーザー・マッピングがありません。
- Emma および Alice、それぞれは保険アプリケーションを使用するユーザーで、ユーザー・マッピングがあります。

このシナリオには次の 3 つのサーバーが含まれます。

- アプリケーション・サーバー。これは、保険アプリケーションをホストするもので、IP アドレスは 9.44.111.111 です。このシナリオにはアプリケーション・サーバーがありますが、任意のクライアントを使用できます。
- フェデレーション・サーバー。このサーバーの IP アドレスは 9.44.111.222 です。
- リモート DB2 データベース・サーバー。これは、フェデレーテッド・サーバー上で JUPITER としてカタログされています。

以下のステップは、構成を説明しています。

注: コマンド内で、可変のオブジェクト名はイタリック体で表記されています。トラステッド・コンテキストをインプリメントする際に、ご使用の特定のシステム構成に該当する変数名を指定してください。

1. リモート DB2 データベース・サーバー上で、次のトラステッド・コンテキスト・オブジェクトを作成します。

```
CREATE TRUSTED CONTEXT MY_DB2_TXC
BASED UPON CONNECTION USING
SYSTEM AUTHID BOSS
ATTRIBUTES (ADDRESS '9.44.111.222')
WITH USE FOR PUBLIC WITHOUT AUTHENTICATION
ENABLE
```

このトラステッド・コンテキストは、 BOSS が接続の発信元であり、アウトバウンド・トラステッド接続の要求は IP アドレスが 9.44.111.222 のフェデレーテッド・サーバーから出される必要があることを指定します。トラステッド・アウトバウンド接続が確立された後、どのユーザーでも認証なしで接続を再利用できます。

```
CREATE TRUSTED CONTEXT MY_DB2_TXC_ALICE
BASED UPON CONNECTION USING
SYSTEM AUTHID ADM
ATTRIBUTES (ADDRESS '9.44.111.222')
WITH USE FOR PUBLIC WITHOUT AUTHENTICATION ROLE Manager
ENABLE
```

このトラステッド・コンテキストは、 ADM が接続の発信元であり、アウトバウンド・トラステッド接続の要求は IP アドレスが 9.44.111.222 のフェデレーテッド・サーバーから出される必要があることを指定します。トラステッド・アウトバウンド接続が確立した後、 Alice は認証なしで接続を再利用できるようになり、トラステッド接続の有効範囲内で Manager の役割を取得します。

2. フェデレーション・サーバー上で、次のサーバー定義を作成します。

```
CREATE SERVER JUPITER TYPE db2/udb
VERSION 9.5 WRAPPER drda...
OPTIONS(DBNAME 'remotedb',FED_PROXY_USER 'BOSS');
```

このサーバー定義には、フェデレーテッド・サーバーが remotedb という名前のリモート DB2 データベースに接続するために必要とする情報が含まれます。この定義は、フェデレーテッド・サーバーへのインバウンド接続は非トラステッドである場合、デフォルトのフェデレーテッド・プロキシー・ユーザーの BOSS が、アウトバウンド・トラステッド接続を確立することを指定します。

3. フェデレーション・サーバー上で、フェデレーテッド・プロキシー・ユーザーのユーザー・マッピングを作成します。

```
CREATE USER MAPPING FOR BOSS SERVER JUPITER
OPTIONS(REMOTE_AUTHID 'BOSS',REMOTE_PASSWORD 'MYPASS');
CREATE USER MAPPING FOR ADM SERVER JUPITER
OPTIONS(REMOTE_AUTHID 'ADM', REMOTE_PASSWORD 'PWD');
```

4. フェデレーテッド・サーバー上で、Emma および Alice 用にこれらのトラステッド・ユーザー・マッピングを作成します。

```
CREATE USER MAPPING FOR EMMA SERVER JUPITER
OPTIONS(REMOTE_ AUTHID 'EGREENE', USE_TRUSTED_CONTEXT 'Y')
CREATE USER MAPPING FOR ALICE SERVER JUPITER
OPTIONS (USE_TRUSTED_CONTEXT 'Y',FED_PROXY_USER 'ADM');
```

両方のユーザー・マッピングで USE_TRUSTED_CONTEXT オプションが Y に設定されているので、ユーザーはトラステッド・コンテキストを使用できます。さらに、Emma および Alice は以下の追加のオプションを必要とします。

- Emma はフェデレーテッド・サーバー上および DB2 データベース・サーバー上で同じユーザー ID を使用しないので、ユーザー・マッピングを必要とします。そのため、Emma のユーザー・マッピングは REMOTE_AUTHID オプションを指定します。
- Alice はトラステッド・コンテキストによって ADM をフェデレーテッド・プロキシー・ユーザーとして使用することが指定されているので、ユーザー・マッピングを必要とします。そのため、Alice のユーザー・マッピングは FED_PROXY_USER オプションを指定します。

このシナリオでは、Mary はユーザー・マッピングを必要としません。 Emma および Alice のユーザー・マッピングはリモート・パスワードを保管しません。そのため、それらのユーザー・マッピングはリモート・パスワードを最新のものに保つために定期的に更新する必要がありません。

シナリオ、ステップバイステップ

以下のステップは、トラステッド・アウトバウンド接続がこのシナリオでどのように機能するかを説明します。

1. アプリケーションは、フェデレーテッド・サーバーに対する非トラステッドのインバウンド接続を作成します。

   ```
   CONNECT TO FEDSVR USER MARY USING '****'
   ```

2. サーバー JUPITER にアクセスする最初のフェデレーテッド要求は、 JUPITER へのアウトバウンド接続を作成します。

   ```
   SELECT * FROM JUPITER_NN01
   CONNECT RESET
   ```

3. JUPITER は FED_PROXY_USER=BOSS を指定していて、 Mary のユーザー・マッピングはフェデレーテッド・プロキシー・ユーザーを指定していないので、 BOSS に対するアウトバウンド接続が作成されてから、すぐに現在のインバウンド・ユーザー (この例では Mary) に切り替わります。
4. Mary のフェデレーテッド要求は完了して、MARY という名前の下で監査ログに記録されます。その後、接続はリセットされます。
5. Emma のための、フェデレーテッド・サーバーに対する非トラステッドのインバウンド接続を作成します。

   ```
   CONNECT TO FEDSVR USER EMMA USING '****'
   ```

6. JUPITER にアクセスする現行接続内の最初のフェデレーテッド要求は、 JUPITER へのフェデレーテッド・アウトバウンド接続を作成します。

   ```
   SELECT * FROM JUPITER_NN01
   CONNECT RESET
   ```

7. アウトバウンド接続が BOSS を使用して作成され、その後すぐに現在のインバウンド・ユーザーであり ID が EGREENE にマップされている Emma に切り替わります。
8. Emma のフェデレーテッド要求は完了して、EGREENE という名前の下で監査ログに記録されます。その後、接続はリセットされます。
9. Alice のための、フェデレーテッド・サーバーに対する非トラステッドのインバウンド接続を作成します。

   ```
   CONNECT TO FEDSVR USER ALICE USING '****'
   ```

10. JUPITER にアクセスする現行接続内の最初のフェデレーテッド要求は、JUPITER へのフェデレーテッド・アウトバウンド接続を作成します。 Alice のトラステッド・ユーザー・マッピングはデフォルトのフェデレーテッド・プロキシー・ユーザー BOSS ではなくフェデレーテッド・プロキシー・ユーザー ADM を使用することを指定しているので、アウトバウンド接続は ADM を使用して作成され、その後すぐに現行インバウンド・ユーザーである Alice に切り替わります。

    ```
    SELECT * FROM JUPITER_NN01
    CONNECT RESET
    ```

11. Alice のフェデレーテッド要求は完了して、監査ログに記録可能になります。その後、接続はリセットされます。

ユーザー・マッピングおよびフェデレーテッド・トラステッド接続

これらの表は、リモート ID およびリモート・パスワードがある場合とない場合のユーザー・マッピングが、エンドツーエンド・フェデレーテッド・トラステッド接続およびアウトバウンド・フェデレーテッド・トラステッド接続でどのように使用されるかを示しています。

以下の表は、接続の発信元である BOSS、および接続の再利用者である Mary について可能なユーザー・マッピングを示しています。表の中で、*非トラステッド接続* の語句はインバウンド接続で非トラステッドの、およびアウトバウンド接続で非トラステッドの、通常の接続を表します。

表のいくつかのセルでは、*使用可能な場合* の語句を使用してパスワードを説明しています。ユーザー ID およびパスワードの両方が接続の一部として明示的に指定されている場合、パスワードはフェデレーテッド・サーバーで使用可能になります。API 接続呼び出しを使用してフェデレーテッド・サーバーに接続するとき、パスワードは明示的に渡されます。例えば CLI/ODBC では、パスワードは SQL_ATTR_TRUSTED_CONTEXT_PASSWORD 接続属性を介して指定されます。CONNECT ステートメントを使用してフェデレーテッド・サーバーに接続する場合、次の構文を使用してパスワードを明示的に渡してください。

```
CONNECT TO database name USER user ID USING password
```

接続の発信元およびフェデレーテッド・プロキシー・ユーザー用のマッピング

表 22. 接続の発信元である BOSS のユーザー・マッピング

BOSS のユーザー・マッピング	**非トラステッド接続**	**エンドツーエンド・フェデレーテッド・トラステッド接続 (ここで BOSS がトラステッド・インバウンド接続を確立する)**	**アウトバウンド・フェデレーテッド・トラステッド接続 (サーバーまたはユーザー・マッピング・オプション FED_PROXY_USER='BOSS' が設定されているとき)**
ユーザー・マッピングなし	BOSS のフェデレーテッド・ユーザー ID およびフェデレーテッド・パスワード (使用可能な場合) をリモート・データ・ソースに渡します。	BOSS のフェデレーテッド・ユーザー ID およびフェデレーテッド・パスワード (使用可能な場合) をリモート・データ・ソースに渡します。	ERROR SQL1101N
リモート・ユーザー ID だけを指定するユーザー・マッピング	BOSS のリモート・ユーザー ID およびフェデレーテッド・パスワード (使用可能な場合) をリモート・データ・ソースに渡します。	BOSS のリモート・ユーザー ID およびフェデレーテッド・パスワード (使用可能な場合) をリモート・データ・ソースに渡します。	ERROR SQL1101N
リモート・ユーザー ID およびリモート・パスワードだけを指定するユーザー・マッピング	BOSS のリモート・ユーザー ID およびリモート・パスワードをリモート・データ・ソースに渡します。	BOSS のリモート・ユーザー ID およびリモート・パスワードをリモート・データ・ソースに渡します。	BOSS のリモート・ユーザー ID およびリモート・パスワードをリモート・データ・ソースに渡します。

接続の再利用者のマッピング

表 23. 接続の再利用者である Mary のユーザー・マッピング

Mary のユーザー・マッピング	**エンドツーエンド・フェデレーテッド・トラステッド接続 (ここで BOSS がトラステッド・インバウンド接続を確立して、接続が Mary に切り替わる)**	**アウトバウンド・フェデレーテッド・トラステッド接続 (サーバーまたはユーザー・マッピング・オプション FED_PROXY_USER='BOSS' が設定されているとき)**
ユーザー・マッピングなし	Mary のフェデレーテッド・ユーザー ID およびフェデレーテッド・パスワード (使用可能な場合) をリモート・データ・ソースに渡します。	Mary のフェデレーテッド・ユーザー ID をリモート・データ・ソースに渡します。
リモート・ユーザー ID だけを指定する非トラステッド・ユーザー・マッピング	Mary のフェデレーテッド・ユーザー ID およびフェデレーテッド・パスワード (使用可能な場合) をリモート・データ・ソースに渡します。	Mary のフェデレーテッド・ユーザー ID をリモート・データ・ソースに渡します。

表 23. 接続の再利用者である Mary のユーザー・マッピング (続き)

Mary のユーザー・マッピング	**エンドツーエンド・フェデレーテッド・トラステッド接続 (ここで BOSS がトラステッド・インバウンド接続を確立して、接続が Mary に切り替わる)**	**アウトバウンド・フェデレーテッド・トラステッド接続 (サーバーまたはユーザー・マッピング・オプション FED_PROXY_USER='BOSS' が設定されているとき)**
リモート・ユーザー ID およびリモート・パスワードだけを指定する非トラステッド・ユーザー・マッピング	Mary のフェデレーテッド・ユーザー ID およびフェデレーテッド・パスワード (使用可能な場合) をリモート・データ・ソースに渡します。	Mary のフェデレーテッド・ユーザー ID をリモート・データ・ソースに渡します。
リモート・ユーザー ID だけを指定するトラステッド・ユーザー・マッピング	Mary のリモート・ユーザー ID をリモート・データ・ソースに渡します。	Mary のリモート・ユーザー ID をリモート・データ・ソースに渡します。
リモート・ユーザー ID およびリモート・パスワードだけを指定するトラステッド・ユーザー・マッピング	Mary のリモート・ユーザー ID およびリモート・パスワードをリモート・データ・ソースに渡します。	Mary のリモート・ユーザー ID およびリモート・パスワードをリモート・データ・ソースに渡します。

第 37 章 ラベル・ベースのアクセス制御 (LBAC) とフェデレーテッド・システム

必ず、適切な権限を持つユーザーのみが表のデータを見ることができるようにしてください。

ラベル・ベースのアクセス制御を使用すると、セキュリティー・ポリシーを表の行と列に適用できます。各セキュリティー・ポリシーは、各ユーザー ID およびセッション ID に付与される証明書を指定します。例えば、表 Prices に列 Wholesale、Retail、および Sale があるとします。ユーザー Alice が列 Retail および Sale にアクセスする資格を持っている場合、照会 SELECT RETAIL、SALE FROM PRICES は正常に行われます。しかし照会 SELECT * WHOLESALE は失敗します。

オブジェクトにニックネームを作成するとき、フェデレーテッド・サーバーはデータ・ソースがラベル・ベースのアクセス制御を使用しているかどうかを自動的に検出します。ラベル・ベースのアクセス制御が使用されている場合、ニックネームはキャッシュされません。ラベル・ベースのアクセス制御が使用可能になる前に作成されたニックネームの場合は、ALTER NICKNAME ステートメントを使用して、キャッシングを許可または不許可にします。例えば、ラベル・ベースのアクセス制御のフェデレーテッド・サポートが使用可能になる前にデータ・ソース・オブジェクトにニックネームを作成した場合、ニックネームを変更して、キャッシングを不許可にすることができます。

各セキュリティー・ポリシーには固有のラベルがあり、これらは表の Label 列に保管されています。データベース管理者は、ラベルが入っている列を非表示にして、ユーザーにその存在を知られないようにすることができます。ラベル列が非表示のニックネームはキャッシュされません。

第 38 章 外部ユーザー・マッピング・リポジトリー

複数のフェデレーテッド・サーバーによって共有される 1 つの外部リポジトリーにさまざまなユーザー・マッピングを保管すれば、各フェデレーテッド・サーバーに保管する場合よりも、ユーザー・マッピングの保守管理作業を削減できます。

保守の点から言えば、ユーザー・マッピングを 1 つの外部リポジトリーに保管する方法は、カタログに保管する方法よりも優れています。リモート・パスワードは頻繁に期限切れになります。ユーザー・マッピングをカタログに保管した場合、リモート・パスワードが期限切れになると、リモート・データ・ソースおよびカタログでパスワードを更新する必要があります。外部リポジトリーを使用すれば、リモート・パスワードが期限切れになったとき、外部リポジトリー内でパスワードを一度更新するだけです。

ユーザー・マッピング用の外部リポジトリーを使用するには、ユーザー・マッピング・プラグインを作成する必要があります。プラグインでは、外部リポジトリーによって使用されるセキュリティー設定と一致するセキュリティー設定を使用する必要があります。フェデレーテッド・サーバーと外部リポジトリーの間の通信を保護するには、Secure Sockets Layer (SSL) を使用します。その後、ユーザー・マッピング・プラグインの構成ファイルを作成するときに、プラグインが SSL を使用することを指定します。さらに、情報を保護するために、プラグインのソース・コードへのアクセスを制限します。

監査を有効にした場合、フェデレーテッド・サーバーがユーザー・マッピング・プラグインを使用するたびに VALIDATE 監査レコードが作成されます。 VALIDATE レコードをキャプチャーするには、db2audit 機能を構成してください。

注: セキュリティーを確保するため、パブリック・ユーザー・マッピングは外部のユーザー・マッピング・リポジトリーからは取得されません。

独自のカスタム・プラグインを作成、テスト、およびデプロイするためのサンプル・プラグインと詳細説明は次のとおりです。

ユーザー・マッピング・プラグイン (C プログラミング言語)

C プラグインは、外部リポジトリーからユーザー・マッピングを取得するためのインターフェースを提供する 5 つの関数で構成されます。

関数呼び出しの順序は次のとおりです。

1. FSUMPluginInit
2. FSUMconnect
3. FSUMfetchUM
4. FSUMdisconnect
5. FSUMPluginTerm

初期化 – FSUMPluginInit

フェデレーテッド・サーバーがプラグイン・ライブラリーをロードした直後に、サーバーは FSUMPluginInit 関数を呼び出します。この関数はプラグインを初期化して、他の関数のポインターをフェデレーテッド・サーバーに渡します。これらの関数はグローバルであり、外部的に解決可能でなければなりません。プラグインが C++ で書かれている場合は、extern "C" を使ってこれらの関数を宣言する必要があります。フェデレーテッド・サーバーは一連のユーティリティー関数へのポインターを渡し、プラグインは必要に応じてこれらを取得できます。

プラグインは、スレッド化された db2fmp プロセスにロードされます。プラグインを使用するすべてのアプリケーションは同じプラグイン・ライブラリーを共有します。プラグイン・ライブラリーはスレッド・セーフでなければなりません。各アプリケーションは、db2fmp プロセス内の 1 つのスレッドを使用します。アプリケーションが正常にユーザー・マッピングを取得してクリーンアップした後、フェデレーテッド・サーバーは将来の使用に備えてスレッドをスレッド・プールに戻します。フェデレーテッド・サーバーは複数のアプリケーションを同時に扱うことができるため、同じプラグイン・ライブラリーを共有する複数のスレッドを同時にアクティブ化することが可能です。各スレッドには、ユーザー・マッピング・リポジトリーへの個別の接続ハンドルがあります。さらに、FSUMPluginInit API は、各スレッドがグローバル・プラグイン・リソースを処理できるようにします (例えば、プラグインの参照カウントの増加)。

リポジトリーへの接続 – FSUMconnect

フェデレーテッド・サーバーは、ユーザー・マッピング・リポジトリーに接続するために FSUMconnect 関数を呼び出します。プラグインには、リポジトリーへの接続に必要なすべての情報を格納する記述子が含まれなければなりません。この情報として、例えば、開いたファイルへのハンドルや、LDAP サーバーへの接続ハンドルが考えられます。また、外部リポジトリーのセキュリティーの実施方法によっては、ユーザー ID とパスワードがこの情報に含まれることもあります。 資格情報が必要な場合には、資格情報の管理方法をセットアップする必要があります。例えば、資格情報を構成ファイルの中に入れた場合、プラグインが外部リポジトリーへの接続を試行するときには、そのファイル内の資格情報がプラグインによって読み取られます。

ユーザー・マッピングの取得 – FSUMfetchUM

プラグインはユーザー・マッピングを外部リポジトリーから取得するために FSUMfetchUM 関数を呼び出し、リモート ID とリモート・パスワードをフェデレーテッド・サーバーに送るために FSUMaddUMOption ユーティリティー関数を呼び出します。リポジトリー内では、フェデレーテッド・サーバー・インスタンス名、データベース名、リモート・サーバー名、およびローカル許可 ID によってそれぞれのユーザー・マッピングが識別されます。さらに、各ユーザー・マッピングには REMOTE_AUTHID および REMOTE_PASSWORD オプションが含まれる必要があります。リモート・パスワードが暗号化されている場合、プラグインはそれをフェデレーテッド・サーバーに送る前に暗号化解除する必要があります。

リポジトリーからの切断 – FSUMdisconnect

フェデレーテッド・サーバーは、ユーザー・マッピング・リポジトリーから切断するために FSUMdisconnect 関数を呼び出します。この関数は、スレッドとユーザー・マッピング・リポジトリーとの関連付けを解除することによって切断します。例えば、この関数は開いているファイルを閉じる操作、または LDAP サーバーとの接続を閉じる操作を行います。

グローバル・リソースの解放 – FSUMPluginTerm

最後の手順として、FSUMPluginInit 関数によって割り振られたグローバル・リソースを解放するために、フェデレーテッド・サーバーは FSUMPluginTerm 関数を呼び出します。この関数は、スレッドとプラグインの関連付けを解除することによって終了処理を行います。

ユーザー・マッピング・プラグインがサポートされるプラットフォーム (C プログラミング言語)

プラグインを作成する前に、サポートされるプラットフォームを使用していることを確認してください。

以下の表は、ユーザー・マッピング・プラグインがサポートされるプラットフォームの一覧です。

プラットフォーム	ファイル名
AIX (64 ビット)	plugin_file_name.a
Linux AMD 64、Power PC (64)、Power PC 390	plugin_file_name.so
Microsoft Windows (32 ビットおよび 64 ビット)	plugin_file_name.dll

ユーザー・マッピング・プラグインを開発する際の制約事項 (C プログラミング言語)

ユーザー・マッピング・プラグインを C で開発する際には、次のような制約事項に注意してください。

C-linkage

C-linkage を使ってプラグイン・ライブラリーをリンクする必要があります。プロトタイプを提供するヘッダー・ファイル、プラグインの実装に必要なデータ構造、およびエラー・コード定義は、C/C++ 用にのみ提供されています。プラグイン・ライブラリーが C++ としてコンパイルされる場合、ロード時に解決される関数は extern "C" で宣言されなければなりません。

.NET 共通言語ランタイムはサポートされない

プラグイン・ライブラリーのソース・コードのコンパイルおよびリンクでは .NET 共通言語ランタイム (CLR) がサポートされていません。

シグナル・ハンドラー

プラグイン・ライブラリーは、シグナル・ハンドラーをインストールしたり、シグナル・マスクを変更してはなりません。このような操作を行った場

合、エラーの報告とエラーからの回復が妨げられます。プラグイン・ライブラリーは C++ 例外を決して発行してはなりません。

スレッド・セーフ
プラグイン・ライブラリーはスレッド・セーフ、再入可能でなければなりません。再入可能であるためには、プラグイン初期化だけが不要です。さまざまなスレッドからプラグイン初期化関数が何度も呼び出される可能性があります。この場合、プラグインはすべての使用済みリソースをクリーンアップして自身を再初期化します。

標準 C ライブラリーおよびオペレーティング・システムの呼び出しのオーバーライド
プラグイン・ライブラリーは、標準 C ライブラリーおよびオペレーティング・システムの呼び出しをオーバーライドしてはなりません。

32 ビットおよび 64 ビット・アプリケーション
32 ビットのフェデレーテッド・サーバーは 32 ビットのプラグインを使用する必要があります。 64 ビットのフェデレーテッド・サーバーは 64 ビットのプラグインを使用する必要があります。クライアントが 32 ビット、サーバーが 64 ビットという混成インスタンスでは、プラグインは 64 ビットでなければなりません。

テキスト・ストリング
入力ストリングは常にヌル終了であるとは限らず、出力ストリングは必ずしもヌル終了である必要はありません。その代わり、すべての入力ストリングに関する整数の長さと、戻される長さに関する整数へのポインターが指定されます。

ユーザー・マッピング・プラグイン用のヘッダー・ファイル fsumplugin.h (C プログラミング言語)

ヘッダー・ファイル fsumplugin.h には、データ構造、関数、およびエラー・コードが格納されます。

ユーザー・マッピング・プラグインは、sqllib/include ディレクトリーにあるこのヘッダー・ファイルを含む必要があります。

```
/* Definition of user mapping option names. */
#define FSUM_REMOTE_AUTHID_OPTION "REMOTE_AUTHID"
#define FSUM_REMOTE_PASSWORD_OPTION "REMOTE_PASSWORD"

/* Definition of option value types. */
#define FSUM_OPTION_VALUE_BINARY_TYPE 1
#define FSUM_OPTION_VALUE_STRING_TYPE 2

/* Data structure to describe an user option. */

typedef struct _FSUMOption
{
   const char* optionName;
   size_t optionNameLen;
   char*  optionValue;
   size_t optionValueLen;
   size_t optionValueType;
   struct _FSUMOption* nextOption;
} FSUMOption;

/* Data structure to describe a user mapping entry. A user mapping entry might
   have multiple user mapping options, such as REMOTE_AUTHID and REMOTE_PASSWORD.
```

```
   These options are placed in a linked-list.  This data structure holds the pointer
   to the first option in the list. */

typedef struct _FSUMEntry
{
   const char* fsInstanceName;
   size_t      fsInstanceNameLen;
   const char* fsDatabaseName;
   size_t      fsDatabaseNameLen;
   const char* fsServerName;
   size_t      fsServerNameLen;
   const char* fsAuthID;
   size_t      fsAuthIDLen;
   FSUMOption* firstOption;
} FSUMEntry;

/* The functions to implement in addition to the FSUMPluginInit function,
   which is not in the FSUMPluginAPIs structure. */

typedef struct _FSUMPluginAPIs
{
   size_t version;
   SQL_API_RC (SQL_API_FN * FSUMconnect)
 (void** a_FSUMRepository,const char* a_cfgFilePath);

   SQL_API_RC (SQL_API_FN * FSUMfetchUM)
 (void* a_FSUMRepository,FSUMEntry* a_entry);

   SQL_API_RC (SQL_API_FN * FSUMdisconnect)
 (void* a_FSUMRepository);

   SQL_API_RC (SQL_API_FN * FSUMPluginTerm) ();
} FSUMPluginAPIs;

/* The federated server provides these utilities as callback functions
to the plug-in. */

typedef SQL_API_RC (SQL_API_FN FSUMallocateFP)
 (size_t a_blkSize, void** a_pblkPtr);

typedef void(SQL_API_FN FSUMdeallocateFP)
 (void* a_blkPtr);

typedef SQL_API_RC (SQL_API_FN FSUMloadFP)
 (const char* a_libName,void** a_lib);

typedef SQL_API_RC (SQL_API_FN FSUMgetFunctionFP)
 (const char* a_functionName,void* a_lib,void** a_pFuncAddress);

typedef SQL_API_RC (SQL_API_FN FSUMunloadFP)
 (void* a_lib);

typedef SQL_API_RC (SQL_API_FN FSUMlogErrorMsgFP)
(sqlint32 a_level, const char* a_msg, size_t a_length);

typedef SQL_API_RC (SQL_API_FN FSUMaddUMOptionFP)
 (FSUMEntry *a_entry,
 const char* optionName,
 size_t optionNameLen,
 const char* optionValue,
 size_t optionValueLen);

/* Structure to hold the utility functions that the federated server provides. */

typedef struct _FSUMPluginUtilities
{
   FSUMallocateFP      *allocate;
```

```
   FSUMdeallocateFP    *deallocate;
   FSUMloadFP          *load;
   FSUMgetFunctionFP   *getFunction;
   FSUMunloadFP        *unload;
   FSUMlogErrorMsgFP   *logErrorMsg;
   FSUMaddUMOptionFP   *addUMOption;
} FSUMPluginUtilities;

/* User mapping plug-in entry point type. */
typedef SQL_API_RC (SQL_API_FN *FSUMPluginInitType)
 (sqlint32, FSUMPluginAPIs*, FSUMPluginUtilities*);

/* User mapping plug-in C interface entry point type. */
typedef SQL_API_RC (*fsum_plugin_hook_type)
 (const char*, FSUMPluginInitType*, FSUMPluginUtilities*);

/* Definition of return codes for utility functions */
#define FSUM_PLUGIN_UTIL_OK 0
#define FSUM_PLUGIN_UTIL_FAILED -1

/* Definition for extern C. */
#define FSUM_PLUGIN_EXT_C extern "C"

/* Error severities that the logErrorMsg function uses.*/
#define FSUM_LOG_NONE      0 /* No logging */
#define FSUM_LOG_CRITICAL  1 /* Severe error encountered */
#define FSUM_LOG_ERROR     2 /* Error encountered */
#define FSUM_LOG_WARNING   3 /* Warning */
#define FSUM_LOG_INFO      4 /* Informational */


/* Error codes that the functions return.*/
#define FSUM_PLUGIN_OK                0
#define FSUM_INITIALIZE_ERROR         1
#define FSUM_PLUGIN_VERSION_ERROR     2
#define FSUM_CONNECTION_ERROR         3
#define FSUM_LOOKUP_ERROR             4
#define FSUM_DECRYPTION_ERROR         5
#define FSUM_DISCONNECT_ERROR         6
#define FSUM_INVALID_PARAMETER_ERROR  7
#define FSUM_UNAUTHORIZED_CALLER      8
#define FSUM_AUTHENTICATION_ERROR     9
#define FSUM_TERMINATION_ERROR        10

/* Maximum length of a name.*/
#define FSUM_MAX_NAME_LEN             128

/* Maximum length of a file path. */
#define FSUM_MAX_PATH_LEN             256

/* Maximum length of an option value. */
#define FSUM_MAX_OPTION_VALUE_LEN     (2048+1)

/* Maximum length of an error message. */
#define FSUM_MAX_ERROR_MSG_SIZE       2048
```

FSUMPluginInit 関数 (C プログラミング言語)

FSUMPluginInit は、ユーザー・マッピング・プラグイン用の初期化関数です。

この関数は次のようなタスクを実行します。

- 必要な他の 4 つの関数のポインターをフェデレーテッド・サーバーに渡す。
- フェデレーテッド・サーバーから渡されるユーティリティー関数を取得する。
- プラグイン・レベルでグローバル・リソースをセットアップする。

構文

```
SQL_API_RC SQL_API_FN FSUMPluginInit
 (sqlint32 a_version,
 FSUMPluginAPIs* a_pluginAPIs,
 FSUMPluginUtilities* a_pluginUtils);
```

入力

sqlint32 a_version
ユーザー・マッピング・プラグイン・インターフェースのバージョン番号。

FSUMPluginUtilities *a_pluginUtils
すべてのユーティリティー関数を含む構造へのポインター。プラグインは、それぞれのユーティリティー関数へのポインターを必要に応じて取得します。

出力

FSUMPluginAPIs *a_pluginAPIs
プラグインはこの構造を使って FSUMconnect、FSUMfetchUM、FSUMdisconnect、および FSUMPluginTerm の関数ポインターをフェデレーテッド・サーバーに渡します。

FSUMconnect 関数 (C プログラミング言語)

外部ユーザー・マッピング・リポジトリーに接続するために、フェデレーテッド・サーバーは FSUMconnect 関数を呼び出します。

構文

```
SQL_API_RC SQL_API_FN FSUMconnect
(void** a_FSUMRepository,
const char* a_cfgFilePath)
```

入力

const char* a_cfgFilePath
プラグイン・ライブラリーの絶対パス。構成ファイルがプラグイン・ライブラリーと同じディレクトリーに存在する場合、プラグインはこのパス情報を使って構成ファイルを見つけることができます。

出力

void a_FSUMRepository**
接続記述子へのポインターは、フェデレーテッド・サーバーに送られる前に void にキャストされます。それ以降の API 関数呼び出しでは、フェデレーテッド・サーバーがポインターをプラグインに渡し、プラグインは接続記述子の実際の構造にポインターをキャストし直す必要があります。

FSUMfetchUM 関数 (C プログラミング言語)

外部リポジトリーからユーザー・マッピング・オプションを取得するために、フェデレーテッド・サーバーは FSUMfetchUM 関数を呼び出します。

それぞれのユーザー・マッピング項目は、フェデレーテッド・インスタンス名 (fsInstanceName)、データベース名 (fsDatabaseName)、リモート・サーバー名 (fsServerName)、およびローカル・ユーザーの許可 ID (fsAuthID) によって識別され

ます。また、REMOTE_AUTHID および REMOTE_PASSWORD オプションがユーザー・マッピングに含まれる場合もあり、これらはプラグインによって取得されます。

構文

```
SQL_API_RC SQL_API_FN FSUMfetchUM (void* a_FSUMRepository,
FSUMEntry* a_entry);
```

入力

void* a_FSUMRepository
外部リポジトリーへの接続ハンドル。このハンドルは、プラグイン内に定義されている実際の構造にキャストされる必要があります。

FSUMEntry* a_entry
このパラメーターは入力パラメーターであり、出力パラメーターでもあります。

入力としては、このパラメーターはユーザー・マッピング・リポジトリー内のユーザー・マッピング項目の識別に必要な情報を渡します。この情報には、fsInstanceName、fsDatabaseName、fsServerName、および fsAuthID が含まれます。

出力としては、ユーザー・マッピング・リポジトリーから取得されるユーザー・マッピング・オプションがこのパラメーターに追加されます。プラグインは FSUMaddUMOption ユーティリティー関数を使ってオプションをユーザー・マッピング項目 a_entry に追加する必要があります。ユーザー・マッピングに REMOTE_PASSWORD オプションが含まれる場合、パスワードが暗号化されていれば、プラグインは FSUMaddUMOption の呼び出し前にパスワードを暗号化解除する必要があります。フェデレーテッド・サーバーは、FSUMaddUMoption 関数に渡されるストリング用にストレージを解放することはありません。

FSUMdisconnect 関数 (C プログラミング言語)

外部ユーザー・マッピング・リポジトリーから切断するために、フェデレーテッド・サーバーはこの関数を呼び出します。

構文

```
SQL_API_RC SQL_API_FN FSUMdisconnect
(void* a_FSUMRepository);
```

入力

void* a_FSUMRepository
外部リポジトリーへの接続ハンドル。このハンドルは、プラグイン内に定義されている実際の構造にキャストされる必要があります。

FSUMPluginTerm 関数 (C プログラミング言語)

ユーザー・マッピング・プラグインは、FSUMPluginInit 関数によって割り振られたグローバル・リソースを解放するためにこの関数を呼び出します。

この関数にはパラメーターはありません。

```
SQL_API_RC SQL_API_FN FSUMPluginTerm ()
```

ユーザー・マッピング・プラグインの開発 (C プログラミング言語)

外部リポジトリーに接続し、ユーザー・マッピングを取得して、リモート・パスワードを暗号化解除するようなプラグインを開発する必要があります。使用するリポジトリーに応じて、プラグインのコーディング方法が異なります。

このタスクについて

重要: プラグインを開発して使用する際には、機密のユーザー ID とパスワードを複数のソース間で送受信することに注意してください。このような情報を保護するには、プラグイン・ソース・コードへのアクセスを制限します。さらに、フェデレーテッド・サーバーがプラグインを使用するたびに診断ログ・ファイル内に VALIDATE レコードをキャプチャーするよう DB2 監査機能を構成します。診断ログ・ファイルは、使用状況を追跡するだけでなく、問題が発生した場合のトラブルシューティングにも役立ちます。

ユーザー・マッピング・プラグインを開発するには、これらのタスクを完了してください。

ユーザー・マッピング・プラグイン用の外部関数の宣言 (C プログラミング言語)

プラグインは関数をインプリメントします。 FSUMPluginInit 関数が呼び出されるとき、関数ポインターがフェデレーテッド・サーバーに渡されます。

この正確な関数名を必要とする関数は、初期化関数 FSUMPluginInit だけです。その他のそれぞれの関数に対しては、任意の有効な C 関数名を使用できます。これらの関数はグローバルであり、外部的に解決される必要があります。 C++ でプラグインを作成する場合は、FSUM_PLUGIN_EXT_C を使って関数を宣言する必要があります。

サンプル・プラグインは以下の API をインプリメントします。

- FSUMconnect API (myConnect 関数の中)
- FSUMfetchUM (myFetchUM 関数の中)
- FSUMdisconnect (myDisconnect 関数の中)
- FSUMPluginTerm (myPluginTerm 関数の中)

サンプルでは、FSUMPluginInit 関数が呼び出されるとき、myConnect、myFetchUM、myDisconnect、および myPluginTerm の関数ポインターがフェデレーテッド・サーバーに渡されます。

`fsumplugin_file.c` ファイル内には次のようなコードがあります。

```
SQL_API_RC SQL_API_FN FSUMPluginInit
(sqlint23 version,
FSUMPluginAPIs *pluginAPIs,
FSUMPluginUtilities* pluginUtils)

SQL_API_RC SQL_API_FN myConnect
(void** FSUMRepository)

SQL_API_RC SQL_API_FN myFetchUM
(void* FSUMRepository,
```

```
FSUMEntry* entry)

SQL_API_RC SQL_API_FN myDisconnect
(void* FSUMRepository)

SQL_API_RC SQL_API_FN myPluginTerm ()
```

ユーザー・マッピング・プラグイン用のユーティリティー関数の宣言 (C プログラミング言語)

ユーザー・マッピング・プラグイン用に 4 つの必須ユーティリティー・オプションを宣言する必要があります。

FSUMlogErrorMSg、FSUMaddUMOption、FSUMallocate、および FSUMdeallocate はユーザー・マッピング・プラグイン用の必須のユーティリティー関数です。ユーザー・マッピング・プラグインを開発するときには、オプションのユーティリティー関数が役立ちます。ユーティリティー関数のポインターは FSUMPluginInit 関数内の FSUMPluginUtilities 構造から取得されます。

必須のユーティリティー関数

```
FSUMlogErrorMsgFP *FSUMlogErrorMsg=NULL;
FSUMaddUMOptionFP *FSUMaddUMOption=NULL;
FSUMallocateFP *FSUMallocate=NULL;
FSUMdeallocateFP *FSUMdeallocate=NULL;
```

オプションのユーティリティー関数

```
FSUMgetFunctionFP *FSUMgetFunction=NULL;
FSUMloadFP *FSUMload=NULL;
FSUMunloadFP *FSUMunload=NULL;
```

ユーザー・マッピング・プラグイン用の関数ポインターの設定 (C プログラミング言語)

必要な関数のポインターをフェデレーテッド・サーバーに渡し、ユーティリティー関数のポインターをフェデレーテッド・サーバーから取得します。

```
/* The plug-in initialization function. Do not change the name. */

SQL_API_RC SQL_API_FN FSUMPluginInit
 (sqlint32 a_version,
 FSUMPluginAPIs* a_pluginAPIs,
 FSUMPluginUtilities* a_pluginUtils)
{
   SQL_API_RC rc = FSUM_PLUGIN_OK ;

   /* Pass the function pointers to federated server */

   a_pluginAPIs->FSUMconnect = &myConnect
   a_pluginAPIs->FSUMfetchUM = &myFetchUM
   a_pluginAPIs->FSUMdisconnect = &myDisconnect
   a_pluginAPIs->FSUMPluginTerm = &myPluginTerm

   /* Get the pointers of the required utility functions */

   FSUMallocate = a_pluginUtils->allocate;
   FSUMdeallocate = a_pluginUtils->deallocate;
   FSUMlogErrorMsg = a_pluginUtils->logErrorMsg;
   FSUMaddUMOption = a_pluginUtils->addUMOption;
   /*You can also get the pointers of the optional utility functions */

   /** If there is an error, do the following:
```

```
    1. Optional --- Call FSUMlogErrorMsg to log the error
    2. Mandatory --- Return error code rc = FSUM_INITIALIZE_ERROR
  **/

  return rc;
}
```

ユーザー・マッピング・プラグインでのエラー処理 (C プログラミング言語)

ユーザー・マッピング・プラグインは、FSUMlogErrorMsg 関数を使ってすべてのエラーおよび情報メッセージを db2diag.log ファイルに記録します。

関数呼び出しが成功した場合、プラグインは FSUM_PLUGIN_OK を戻します。呼び出しが成功しなかった場合、プラグインはエラー・コードを戻します。以下の表は、エラーについて説明しています。

表 24. ユーザー・マッピング・プラグインのエラー

エラー番号	定数名	エラー・メッセージ	エラーを戻す関数	エラー・コード
0	FSUM_PLUGIN_OK	プラグインが正常に実行されました。	• FSUMPluginInit • FSUMconnect • FSUMfetchUM • FSUMdisconnect • FSUMPluginTerm	SQL_SUCCESS (エラーなし)
1	FSUM_INITIALIZE_ERROR	プラグインは初期化に失敗しました。	• FUMPluginInit	SQL20349N、理由コード 1
2	FSUM_PLUGIN_VERSION_ ERROR	プラグインのバージョンが正しくありません。	• FUMPluginInit	SQL20349N、理由コード 2
3	FSUM_CONNECTION_ ERROR	外部リポジトリーに接続できません。	• FSUMPluginInit	SQL20349N、理由コード 3
4	FSUM_LOOKUP_ERROR	リポジトリーの検索に失敗しました。	• FSUMfetchUM	SQL20349N、理由コード 4
5	FSUM_DECRYPTION_ERROR	暗号化解除は失敗しました。	• FSUMfetchUM	SQL20349N、理由コード 5
6	FSUM_DISCONNECT_ERROR	リポジトリーから切断できません。	• FSUMdisconnect	SQL20349N、理由コード 6
7	FSUM_INVALID_ PARAMETER_ERROR	パラメーターが無効です。	• FSUMfetchUM • FSUMdisconnect • FSUMPluginTerm	SQL20349N、理由コード 7
8	FSUM_UNAUTHORIZED_ CALLER	呼び出し元にプラグインの呼び出し権限がありません。	• FSUMPluginInit	SQL20349N、理由コード 8
9	FSUM_AUTHENTICATION_ ERROR	リポジトリーを認証できません。	• FSUMconnect	SQL20350N、理由コードなし

表 24. ユーザー・マッピング・プラグインのエラー (続き)

エラー番号	定数名	エラー・メッセージ	エラーを戻す関数	エラー・コード
10	FSUM_TERMINATION_ERROR	プラグイン・レベルでのリソースのクリーンアップに失敗しました。	• FSUMPluginTerm	SQL20349N、理由コード 9
0 から 10 の数値以外		不明なエラーが発生しました。	• FSUMPluginInit • FSUMconnect • FSUMfetchUM • FSUMdisconnect • FSUMPluginTerm	SQL20349N、理由コード 10

また、プラグインで FSUMlogErrorMsg ユーティリティー関数を使ってログ・メッセージを db2diag.log ファイルに記録することもできます。メッセージの重大度を指定するには、
FSUM_LOG_CRITICAL、FSUM_LOG_ERROR、FSUM_LOG_WARNING、または FSUM_LOG_INFO を使用してください。

テスト・プログラムを使ってプラグインを検査する際には、メッセージは画面に直接出力されます。

ユーザー・マッピング・プラグインの作成 (C)

ユーザー・マッピング・プラグインを作成するには bldplugin スクリプトを使用します。

このタスクについて

次のようにして、ユーザー・マッピング・プラグインを作成します。

手順

1. bldplugin スクリプトを調べて、特定のインストール場所に一致するようパス情報を更新します。
2. 次のコマンドを発行して、プラグインを作成します。

   ```
   bldplugin plugin_file_name
   ```

   詳細については、bldplugin ファイルをご覧ください。
3. ユーザー・マッピング情報を格納して暗号化方式を実施するための外部リポジトリーを設定します。 サンプル・ユーザー・マッピング・プラグインを作成する場合には、fsumplugin_file.txt ファイルを開いて、実際のシステム用の新しいユーザー・マッピング項目を作成してください。サンプル・プラグインは、リモート・パスワードのバイトを逆にする簡単な暗号化方式を使用します。

ユーザー・マッピング・プラグインのテスト (C プログラミング言語)

サンプル・ユーザー・マッピング・プラグインまたは独自のカスタム・ユーザー・マッピング・プラグインを、フェデレーテッド・サーバーにデプロイする前にロードして検査するには、テスト・プログラムを使用します。

このタスクについて

セットアップ・プログラム fsumsetup_file (Microsoft Windows では fsumsetup_file.exe) およびテスト・プログラム fsumlookup (Windows では fsumlookup.exe) は、サンプル・プラグイン・ファイルのインストール場所と同じディレクトリーにインストールされます。

fsumsetup_file プログラムは、fsumplugin_file.cfg という構成ファイルを生成します。このファイルは、ユーザー・マッピング項目が入っているテキスト・ファイルへの絶対パスを格納します。 fsumlookup プログラム (Microsoft Windows では fsumlookup.exe) はプラグインを検査します。

手順

1. ユーザー・マッピングを格納するファイルの絶対パスとファイル名を設定するには、fsumsetup_file プログラムを実行します。
2. プラグインを検査するには、fsumlookup プログラムを次のように実行します。

   ```
   fsumlookup pluginName fsInstance fsDatabase fsRemoteServer fsAuthid
   ```

   ここで、

   - *pluginName* はプラグインの名前。
   - *fsInstance* はフェデレーテッド・サーバー・インスタンスの名前。
   - *fsDatabase* はフェデレーテッド・データベースの名前。
   - *fsRemoteServer* は、CREATE SERVER ステートメントで指定されたリモート・データ・ソース・サーバーの名前。
   - *fsAuthid* はローカル・ユーザー ID。

   プラグインが正常に機能している場合、ユーザー・マッピング項目を識別するパラメーター、およびオプション名とオプション値が fsumlookup プログラムによって画面に出力されます。出力は、例えば以下のようになります。

   ```
   fsumlookup fsumplugin_file.a FSinst1 FSdb remoteDB localID
   fsInstanceName: FSinst1
   fsDatabaseName: FSdb
   fsServerName: remoteDB
   fsAuthID: localID
   optionName:REMOTE_PASSWORD
   optionValue:p4ssw0rd
   optionName:REMOTE_AUTHID
   optionValue:remoteID
   ```

   エラーが発生した場合、テスト・プログラムはエラー・メッセージを画面に出力します。

ユーザー・マッピング・プラグインのデプロイ (C プログラミング言語)

任意のディレクトリーでユーザー・マッピング・プラグインを作成してコンパイルすることができます。その後、プラグインをデプロイするには、フェデレーテッド・サーバー上の適切なディレクトリーにそれをコピーします。

手順

プラグインをフェデレーテッド・サーバーにデプロイするには、フェデレーテッド・サーバー上の以下の適切なディレクトリーにプラグインおよび構成ファイルをコピーします。

- UNIX および Linux では *inst_home*/sqllib/function (ただし *inst_home* はインスタンスのホーム・ディレクトリー)
- Windows では *%DB2PATH%*¥function (ただし *%DB2PATH%* は DB2 データベース・システムがインストールされているディレクトリー)

外部ユーザー・マッピング・リポジトリーへのアクセスの有効化

ユーザー・マッピング・プラグインを作成した後、外部リポジトリーに保管されたユーザー・マッピングにフェデレーテッド・サーバーがアクセスできるようにするために、DB2_UM_PLUGIN および DB2_UM_PLUGIN_LANG オプションを設定する必要があります。

このタスクについて

これらのオプションはラッパー・レベルまたはサーバー・レベルのどちらでも設定できますが、両方のオプションを同じレベルで設定する必要があります。
DB2_UM_PLUGIN はプラグインの名前を指定し、DB2_UM_PLUGIN_LANG はプラグインが作成された言語を指定します。 2 つのオプションは互いに依存しています。このため、DB2_UM_PLUGIN_LANG を追加する前に DB2_UM_PLUGIN を追加する必要があります。両方のオプションを同じステートメント内で追加する場合には、指定する順序は任意です。これらのオプションをドロップする場合には、DB2_UM_PLUGIN をドロップする前に DB2_UM_PLUGIN_LANG をドロップする必要があります。

例えば、以下のステートメントでは、サンプル・プラグインの名前を使ってラッパー w1 を作成し、ラッパー w2 を変更し、サーバー s1 を作成して、ユーザー・マッピング・プラグインを使用するようサーバー s2 を変更します。

```
CREATE WRAPPER w1 LIBRARY 'libdb2drda.a'
 OPTIONS
 (DB2_UM_PLUGIN 'fsumplugin_file.a', DB2_UM_PLUGIN_LANG 'C');

ALTER WRAPPER w2
 OPTIONS
 (ADD DB2_UM_PLUGIN 'fsumplugin_file.a', ADD DB2_UM_PLUGIN_LANG 'C');

CREATE SERVER s1 TYPE db2/cs VERSION 10 WRAPPER w2
 AUTHORIZATION remoteID PASSWORD p4ssw0rd
 OPTIONS
 (NODE 'node1',
 DBNAME 'db1',
 DB2_UM_PLUGIN 'fsumplugin_file.a',
 DB2_UM_PLUGIN_LANG 'C');

ALTER SERVER s2
 OPTIONS
 (ADD DB2_UM_PLUGIN 'fsumplugin_file.a', ADD DB2_UM_PLUGIN_LANG 'C');
```

ユーザー・マッピング・プラグインの更新 (C プログラミング言語)

既存のユーザー・マッピング・プラグインを変更した後、この更新バージョンをフェデレーテッド・サーバーにコピーする必要があります。

このタスクについて

次のようにして、ユーザー・マッピング・プラグインを更新します。

手順

1. 更新されたプラグインを sqllib/function ディレクトリーにコピーします。
2. フェデレーテッド・サーバー・インスタンスを停止して再始動するために、次の一連のコマンドを入力します。

```
db2stop
db2start
```

ユーザー・マッピング・プラグインのサンプル (C プログラミング言語)

サンプル・ソース・コードを調べれば、外部ユーザー・マッピング・リポジトリーへの C インターフェースを提供するプラグインの開発方法を学習できます。

このユーザー・マッピング・プラグインのサンプルは、外部リポジトリーとしてファイルを使用します。サンプル・ソース・ファイルは、システムに自動的にインストールされます。インストール場所は、ご使用のオペレーティング・システムによって次のように異なります。

- Windows では *%DB2PATH%*¥samples¥federated¥umplugin¥file (ただし *%DB2PATH%* は DB2 データベース・システムがインストールされているディレクトリー)
- Linux および UNIX では *inst_home*/sqllib/samples/federated/umplugin/file (ただし *inst_home* はインスタンスのホーム・ディレクトリー)

サンプルには以下のファイルが含まれています。

fsumplugin_file.h
サンプル・プラグイン用のデータ構造と関数定義が入っているヘッダー・ファイル。

fsumplugin_file.c
サンプル・プラグインのソース・コード。外部リポジトリーとしてファイルを使用します。

fsumplugin_file.txt
プレーン・テキスト形式のサンプル・ユーザー・マッピング項目が入っているファイル。

fsumlookup
フェデレーテッド・サーバー環境の外側でサンプル・プラグインをテストする独立型プログラム。 Microsoft Windows では、ファイル名は fsumlookup.exe です。

fsumsetup_file
ユーザー・マッピングを格納するファイルの絶対パスとファイル名を設定する独立型プログラム。 Microsoft Windows では、ファイル名は fsumsetup_file.exe です。

bldplugin
プラグインをコンパイルおよびリンクするビルド・スクリプト。 Microsoft Windows では、ファイル名は `bldplugin.bat` です。

README サンプル・プラグインのコンパイル、構築、テスト、デプロイに関する説明が入っているファイル。

ファイル `fsumplugin_file.txt` では、各ユーザー・マッピング項目が 1 行からなり、バイトを逆にするという単純な方法でパスワードが暗号化されています。これは非常に簡単な暗号化方式です。実稼働環境で使用するためにサンプル・プラグインを変更する場合には、実際の環境のセキュリティー要件に適合する暗号化方式を使用する必要があります。

以下の構文は、サンプル・ユーザー・マッピング項目の形式を示しています。

注: ここでは複数行にわたってコードが示されていますが、アプリケーションでは、このコード全体を 1 行で入力する必要があります。セミコロンはユーザー・マッピングの各部分を分割し、コロンは REMOTE_AUTHID および REMOTE_PASSWORD オプション名と、それらに対応する設定値とを分割しています。

```
fsInstance;fsDatabase;fsRemoteServer;
fsAuthid;REMOTE_AUTHID:remote_authID;
REMOTE_PASSWORD:remote_password
```

例:

```
DB2INST1;TESTDB;MSSQL2K;ALICE;REMOTE_AUTHID:TESTUSR1;REMOTE_PASSWORD:drowssap
```

ユーザー・マッピング・プラグイン (Java プログラミング言語)

Java プラグインのアーキテクチャーには、2 つのインターフェース・クラスと 3 つのユーティリティー・クラスが含まれています。これらを使用して、外部リポジトリーからユーザー・マッピングを取得するプラグインを開発します。

データ・ソースのユーザー・マッピングはデフォルトで、各フェデレーテッド・サーバーのローカルに保管されます。 LDAP サーバーはユーザー・マッピング項目などのオブジェクトをディレクトリー・ツリーに保管します。これらのオブジェクトにはパスワードなどの属性があります。さらに LDAP サーバーは、ユーザーの E メール・アドレスや電話番号など、ユーザーに関するその他の情報を保管することもよくあります。

次の図は、サンプル・ユーザー・マッピング・プラグインのアーキテクチャーを示しています。

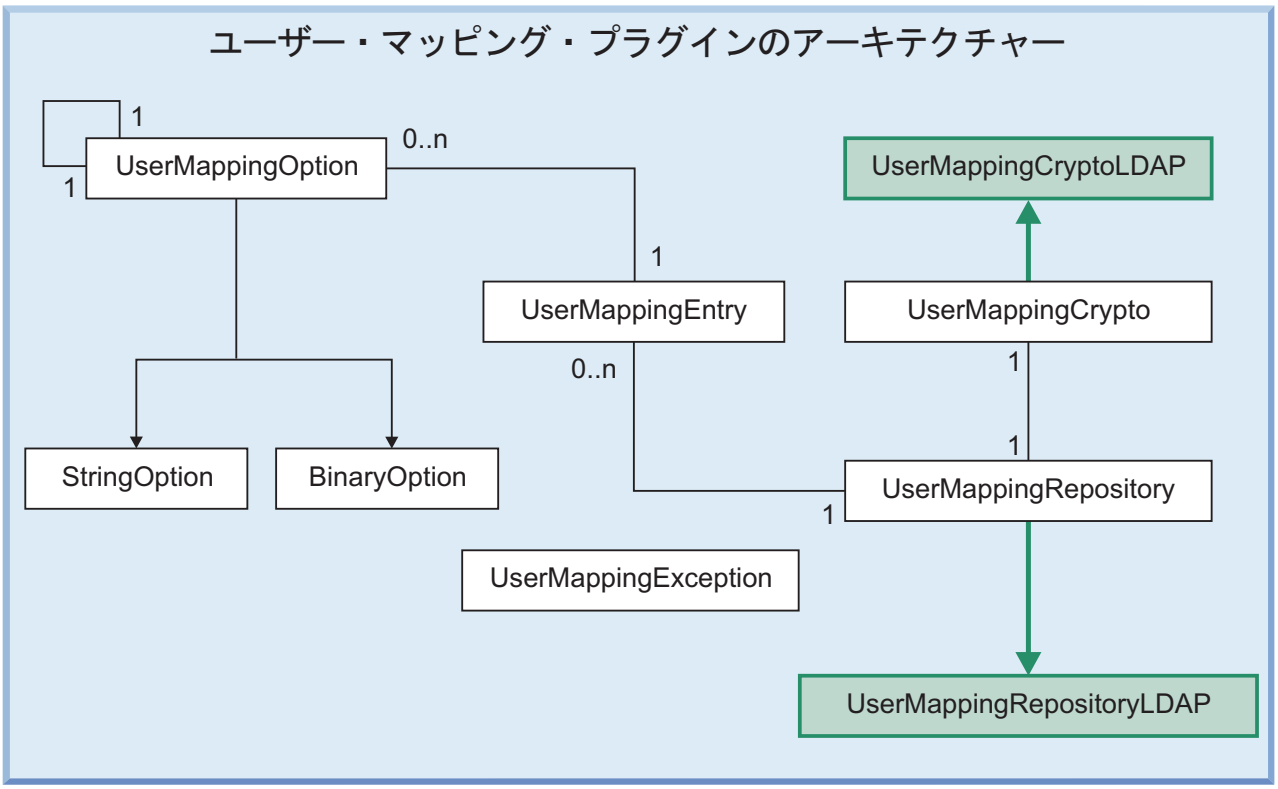


図 15. ユーザー・マッピング・プラグインのアーキテクチャー

UserMappingCrypto と UserMappingRepository は、インターフェース・クラスです。組織が使用する外部リポジトリーからユーザー・マッピングを取り出すユーザー・マッピング・プラグインを作成するには、これらのインターフェース・クラスを継承する必要があります。 UserMappingEntry、UserMappingOption、および UserMappingException はユーティリティー・クラスです。ユーティリティー・クラスは変更なしで使用できます。インターフェース・クラスの関数およびメソッドの中には、ユーティリティー関数として機能するものもあります。例えば、UserMappingCrypto クラスの getChars() 関数と getBytes() 関数は変更なしで使用できます。

図の中で *0..n* とあるのは、そのオブジェクトが任意の数 (ゼロ以上) 存在し得ることを意味します。例えば UserMappingEntry オブジェクトは、UserMappingOption オブジェクトについては複数所有することができますが、UserMappingRepository オブジェクトは 1 つしか所有できません。 UserMappingCryptoLDAP クラスと UserMappingRepositoryLDAP クラスは、その親クラスから継承されたものです。

ユーザー・マッピング・プラグイン用のクラス (Java プログラミング言語)

Java プラグインのアーキテクチャーには、2 つのインターフェース・クラスと 3 つのユーティリティー・クラスが含まれています。これらを使用して、外部リポジトリーからユーザー・マッピングを取得するプラグインを開発します。

UserMappingRepository クラス (Java プログラミング言語)

UserMappingRepository クラスは、コンストラクターを持たない抽象クラスです。独自のユーザー・マッピング・プラグインを作成するには、UserMappingRepository クラスのサブクラスを作成するか、またはサンプル Java プラグインのサブクラスに変更を加える必要があります。

UserMappingRepository クラスには、getVersionNumber()、getCrypto()、connect()、disconnect()、fetchUM()、および lookupUM() という共通メソッドが含まれています。これらの関数を含む UserMappingRepository の独自のサブクラスを作成する必要があります。これらの関数に、プラグインが外部リポジトリーとの対話に使用するコードを作成します。

LDAP サーバーからユーザー・マッピングを取り出すサンプル Java プラグインで、これらの関数のインプリメンテーションを参照できます。このファイルは sqllib/samples/federated/umplugin/ldap/ ディレクトリーにあります。このクラスの関数は、UserMappingRepositoryLDAP.java および UserMappingLookupLDAP.java ファイルで使用されます。

共通メソッド

int getVersionNumber()
プラグインによって使用されるプラグイン開発キットのバージョン番号を戻します。

UserMappingCrypto getCrypto()
この UserMappingRepository オブジェクトに関連付けられている UserMappingCrypto オブジェクトを戻します。

abstract void connect()
リポジトリーに接続するための独自のメソッドをこの関数にインプリメントする必要があります。

abstract void disconnect()
リポジトリーから切断するための独自のメソッドをこの関数にインプリメントする必要があります。

abstract voidfetchUM(UserMappingEntry **um)**
リポジトリーからユーザー・マッピングを取り出すための独自のメソッドをこの関数にインプリメントする必要があります。 **um** パラメーターには、取り出すユーザー・マッピングを判別するために使用される詳細な照会情報が含まれます。

UserMappingEntrylookupUM(UserMappingRepository repository, String **iiInstanceName,** String **iiDatabaseName,** String **iiRemoteServerName,** String **iiAuthid)**
この関数は主に、プラグインのテストで使用されます。この関数は入力として **iiInstanceName**、**iiDatabaseName**、**iiRemoteServerName**、および **iiAuthid** パラメーターを使用して、UserMappingEntry クラスを作成し、初期化します。この関数は、connect、fetchUM、および disconnect メソッドを呼び出します。

UserMappingCrypto クラス (Java プログラミング言語)

外部リポジトリーがリモート・パスワードを暗号化またはエンコードする場合、UserMappingCrypto クラスの独自のサブクラスを作成する必要があります。作成する

サブクラスのコンストラクターは、暗号化オブジェクトの構成に使用されます。暗号化クラスのメソッドは、ユーザー・マッピング・パスワードの暗号化、暗号化解除、エンコード、またはデコードが必要なときに他のクラスによって呼び出されます。

UserMappingCrypto クラスには、encrypt()、decrypt()、encode()、および decode() という共通メソッドが含まれています。これらの関数の中に、リモート・パスワードを暗号化、暗号化解除、エンコード、およびデコードするコードを作成する必要があります。 getBytes() 関数と getChars() 関数は継承されるユーティリティー関数で、変更なしで使用できます。コーディングする暗号化、暗号化解除、エンコード、デコードのメソッドは、保管されたパスワードを保護するために外部リポジトリーが使用する暗号化およびエンコード・メソッドと一致しなければなりません。

LDAP サーバーからユーザー・マッピングを取り出すサンプル Java プラグインで、これらの関数のインプリメンテーションを参照できます。このファイルは sqllib/samples/federated/umplugin/ldap/ ディレクトリーにあります。このクラスの関数は、UserMappingRepositoryLDAP.java および UserMappingSetupLDAP.java サンプル・ファイルで使用されます。

共通メソッド

abstract byte[] encrypt(byte[] **plainValue)**
外部リポジトリーが使用する暗号化アルゴリズムと一致する暗号化アルゴリズムをインプリメントします。

abstract byte[] decrypt(byte[] **encryptedValue)**
外部リポジトリーが使用する暗号化アルゴリズムを解除してパスワードを戻す暗号化解除アルゴリズムをインプリメントします。

abstract string encode(byte[] **bytes)**
bytes パラメーターをストリングにエンコードする関数を作成またはインプリメントします。この関数は暗号化されたバイトの値をストリングにエンコードします。

abstract byte[] decode(String[] **string)**
string パラメーターをバイトにデコードする関数を作成またはインプリメントします。この関数は、値を暗号化解除できるよう、取り出されたストリングのパスワードをバイトにデコードします。

byte[] getBytes(char[] **chars)**
この関数は継承され、変更なしで使用できます。この関数はストリングの各文字をバイトにトランスフォームします。

char[] getChars(byte[] **bytes)**
この関数は継承され、変更なしで使用できます。この関数は各バイトを文字にトランスフォームします。

保護属性

SecretKey キー
リモート・パスワードの暗号化および暗号化解除に使用される秘密鍵。

Cipher 暗号
秘密鍵がパスワードの暗号化に使用するアルゴリズム。

UserMappingEntry クラス (Java プログラミング言語)

UserMappingEntry クラスは、ユーザー・マッピング・オプションを作成し、保持するユーティリティー・クラスです。 UserMappingEntry クラスのメソッドは、UserMappingRepository クラスの fetchUM() および lookupUM() 関数によって呼び出されます。

共通メソッド

LDAP サーバーからユーザー・マッピングを取り出すサンプル Java プラグインで、これらの関数の使用法を参照できます。このファイルは sqllib/samples/federated/umplugin/ldap/ ディレクトリーにあります。このクラスの関数は、UserMappingRepositoryLDAP.java および UserMappingLookupLDAP.java ファイルで使用されます。

UserMappingEntry(UserMappingRepository **repository,** String **iiInstanceName,** String **iiDatabaseName,** String **iiRemoteServerName,** String **iiAuthID**)
このコンストラクターは UserMappingEntry オブジェクトをインスタンス化するために使用されますが、その際には次の入力パラメーターを使用します。

- **iiInstance** - フェデレーテッド・サーバーのインスタンス名
- **iiDatabase** - フェデレーテッド・サーバー上のデータベース名
- **iiRemoteServerName** - データ・ソースのリモート・サーバー名
- **iiAuthid** - ユーザー・マッピングに関連付けられているローカル・ユーザー ID

UserMappingRepository getRepository()
この UserMappingEntry に関連付けられている UserMappingRepository オブジェクトを戻します。

String getiiInstanceName()
インスタンスの名前を戻します。

String getiiDatabaseName()
データベースの名前を戻します。

String getiiRemoteServerName()
リモート・サーバーの名前を戻します。

String getiiAuthID()
ユーザー・マッピングに関連付けられているローカル・ユーザー ID の名前を戻します。

UserMappingOption getFirstOption()
この UserMappingEntry オブジェクトに属する最初の UserMappingOption オブジェクトを戻します。

void addOption(UserMappingOption **newOption)**
この UserMappingEntry オブジェクトに新規の UserMappingOption オブジェクトを追加します。

UserMappingOption クラス (Java プログラミング言語)

UserMappingOption クラスは、フェデレーテッド・サーバーにリモート・ユーザー ID とリモート・パスワードを提供する関数を含むユーティリティー・クラスです。

共通メソッド

LDAP サーバーからユーザー・マッピングを取り出すサンプル Java プラグインで、これらの関数の使用法を参照できます。このファイルは sqllib/samples/federated/umplugin/ldap/ ディレクトリーにあります。このクラスの関数は、UserMappingRepositoryLDAP.java および UserMappingLookupLDAP.java ファイルで使用されます。

UserMappingEntry getEntry()
この UserMappingOption オブジェクトに関連付けられている UserMappingEntry オブジェクトを戻します。

String getName()
オプションの名前を戻します。

void setName()
オプションの名前を設定します。

UserMappingOption getNextOption()
次のオプションを戻します。

void setNextOption(UserMappingOption **nextOption)**
次のオプションを設定します。

abstract object getValue()
オプションの値を戻します。この関数は、ストリング値またはバイナリー値のいずれかを戻さなければなりません。以下にリストされている StringOption および BinaryOption メソッドを参照してください。

StringOption クラス

StringOption クラスは UserMappingOptions クラスから継承されたもので、getValue() および setValue() という共通メソッドが含まれています。

共通メソッド

object getValue()
オプションがストリングの場合にストリング・オプションの値を戻します。

void setValue(String value)
ストリング・オプションの値を設定します。

BinaryOption クラス

BinaryOption クラスは UserMappingOption クラスから継承されたもので、getValue() および void setValue() という共通メソッドが含まれています。

共通メソッド

object getValue()
バイナリー・オプションの値を戻します。

void setValue(byte[] value)
バイナリー・オプションの値を設定します。

UserMappingException クラス (Java プログラミング言語)

Java ユーザー・マッピング・プラグインは、エラーの報告に java.lang.Exception クラスのサブクラスである UserMappingException クラスを使用します。

共通メソッド

LDAP サーバーからユーザー・マッピングを取り出すサンプル Java プラグインで、これらの関数の使用法を参照できます。このファイルは `sqllib/samples/federated/umplugin/ldap/` ディレクトリーにあります。このクラスの関数は、各サンプル Java ファイルで使用されます。

`UserMappingException(int errorNumber)`
UserMappingException オブジェクトのインスタンス化に使用されるコンストラクターは、エラーを報告するために用いられます。 UserMappingException オブジェクトに送信される **`errorNumber`** パラメーターは、レポートされるエラーのタイプを定義します。

`int getErrorNumber()`
例外のエラー番号を戻します。

UserMappingRepositoryLDAP.java ファイルには、LDAP プラグインをテストするためにエラーをキャッチして報告するこの関数が含まれます。

`String getErrorMessage()`
例外のエラー・メッセージを戻します。

UserMappingRepositoryLDAP.java ファイルには、LDAP プラグインをテストするためにエラーをキャッチして報告するメソッドが含まれます。

エラー・メッセージ

次の表に、エラー番号、定数名、およびエラー・メッセージをリストします。

表 25. エラー番号およびメッセージ

エラー番号	定数名	エラー・メッセージ
1	INITIALIZE_ERROR	プラグインは初期化に失敗しました。
2	CONNECTION_ERROR	リポジトリーに接続できません。
3	AUTHENTICATION_ERROR	リポジトリーを認証できません。
4	LOOKUP_ERROR	リポジトリーの検索に失敗しました。
5	DECRYPTION_ERROR	暗号化解除は失敗しました。
6	DISCONNECT_ERROR	リポジトリーから切断できません。
7	INVALID_PARAMETER_ERROR	パラメーターが無効です。
8	UNAUTHORIZED_CALLER	呼び出し元にプラグインの呼び出し権限がありません。

サンプル・ユーザー・マッピング・プラグイン (Java プログラミング言語)

サンプル Java プラグインは、LDAP サーバーからユーザー・マッピング項目を取り出します。ご使用の環境でプラグインを使用するには、LDAP サーバーが使用している設定と一致するようにサンプル・プラグインを変更します。

プラグインのファイルは sqllib/samples/federated/umplugin/ldap/ ディレクトリーにあります。次の表で、各ファイルを説明します。ファイルに変更を加える前に、ファイルを空の作業ディレクトリーにコピーしてください。その後で、コピーを使用して作業します。

表 26. サンプル・ユーザー・マッピング・プラグインのファイルの説明 (Java)

ファイル名	説明
README.txt	このファイルには、サンプル・プラグインをテストおよび使用するための指示および説明の要約が含まれています。
UserMappingCryptoLDAP.java	この Java クラスには、LDAP サーバーから取り出されるユーザー・マッピングの暗号化、暗号化解除、エンコード、およびデコードに関するセキュリティー対策をインプリメントするコードが含まれています。このファイルは、LDAP サーバーを使用するように変更する必要があります。
UserMappingSetupLDAP.java	この Java クラスは、LDAP 接続情報およびその他の構成パラメーター (例えば IP アドレスまたはホスト名、SSL または非 SSL、ユーザー ID、およびパスワード) を保管する構成ファイルを作成します。
UserMappingRepositoryLDAP.java	この Java クラスには、LDAP サーバーとの接続および切断、LDAP サーバーからのユーザー・マッピングの取り出しを行うコードが含まれています。このファイルのコードは、schema.ldif ファイルで定義されているスキーマを使用します。スキーマを変更する場合はこのファイルのセクションも変更しなければなりません。
UserMappingLookupLDAP.java	この Java クラスには、LDAP 検索テストを実行するコードが含まれています。このファイルを使用して、フェデレーテッド・サーバーの外にあるプラグインをテストします。次に、フェデレーテッド・サーバーにあるプラグインをテストします。
schema.ldif	このファイルはスキーマを定義するために LDAP サーバーにロードされます。 LDIF ファイルには、LDAP サーバーに追加されるオブジェクトおよび属性が含まれます。
entry.ldif	entry.ldif ファイルは LDAP サーバーにユーザー項目を追加します。ユーザー項目はユーザー・マッピングの保管に、schema.ldif ファイルのオブジェクトと属性を使用します。

ユーザー・マッピング・プラグインの作成 (Java プログラミング言語)

ユーザー・マッピングを外部リポジトリーから取り出す Java プラグインを作成する開始点として、サンプル・コードを使用します。このサンプル・コードは、マッピングを LDAP リポジトリーから取り出しますが、このコードを任意の外部リポジトリーにアクセスするように変更することができます。

始める前に

以下を確認してください。

- Java Development Kit (JDK) バージョン 1.4 以降がインストールされている。
- db2umplugin.jar ファイル。この Java アーカイブ (JAR) ファイルは、DB2 サーバーのインストールまたは DB2 クライアントのインストールの一部としてインストールされます。
- サンプル・ユーザー・マッピング・プラグイン・ファイルがインストールされている。これらのファイルは DB2 クライアントのインストールの一部としてインストールされるもので、sqllib/samples/federated/umplugin/ldap/ ディレクトリーにあります。
- **java_heap_sz** パラメーターが 2048 に設定されている。
-

このタスクについて

作成するプラグインは、外部リポジトリーへの接続、ユーザー・マッピングの取り出し、およびリモート・パスワードの暗号化解除が行えなければなりません。使用するリポジトリーに応じて、プラグインのコーディング方法が異なります。例えば、暗号化されたパスワードを保管する LDAP リポジトリーを使用する場合、プラグインには暗号化スキーマとパスワードのデコードに必要な秘密鍵が含まれている必要があります。

プラグインを開発して使用する際には、機密のユーザー ID とパスワードを複数のソース間で送受信することに注意してください。この情報を保護するには、プラグイン・ソース・コードへのアクセスを制限して、フェデレーテッド・サーバーがプラグインを使用するたびに診断ログ・ファイル db2diag.log の VALIDATE レコードをキャプチャーするように db2audit 機能を構成します。診断ログ・ファイルは、使用状況を追跡するだけでなく、問題が発生した場合のトラブルシューティングにも役立ちます。

プラグインを作成するには、以下のタスクを実行します。

サンプル・ユーザー・マッピング・プラグイン・ファイルの変更 (Java プログラミング言語)

サンプル Java プラグインは、LDAP サーバーからユーザー・マッピングを取り出します。このプラグインに変更を加えることにより、他のタイプの外部リポジトリーからユーザー・マッピングを取り出すことができます。このサンプル・プラグインは、特定の外部リポジトリーで使用するカスタマイズされたプラグインを作成するための開始点として使用します。

それぞれのサンプル・ファイルは、ユーザー・マッピングの取り出しプロセスで異なるタスクを果たします。ご使用の外部リポジトリーで使用するには、サンプル・コードの関数およびクラスの多くに変更を加えます。以下の表に、重要な関数をリストします。

表 27. 変更する関数およびクラス

ファイル名	変更するエレメント	参照クラス
UserMappingCryptoLDAP.java	UserMappingCryptoLDAP() encrypt() decrypt() getKey() decode() encode()	UserMappingCrypto クラス UserMappingException クラス
UserMappingRepositoryLDAP.java	class UserMappingRepositoryLDAP(String configFilePath) connect() disconnect() fetchUM()	UserMappingRepository クラス UserMappingEntry クラス UserMappingOption クラス UserMappingException クラス
UserMappingSetupLDAP.java	このファイルに変更を加えて、外部リポジトリーに接続してそこからユーザー・マッピングを取り出すためにプラグインが必要とする値を保管する構成ファイルを作成します。構成ファイルを手動で作成する場合には、ファイルに保管するパスワードを確実に暗号化してください。構成ファイル名は、リポジトリー・クラスの名前と一致する必要があります (例えば UserMappingRepository*XXXX* クラスと UserMappingRepository*XXXX*.cfg ファイルなど)。	

UserMappingCryptoLDAP サンプル・ファイルの変更 (Java プログラミング言語)

LDAP サーバーが使用するセキュリティー・メソッドをインプリメントするには、リモート・パスワードの暗号化、暗号化解除、エンコード、およびデコードを行う関数を変更します。

このタスクについて

暗号化メソッドは秘密かつ固有のものでなければならないため、このタスクにより変更するセクションと関数が指定されます。このコードをカスタマイズして、LDAP サーバーが使用するセキュリティー・メソッドをインプリメントします。

UserMappingCryptoLDAP ファイルのセキュリティー機能を変更するには、次のようにします。

手順

1. テキスト・エディターで UserMappingCryptoLDAP.java ファイルを開きます。
2. IBM の著作権および特記事項のもとで、コードが参照するパッケージをインポートします。 サンプル・プラグインは javax.crypto および javax.crypto.spec の各 Java パッケージを使用します。これらのパッケージには、暗号 (暗号化および暗号化解除) のクラス、およびキーとアルゴリズムのパラメーターがあります。これらの Java パッケージを、独自のものと置き換えます。
3. 次の関数を更新します。

public UserMappingCryptoLDAP()
暗号のコードを、LDAP サーバーが使用するパスワードの暗号化と一致する暗号のコードで置き換えます。

public byte[] encrypt(byte[] plainValue)
この関数は、パスワードを LDAP サーバーに保管できるように暗号化するコードを提供します。さらにこの関数は、構成ファイルに保管される LDAP 接続パスワードも暗号化します。

この関数のコードを、**plainValue** パラメーターを暗号化する独自のコードで置き換えます。

public byte[] decrypt(byte[] encryptedValue)
この関数のコードを、**encryptedValue** パラメーターを暗号化解除する独自のコードで置き換えます。

private SecretKey getKey()

この関数のコードを、パスワードの暗号化および暗号化解除に使用されるキーをプラグインに提供するコードで置き換えます。

public byte[] decode(String string)
パスワードはまず暗号化され、その後エンコードされます。この関数は、パスワードの暗号化解除を行う前にこれをデコードするコードを提供します。暗号化されたパスワードは、そのバイナリー出力を ASCII 文字にトランスフォームするためにエンコードされます。

この関数のコードを、**string** パラメーターをデコードする独自のコードで置き換えます。

public String encode(byte[] bytes)
パスワードはまず暗号化され、その後エンコードされます。この関数は、暗号化されたパスワードのバイナリー出力をエンコードするコードを提供します。暗号化されたパスワードは、そのバイナリー出力を ASCII 文字にトランスフォームするためにエンコードされます。

この関数のコードを、**bytes** パラメーターをエンコードする独自のコードで置き換えます。

UserMappingRepositoryLDAP サンプル・ファイルの変更 (Java プログラミング言語)

Java プラグインの UserMappingRepositoryLDAP.java ファイルには、LDAP サーバーに接続し、ユーザー・マッピングを取り出し、切断するための関数が含まれています。 LDAP サーバーのディレクトリー構造で定義されているオブジェクト・クラスと一致するようにサンプル・コードを変更してください。

このタスクについて

LDAP サーバーからユーザー・マッピングを取り出すには、プラグインがディレクトリーを検索して、ユーザー・マッピングを定義する属性を持つユーザー項目を探す必要があります。ユーザーの属性として次の情報が保管されます。

- リモート・サーバー名
- インスタンス名
- データベース名

- リモート・ユーザー名
- リモート・ユーザー・パスワード

サンプル・コードは、ユーザー項目が *inetOrgPerson* オブジェクト・クラスによって識別され、ユーザー・マッピング項目が *IIUserMapping* オブジェクト・クラスによって識別されることを前提とします。 Lightweight Directory Interchange Format (LDIF) サンプル・ファイル (`schema.ldif` および `entry.ldif`) は、サンプル・スキーマおよびサンプル項目を LDAP サーバーにロードします。

`UserMappingRepositoryLDAP.java` ファイルのコードは、次の名前のオブジェクトおよび属性の `schema.ldif` ファイルに LDIF コードが含まれることを前提とします。

- *IIUserMapping* オブジェクト
  - *IIRemoteServerName* 属性
  - *IIInstanceName* 属性
  - *IIDatabaseName* 属性
  - *IIRemotePassword* 属性
  - *uid* 属性

LDAP サーバーが使用するスキーマと一致するように `UserMappingRepositoryLDAP.java` ファイルを変更してください。`UserMappingRepositoryLDAP.java` ファイルは LDAP サーバーからユーザー・マッピング項目を検索し、それを取り出します。ユーザー・マッピング項目を保管するサンプル・スキーマおよびメソッドとして、LDIF ファイルが提供されます。

`UserMappingRepositoryLDAP.java` ファイルが使用するスキーマを変更するには、次のようにします。

手順

1. コード `private String UserObjectClassName = "inetOrgPerson"` を見つけ、*inetOrgPerson* 値を、LDAP サーバーがユーザー項目に使用するオブジェクト・クラスの名前で置き換えます。
2. オプション: プラグインが使用する属性名を変更します。*IIRemoteServerAttrName*、*IIInstanceAttrName*、*IIDatabaseAttrName*、および *IIRemotePasswordAttrName* 変数の値を、選択する属性名で置き換えます。
3. オプション: LDIF ファイルを使用する場合は、LDIF ファイルのスキーマを、`UserMappingRepositoryLDAP.java` ファイルが検索する構造と一致させてください。

ユーザー・マッピング・プラグイン・ファイルのコンパイル (Java プログラミング言語)

Java プラグインのソース・ファイルを変更した後は、コンパイルが必要です。

このタスクについて

以下のコマンドの絶対パスにスペースが含まれる場合、絶対パスを引用符で囲む必要があります。例えば、`"C:¥program files¥sqllib¥java¥db2umplugin.jar"` としま

す。 *%DB2PATH%* は DB2 がインストールされているディレクトリーで、例えば C:¥ProgramFiles¥IBM¥sqllib のようになります。*inst_home* はインスタンスのホーム・ディレクトリーです。

以下のコマンドでは、クラスの名前に基づくファイル命名規則を使用していることを前提としています。 *xxxx* は、選択する名前で置き換えます。

ソース・ファイルをコンパイルするには、次のようにします。

手順

1. コンパイル・コマンドを実行します。

   Windows:

   ```
   javac -classpath" %DB2PATH%¥java¥db2umplugin.jar; ^
          %CLASSPATH%" -d . ^
          .¥UserMappingRepositoryxxxx.java ^
          .¥UserMappingCryptoxxxx.java ^
          .¥UserMappingSetupxxxx.java ^
          .¥UserMappingLookupXXXX.java
   ```

   UNIX:

   ```
   javac -classpath inst_home/sqllib/java/db2umplugin.jar:¥
          $CLASSPATH -d . \
          ./UserMappingRepositoryxxxx.java ¥
          ./UserMappingCryptoxxxx.java ¥
          ./UserMappingSetupxxxx.java ¥
          ./UserMappingLookupxxxx.java
   ```

2. Java クラス・ファイルを 1 つの Java Archive (JAR) ファイルにアーカイブします。 出力ファイル名の後ろのピリオドは、同じディレクトリー内でファイルの検出および作成を行うようコマンドに命令します。ディレクトリーを変更する場合、ご使用のオペレーティング・システムに適したファイル・パスを使用してください (たとえば、/home/user/folder または C:¥test¥folder)。

   ```
   jar -cfM0 UserMappingRepositoryXXXX.jar .
   ```

次のタスク

ユーザー・マッピング・プラグインの構成ファイルの作成 (Java プログラミング言語)

構成ファイルには、Java プラグインが LDAP サーバーへの接続に使用する接続情報が保管されます。

このタスクについて

構成ファイルを作成するには、構成プログラムを実行します。このとき、以下の情報の入力を求められます。

- LDAP サーバーのホスト名または IP アドレス
- LDAP サーバーのポート番号 (デフォルトは 389)
- LDAP サブツリーの識別名 (例えば ou=ii、o=ibm、c=us など)
- LDAP サーバーに接続するためのユーザー ID
- LDAP サーバーに接続するためのパスワード
- SSL 構成

入力した情報は、UserMappingRepositoryLDAP.cfg ファイルの作成に使用されます。このファイルには構成情報が保管されます。LDAP サーバーへの接続にパスワードが必要な場合、このパスワードは UserMappingCryptoLDAP.java ファイルで指定したアルゴリズムを使用して暗号化されます。

以下のコマンドの絶対パスにスペースが含まれる場合、絶対パスを引用符で囲む必要があります。例えば、"C:¥program files¥sqllib¥java¥db2umplugin.jar" とします。 *%DB2PATH%* は DB2 がインストールされているディレクトリーで、例えば C:¥ProgramFiles¥IBM¥sqllib のようになります。*inst_home* はインスタンスのホーム・ディレクトリーです。

サンプル LDAP プラグインの構成ファイルを作成するには、次のようにします。

Windows:

```
java -classpath "%DB2PATH%¥java¥db2umplugin.jar; ^
 .¥UserMappingRepositoryLDAP.jar;%CLASSPATH%" UserMappingSetupLDAP
```

UNIX:

```
java -classpath inst_home/sqllib/java/db2umplugin.jar: ¥
 ./UserMappingRepositoryLDAP.jar:$CLASSPATH UserMappingSetupLDAP
```

ユーザー・マッピング・プラグインのテスト (Java プログラミング言語)

フェデレーテッド・サーバーの外部で Java プラグインのテストを行うために、外部リポジトリーのユーザー・マッピングへの接続およびその取り出しを行うアプリケーションを作成します。

このタスクについて

UserMappingRepository*XXXX* クラスが UserMappingRepository クラスから継承した lookupUM() メソッドを呼び出す単純なプログラムを作成して、外部リポジトリーに接続し、ユーザー・マッピングを取り出すことができます。 sqllib/samples/federated/umplugin/ldap/ディレクトリーにある UserMappingLookupLDAP.java ファイルを表示することができます。

以下のコマンドの絶対パスにスペースが含まれる場合、絶対パスを引用符で囲む必要があります。例えば、"C:¥program files¥sqllib¥java¥db2umplugin.jar" とします。 *%DB2PATH%* は DB2 がインストールされているディレクトリーで、例えば C:¥ProgramFiles¥IBM¥sqllib のようになります。*inst_home* はインスタンスのホーム・ディレクトリーです。

テスト・プログラムは、ユーザー・マッピングの識別に必要な次のパラメーターを取る必要があります。

- *remoteServerName* - データ・ソースのリモート・サーバー名
- *iiAuthid* - ユーザー・マッピングに関連付けられているローカル・ユーザー ID
- *iiInstance* - フェデレーテッド・サーバーのインスタンス名
- *iiDatabase* - フェデレーテッド・サーバー上のデータベース名

ユーザー・マッピング・プラグインをテストするには、次のようにします。

Windows:

```
java -classpath "%DB2PATH%¥java¥db2umplugin.jar; ^
  .¥UserMappingRepositoryXXXX.jar;%CLASSPATH%" ^
  UserMappingLookupXXXX -server remoteServerName ^
   -authid iiAuthid ^
  -instance iiInstance -database iiDatabase
```

UNIX:

```
java -classpath inst_home/sqllib/java/db2umplugin.jar: ^
  ./UserMappingRepositoryXXXX.jar:$CLASSPATH ¥
  UserMappingLookupXXXX -server remoteServerName ¥
  -authid iiAuthid ¥
  -instance iiInstance -database iiDatabase
```

次のタスク

ユーザー・マッピング・プラグイン・ファイルのデプロイ (Java プログラミング言語)

Java プラグインをコンパイルしてテストした後で、このファイルをフェデレーテッド・サーバーにデプロイします。

このタスクについて

以下のコマンドの *sqllib* は、DB2 インストールの絶対パスを表します。このコマンドでは、クラスの名前に基づくファイル命名規則を使用していることを前提としています。 *xxxx* は、選択する名前で置き換えます。

コンパイル済みのファイルをデプロイするには、次のようにします。

手順

UserMappingRepository*xxxx*.jar および UserMappingRepository*xxxx*.cfg ファイルを *sqllib*/function/ ディレクトリーにコピーします。

タスクの結果

このディレクトリーにファイルを置くことにより、フェデレーテッド・サーバーはそれらのファイルに含まれているクラスをロードし、呼び出すことができます。

ユーザー・マッピング・プラグインへのアクセスの構成 (Java プログラミング言語)

Java プラグインを使ってユーザー・マッピングを取得するようフェデレーテッド・サーバーを構成するには、DB2_UM_PLUGIN オプションを設定します。

始める前に

始める前に

外部リポジトリー内のユーザー・マッピングにアクセスするようフェデレーテッド・サーバーを構成する前に、以下の作業を行う必要があります。

- ユーザー・マッピング・プラグインを開発する
- プラグインをフェデレーテッド・サーバーにデプロイする
- データベース・マネージャー構成を次のように更新する

```
db2 update dbm cfg using JDK_PATH your_jdk_path
db2 terminate
db2stop
db2start
```

このタスクについて

DB2_UM_PLUGIN オプションには、パッケージ名を含む、クラスの絶対パスが含まれていなければなりません。パッケージを作成する場合、クラス名の前にパッケージ名を含める必要があります (たとえば 「package.classname」)。LDAP サンプル・プラグインを使用する場合、DB2_UM_PLUGIN オプションの中で 'UserMappingRepositoryLDAP' という値を指定します。サンプル・プラグインはパッケージとしては開発されていません。

外部リポジトリーにアクセスするようフェデレーテッド・サーバーを構成するには、次のようにします。

手順

ユーザー・マッピング・プラグインをインプリメントする方法を選択します。

Method	SQL ステートメント
ラッパーの作成時にユーザー・マッピング・プラグインを指定する	CREATE WRAPPER *wrapper-name* OPTIONS (DB2_UM_PLUGIN 'UserMappingRepositoryLDAP');
ユーザー・マッピング・プラグインを指定するよう既存のラッパーを変更する	ALTER WRAPPER *wrapper-name* (ADD DB2_UM_PLUGIN 'UserMappingRepositoryLDAP');
サーバー定義の作成時にユーザー・マッピング・プラグインを指定する	CREATE SERVER *server_name* TYPE *date_source_type* VERSION *version_number* WRAPPER *wrapper-name* OPTIONS (DB2_UM_PLUGIN 'UserMappingRepositoryLDAP');
ユーザー・マッピング・プラグインを指定するよう既存のサーバー定義を変更する	ALTER SERVER *server_name* OPTIONS (ADD DB2_UM_PLUGIN 'UserMappingRepositoryLDAP');

次のタスク

DB2_UM_PLUGIN オプションを設定した後、フェデレーテッド・サーバーは UserMappingRepository*XXXX*.cfg ファイルで指定された接続情報を使用して、ユーザー・マッピングを外部リポジトリーから取得します。 *XXXX* はプラグインの名前です。

別のプラグインを使用するようラッパーを変更するには、次のようにします。

```
ALTER WRAPPER wrapper-name (
  SET DB2_UM_PLUGIN 'com.package_name.um.UserMappingRepositoryXXXX'
  );
```

別のプラグインを使用するようサーバー定義を変更するには、次のようにします。

```
ALTER SERVER server_name
  OPTIONS (
     SET DB2_UM_PLUGIN 'UserMappingRepositoryXXXX'
     );
```

第 39 章 フェデレーテッド・システムにおける Oracle セキュリティー

Federation は、Oracle セキュリティー・フィーチャーのセットをサポートします。

これらのフィーチャーは、Oracle に実装すると、フェデレーテッド・サーバーで利用できるようになります。

Oracle Label Security

フェデレーテッド・サーバーのサポートする Oracle Label Security を使用して、適切な権限を持つユーザーのみが見ることができるように、データを保護できます。

管理者は、Oracle Label Security を使用して表内の各行にセキュリティー・ポリシーを適用します。セキュリティー・ポリシーは、データに対するユーザーのアクセス・レベルを、そのユーザー ID またはセッション ID に認可された権限に基づいて判別します。 Oracle データ・ソース・オブジェクトにニックネームを作成すると、フェデレーテッド・サーバーはそのデータ・ソースが Oracle Label Security を使用するかどうかを自動的に検出します。 Oracle Label Security が使用されている場合、ニックネームはキャッシュされません。

ALTER NICKNAME ステートメントを使用して、キャッシングを許可または不許可にできます。例えば、Oracle Label Security を使用するデータ・ソース・オブジェクトに、このフィーチャーに対するフェデレーテッド・サポートが使用可能になる前にニックネームを作成していた場合、ニックネームを変更してキャッシングを不許可にすることができます。 Oracle Label Security を使用するデータ・ソース・オブジェクトにニックネームを作成していて、Oracle Label Security が除去された場合、ニックネームを変更してキャッシングを許可することができます。

データベース管理者は、ラベルを非表示にして、一部のユーザーに特定の行が存在することが分らないようにすることを選択できます。この場合、その列は表内で非表示になります。ラベル列が非表示のニックネームはキャッシュされません。

Oracle プロキシー認証およびフェデレーテッド・トラステッド・コンテキスト

Oracle データ・ソースに対する物理接続を作成し、次にその接続を同じ接続上の別のユーザーに切り替えます。

Oracle プロキシー認証とフェデレーテッド・トラステッド・コンテキストを使用すると、接続ユーザーの ID を引き続き Oracle データ・ソースに対して明確にアサートしながら、ユーザーごとにフェデレーテッド・サーバーから Oracle データベースへの個別のネットワーク接続を作成するネットワーク・オーバーヘッドを削減できます。アプリケーションは必要に応じてユーザーを切り替えて、ユーザーの代わりにトランザクションを処理することができます。

このシナリオを構成するには、以下のタスクを実行します。

1. Oracle サーバーで、Oracle の ALTER USER ステートメントを使用して各プロキシー・ユーザーを登録します。ここで、プロキシー・ユーザー Mary は BOSS という名前のプロキシーを使用する許可を付与され、接続の間は CLERK という役割を得ます。

   ```
   ALTER USER MARY GRANT
   CONNECT THROUGH BOSS
   WITH ROLE CLERK
   ```

2. フェデレーテッド・サーバー上で、トラステッド・コンテキスト・オブジェクトを作成します。

   ```
   CREATE TRUSTED CONTEXT MY_FED_TCX
   BASED UPON CONNECTION USING SYSTEM AUTHID BOSS
   ATTRIBUTES (ENCRYPTION 'NONE')
   WITH USE FOR MARY WITHOUT AUTHENTICATION
   ENABLE
   ```

この構成では、フェデレーテッド・サーバーは、クライアントからフェデレーテッド・サーバーを介して Oracle データ・ソースに対するエンドツーエンドのトラステッド接続をセットアップできます。 BOSS はトラステッド接続を確立でき、MARY はその接続を再利用できます。

トラステッド接続を確立および再利用するには、DB2 提供の API を使用してください。

第 40 章 フェデレーテッド・フィーチャーのデータ・ソース・サポート

データ・ソースが特定のフェデレーテッド・フィーチャーをサポートするかどうかを知るためには、この表を参照してください。

これらのフィーチャーを使用する前に、特定のラッパーまたはサーバー・オプションを設定するか、あるいは他の操作を行って、その機能を使用可能にする必要がある場合があります。詳しくは、各フィーチャーの特定のトピックを参照してください。

表 28. フィーチャーおよびサポートされるデータ・ソース

フィーチャー	データ・ソース
ニックネームに対して WRITE 操作を指定したアプリケーション・セーブポイント	DB2 for Linux, UNIX, and Windows
非同期最適化	すべてのデータ・ソース
キャッシュ表	DB2 ファミリー Informix Microsoft SQL Server Oracle Sybase
ニックネームへのデータのインポート	DB2 ファミリー Informix Microsoft SQL Server Oracle Sybase Teradata
ネストされた表の式におけるエラー耐性	DB2 ファミリー Informix JDBC Microsoft SQL Server ODBC Oracle Sybase Teradata
外部ユーザーのマッピング・リポジトリー	すべてのデータ・ソース

表 28. フィーチャーおよびサポートされるデータ・ソース (続き)

フィーチャー	データ・ソース
フェデレーテッド・ヘルス・インディケーター	DB2 ファミリー Excel Informix JDBC Microsoft SQL Server ODBC Oracle Sybase 表構造ファイル Teradata XML (ルート・ニックネームのみ)
フェデレーテッド・プロシージャー	DB2 ファミリー (トラステッド・モード) Oracle (トラステッド・モード) Microsoft SQL Server (トラステッド・モード) Sybase (fenced モード、UNIX にフェデレーテッド・サーバーをインストール済み) Sybase (fenced モードまたはトラステッド・モード、Linux または Microsoft Windows にフェデレーテッド・サーバーをインストール済み)
フェデレーテッド・トラステッド・コンテキスト	DB2 for Linux, UNIX, and Windows バージョン 9.5 DB2 for z/OS バージョン 9 Oracle
HTTP プロキシー	Web サービス XML
接続レベルの分離	DB2 ファミリー Informix JDBC Microsoft SQL Server ODBC Oracle Sybase
ラベル・ベースのアクセス制御	DB2 for Linux, UNIX, and Windows バージョン 9.1 および 9.5 Oracle
LOB 読み取りおよび書き込み操作	DB2 for z/OS DB2 for Linux, UNIX, and Windows DB2 for System i Oracle Teradata
LOB 読み取り専用操作	BioRS Informix JDBC Microsoft SQL Server ODBC スクリプト Sybase Web サービス XML

表 28. フィーチャーおよびサポートされるデータ・ソース (続き)

フィーチャー	データ・ソース
マテリアライズ照会表	すべてのデータ・ソース (固有の制約事項あり)
ニックネーム統計の更新機能	BioRS DB2 ファミリー Excel Informix JDBC Microsoft SQL Server ODBC Oracle Sybase 表構造ファイル Teradata XML (ルート・ニックネームのみ)
パススルー・セッション	DRDA Informix Oracle Microsoft SQL Server Sybase Teradata
リモート XML データ・タイプ	DB2 for Linux, UNIX, and Windows XML ラッパー
SOCKS プロキシー	BioRS スクリプト Web サービス XML すべてのリレーショナル・データ・ソース **注:** リレーショナル・データ・ソースに接続するには、 DB2ENVLIST=SOCKS5C_CONFIG を設定する必要があります。
Secure Socket Layer (SSL)	Web サービス XML
ステートメント・レベルの分離	DB2 ファミリー Microsoft SQL Server
2 フェーズ・コミット・トランザクション	DB2 for Linux, UNIX, and Windows (トラステッド・モード) DB2 for System i (トラステッド・モード) DB2 for z/OS (トラステッド・モード) Informix (トラステッド・モード) Microsoft SQL Server (トラステッド・ モード、Microsoft Windows にフェデレーテッド・ サーバーをインストール済み) Oracle (トラステッド・モード) Sybase (トラステッド・モード、Microsoft Windows にフェデレーテッド・サーバーをインストール済み) Sybase (fenced モード、UNIX にフェデレーテッド・ サーバーをインストール済み)
Unicode サポート	すべてのデータ・ソース

第 41 章 データ・ソース・オプションの参照情報

各データ・ソースは、特定のラッパー、サーバー、ユーザー・マッピング、ニックネーム、および列のオプションをサポートします。

BioRS オプション・リファレンス

フェデレーテッド・サーバーおよびそのユーザーがデータ・ソースと対話する方法を構成するには、ラッパー、サーバー、ユーザー・マッピング、ニックネーム、および列のオプションを設定および変更します。

ラッパー・オプション

以下の表に、このデータ・ソースに適用されるオプションと、CREATE WRAPPER ステートメントおよび CREATE SERVER ステートメントに指定する必要のある必須指定のオプションをまとめました。

表 29. BioRS のラッパー・オプション

名前	説明
DB2_FENCED	必須。ラッパーを fenced モードとトラステッド・モードのどちらで実行するかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合、ラッパーはトラステッド・モードで実行されます。
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「UserMappingRepositoryLDAP」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は、Java および C です。デフォルトは Java です。
PROXY_SERVER_NAME	プロキシー・サーバーの名前または IP アドレスを指定します。 PROXY_TYPE の値が HTTP または SOCKS の場合、このオプションは必須です。有効な IP アドレスは IPv4 (ドット区切り) 形式または IPv6 (コロン区切り) 形式です。 IPv6 形式は、IPv6 を構成する場合のみ使用します。

表 29. *BioRS* のラッパー・オプション *(続き)*

名前	説明
PROXY_SERVER_PORT	プロキシー・サーバー上のプロキシー・サービスのポートまたはサービス名を指定します。 PROXY_TYPE の値が HTTP または SOCKS の場合、このオプションは必須です。有効な値は、10 進数の 1 から 32760 までのポート番号か、サービス名です。
PROXY_TYPE	フェデレーテッド・サーバーがファイアウォールの背後にある場合に、インターネットへのアクセスに使用するプロキシー・タイプを指定します。有効な値は NONE、HTTP、および SOCKS です。デフォルト値は NONE です。このオプションを HTTP または SOCKS に設定した場合は、PROXY_SERVER_NAME と PROXY_SERVER_PORT も指定する必要があります。

サーバー・オプション

表 30. *BioRS* のサーバー・オプション

名前	説明
CASE_SENSITIVE	BioRS サーバーが大/小文字の区別をして名前を扱うかどうかを指定します。有効な値は Y と N です。デフォルトは N です。この場合、名前は大/小文字の区別をして扱われます。 BioRS 製品では、BioRS サーバーに保管されているデータの大/小文字の区別を構成パラメーターで制御します。大/小文字の区別の指定は、CASE_SENSITIVE オプションと構成パラメーターとで同一でなければなりません。サーバー定義を作成した後で CASE_SENSITIVE オプションの値の変更が必要になった場合は、サーバー定義をドロップして、その定義とすべてのニックネームを再度作成する必要があります。
DB2_MAX_ASYNC_REQUESTS_ PER_QUERY	照会からの並行非同期要求の最大数を指定します。有効な値は -1 から 64000 です。デフォルトは 1 です。 -1 を指定すると、要求の数はフェデレーテッド照会オプティマイザーが決定します。 0 は、データ・ソースが、追加の非同期要求に対応できないということを指定します。

表 30. BioRS のサーバー・オプション (続き)

名前	説明
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「UserMappingRepositoryLDAP」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は Java および C です。デフォルトは Java です。
NODE	必須。BioRS 照会ツールが使用可能なシステムの DNS ホスト名または IP アドレスを指定します。有効な IP アドレスは IPv4 (ドット区切り) 形式または IPv6 (コロン区切り) 形式です。 IPv6 形式は、IPv6 を構成する場合のみ使用します。デフォルト値は localhost です。
PORT	BioRS サーバーに接続するポートを指定します。有効な値は、数値ポートまたは TCP/IP サービス名です。デフォルトは 5014 です。
PROXY_AUTHID	プロキシー・サーバー認証のユーザー名を指定します。
PROXY_PASSWORD	プロキシー・サーバー認証のパスワードを指定します。
PROXY_SERVER_NAME	プロキシー・サーバーの名前または IP アドレスを指定します。有効な値は、1 から 32760 までの 10 進数のポート番号、またはサービス名です。 IPv6 形式は、IPv6 を構成する場合のみ使用します。
PROXY_SERVER_PORT	プロキシー・サーバー上のプロキシー・サービスのポートまたはサービス名を指定します。有効な値は、10 進数の 1 から 32760 までのポート番号か、サービス名です。
PROXY_TYPE	フェデレーテッド・サーバーがファイアウォールの背後にある場合に、インターネットへのアクセスに使用するプロキシー・タイプを指定します。有効な値は NONE、HTTP、および SOCKS です。デフォルト値は NONE です。
TIMEOUT	リモート・サーバーからの応答をフェデレーテッド・サーバーが待つ最大時間 (分単位) を指定します。デフォルトは 10 です。

ユーザー・マッピング・オプション

表 31. BioRS のユーザー・マッピング・オプション

オプション	説明
GUEST	操作実行のために GUEST BioRS 認証 ID を使用することを指定します。有効な値は Y と N です。デフォルトは N です。この場合、GUEST BioRS ID は使用されません。REMOTE_AUTHID オプションおよび REMOTE_PASSWORD オプションを指定した場合、このオプションは無効です。
PROXY_AUTHID	プロキシー・サーバー認証のユーザー名を指定します。
PROXY_PASSWORD	プロキシー・サーバー認証のパスワードを指定します。このパスワードは、フェデレーテッド・データベース・カタログに保管されるときに暗号化されます。
REMOTE_AUTHID	ローカル・ユーザー ID のマップ先となるリモート・ユーザー ID を指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用する ID が使用されます。
REMOTE_PASSWORD	リモート・ユーザー ID のリモート・パスワードを指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用したパスワードが使用されます。

ニックネーム・オプション

表 32. BioRS のニックネーム・オプション

オプション	説明
REMOTE_OBJECT	ニックネームに関連する BioRS データ・バンクの名前を指定します。この名前によって、ニックネームのスキーマと BioRS データ・バンクが決まります。この名前によって、そのニックネームと他のニックネームの関係も指定されます。このオプションの大/小文字の区別をするかどうかは、BioRS サーバーで大/小文字の区別をするかどうか、および CASE_SENSITIVE サーバー・オプションの値によって決まります。 ALTER NICKNAME ステートメントを使用して、この名前の変更または削除を行うことはできません。 BioRS データ・バンクの名前を変更した場合、ニックネームを削除してから、再びそれを作成しなければなりません。

表 32. *BioRS* のニックネーム・オプション *(続き)*

オプション	説明
TIMEOUT	データ・ソース・サーバーからの応答を待つ最大時間 (分単位) を指定します。デフォルトは 10 です。

列オプション

表 33. *BioRS* の列オプション

オプション	説明
ELEMENT_NAME	BioRS エレメント名を指定します。この名前の大/小文字の区別をするかどうかは、BioRS サーバーで大/小文字の区別をするかどうか、および CASE_SENSITIVE サーバー・オプションの値によって決まります。 BioRS エレメント名の指定が必須なのは、エレメント名が列名と異なる場合だけです。
IS_INDEXED	対応する列が索引付けされ、述部で列が参照可能かどうかを指定します。有効な値は Y と N です。デフォルトは N です。この場合、列は索引付けされません。
REFERENCED_OBJECT	現在の列が参照する BioRS データ・バンクの名前を指定します。この名前の大/小文字の区別をするかどうかは、BioRS サーバーで大/小文字の区別をするかどうか、および CASE_SENSITIVE サーバー・オプションの値によって決まります。このオプションは、BioRS データ・タイプが Reference である列にのみ有効です。

DB2 データベース・オプション・リファレンス

フェデレーテッド・サーバーおよびそのユーザーがデータ・ソースと対話する方法を構成するには、ラッパー、サーバー、ユーザー・マッピング、ニックネーム、および列のオプションを設定および変更します。

ラッパー・オプション

次の表は、DB2 データ・ソースに適用されるオプションをリストし、指定する必要がある必須選択のオプションを示しています。

表 34. *DB2* データ・ソースのラッパー・オプション

名前	説明
DB2_FENCED	必須。ラッパーを fenced モードとトラステッド・モードのどちらで実行するかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合、ラッパーはトラステッド・モードで実行されます。

表 34. DB2 データ・ソースのラッパー・オプション (続き)

名前	説明
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「UserMappingRepositoryLDAP」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は Java および C です。デフォルトは Java です。

サーバー・オプション

表 35. DB2 データ・ソースのサーバー・オプション

名前	説明
APP_ISOLATION_ENABLE	クライアント接続の分離レベルを指定します。有効な値は Y と N です。デフォルトは N であり、指定するとフェデレーテッド・サーバーはクライアントへの接続の分離レベルを設定しません。その場合、クライアントは、db2cli.ini などの構成ファイルの分離レベル設定を使用できます。 Y を指定すると、フェデレーテッド・サーバーは分離レベルをアプリケーションから取得し、アプリケーション分離を接続属性として設定します。
COLLATING_SEQUENCE	データ・ソースがフェデレーテッド・データベースと同じデフォルト照合シーケンスを使用するかどうかを指定します。有効な値は Y、N、および I です。I は大/小文字を区別しないことを指定します。デフォルトは Y です。フェデレーテッド・サーバーに指定された照合シーケンスは、リモート・データ・ソースの照合シーケンスと一致している必要があります。
COMM_RATE	フェデレーテッド・サーバーとデータ・ソース・サーバーとの間の通信速度を指定します (MB/秒)。有効な値は 0 より大きく、2147483648 より小さい整数です。デフォルトは 2 です。

表 35. DB2 データ・ソースのサーバー・オプション (続き)

名前	説明
CPU_RATIO	データ・ソースの CPU がフェデレーテッド・サーバーの CPU と比較してどれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。 1 を設定すると、フェデレーテッド・サーバーとデータ・ソース・サーバーの CPU 速度が同じで、1:1 の比率であるということです。 0.5 を設定すると、フェデレーテッド・サーバーの CPU の速度がデータ・ソースの CPU の半分であることを示します。 2 を設定すると、フェデレーテッド CPU がデータ・ソース CPU の 2 倍高速であるということです。
DATE_COMPAT	データベースに date_compat パラメーターを適用するかどうかを指定します。有効な値は Y および N です。デフォルトは N です。このサーバー・オプションは、DB2 Database for Linux, UNIX, and Windows バージョン 9.7 以降でのみ有効です。 InfoSphere Federation Server バージョン 9.7 フィックスパック 2 以降では、CREATE SERVER ステートメントを実行すると、このサーバー・オプションはデータ・ソースの構成に基づいて自動的に構成されます。このサーバー・オプションを手動で構成しようとすると、SQL1841N メッセージを受け取ります。
DBNAME	必須。初期のリモート DB2 データベース接続で使用する特定のデータベースを指定します。この特定のデータベースとは、CATALOG DATABASE コマンドまたは DB2 構成アシスタントを使用して、フェデレーテッド・サーバーでカタログされた、リモート DB2 データベースのデータベース別名です。
DB2_MAXIMAL_PUSHDOWN	照会オプティマイザーがアクセス・プランの選択に使用する 1 次基準を指定します。有効な値は Y および N です。デフォルトは N であり、その場合、照会オプティマイザーは見積コストが最小になるプランを選択します。 Y は、照会オプティマイザーが、照会操作を最も多くデータ・ソースにプッシュダウンするアクセス・プランを選択するということを指定します。

表 35. DB2 データ・ソースのサーバー・オプション (続き)

名前	説明
DB2_MAX_ASYNC_REQUESTS_ PER_QUERY	照会からの並行非同期要求の最大数を指定します。有効な値は -1 から 64000 です。デフォルトは 1 です。 -1 を指定すると、要求の数はフェデレーテッド照会オプティマイザーが決定します。 0 を指定すると、データ・ソースは付加的な非同期要求を受け入れることができません。
DB2_TWO_PHASE_COMMIT	フェデレーテッド・サーバーがデータ・ソースに 2 フェーズ・コミット・プロトコルと 1 フェーズ・コミット・プロトコルのどちらで接続するかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合、フェデレーテッド・サーバーは 1 フェーズ・コミット・プロトコルを使用して接続します。 Y を指定すると、フェデレーテッド・サーバーは 2 フェーズ・コミット・プロトコルを使用して接続します。
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「UserMappingRepositoryLDAP」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は Java および C です。デフォルトは Java です。
FED_PROXY_USER	インバウンド接続が非トラステッド接続である場合に、すべてのアウトバウンド・トラステッド接続を確立するために使用する許可 ID を指定します。このオプションで ID が指定されているユーザーのユーザー・マッピングには、REMOTE_AUTHID と REMOTE_PASSWORD の両方のオプションが指定されている必要があります。 **制約事項:** このサーバー・オプションは、DB2 Database for Linux, UNIX, and Windows バージョン 9.5 以降、および DB2 for z/OS バージョン 9 以降でのみ有効です。

表 35. DB2 データ・ソースのサーバー・オプション (続き)

名前	説明
FOLD_ID	データ・ソースに送信されるユーザー ID で大/小文字のどちらを使用するかを指定します。デフォルト値はありません。フェデレーテッド・サーバーは、ユーザー ID を大文字で送信します。大文字のユーザー ID で失敗した場合、サーバーはユーザー ID を小文字で送信します。有効な値は U (大文字)、L (小文字)、および N (NULL) です。NULL に設定すると、パフォーマンスが低下する場合があるため、NULL 値の使用は避けてください。
FOLD_PW	データ・ソースに送信されるパスワードで大/小文字のどちらを使用するかを指定します。デフォルト値はありません。フェデレーテッド・サーバーは、パスワードを大文字で送信し、このパスワードが失敗した場合は、パスワードを小文字で送信します。有効な値は U (大文字)、L (小文字)、および N (NULL) です。NULL に設定すると、パフォーマンスが低下する場合があるため、NULL 値の使用は避けてください。
IO_RATIO	データ・ソース入出力システムがフェデレーテッド・サーバー入出力システムと比較して、どれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。 1 を設定すると、フェデレーテッド・サーバーとデータ・ソース・サーバーの入出力速度が同じで、1:1 の比率であるということです。 0.5 を設定すると、フェデレーテッド・サーバーの速度がデータ・ソースの速度の半分であることを示します。 2 を設定すると、フェデレーテッドの速度がデータ・ソースの速度の 2 倍高速であるということです。

表 35. DB2 データ・ソースのサーバー・オプション (続き)

名前	説明
NO_EMPTY_STRING	リモート・データ・ソース・サーバーに、空ストリングを格納できるかどうかを指定します。有効な値は Y と N です。デフォルト値は、ご使用のリモート・データ・ソースによって異なります。リモート Oracle データ・ソースの場合、デフォルトは 1 であり、すべての空ストリング値は NULL 値に変換されます。その他すべてのリモート・データ・ソースの場合、デフォルトは N であり、データ・ソースに空ストリングを格納できます。 フェデレーテッド・サーバーが VARCHAR2 互換モードであるが、リモート・データ・ソースが VARCHAR2 互換ではない場合、システム構成でこのオプションを Y に設定すると、システムのパフォーマンスを改善できます。
NUMBER_COMPAT	データ・ソース・サーバーで NUMBER データ・タイプをサポートするかどうかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合、データ・ソース・サーバーは NUMBER データ・タイプをサポートしません。 NUMBER データ・タイプがフェデレーテッド・サーバーではサポートされず、データ・ソース・サーバーではサポートされるシステムの場合、NUMBER_COMPAT オプションを Y に設定する必要があります。これは、データ・ソース・サーバーが DECIMAL データ・タイプの範囲外である DECFLOAT 結果を返すことがあり、SQLSTATE 560BD エラーが発生する可能性があるためです。 **制約事項:** このサーバー・オプションは、DB2 Database for Linux, UNIX, and Windows バージョン 9.7 以降でのみ有効です。 InfoSphere Federation Server バージョン 9.7 フィックスパック 2 以降では、CREATE SERVER ステートメントを実行すると、このサーバー・オプションはデータ・ソースの構成に基づいて自動的に構成されます。このサーバー・オプションを手動で構成しようとすると、SQL1841N メッセージを受け取ります。

表 35. DB2 データ・ソースのサーバー・オプション (続き)

名前	説明
OLD_NAME_GEN	データ・ソースの列名と索引名を、フェデレーテッド・サーバーのニックネーム列名とローカル索引名に変換する方法を指定します。有効な値は Y と N です。デフォルトは N であり、その場合、生成される名前とデータ・ソース中の名前には密接な対応関係があります。 Y を指定すると、生成される名前は IBM WebSphere Federation Server バージョン 9 以前で作成された名前と同じになります。したがって、それらの名前はデータ・ソースの名前と密接に対応しないことがあります。
PUSHDOWN	フェデレーテッド・サーバーが、データ・ソースによる操作の評価を許可するかどうかを指定します。有効な値は Y および N です。デフォルトは Y であり、その場合、データ・ソースは操作を評価します。 N は、列名を指定した SELECT のみを含む SQL ステートメントをフェデレーテッド・サーバーが送信するということを指定します。述部 (WHERE= など)、列およびスカラー関数 (MAX や MIN など)、ソート (ORDER BY または GROUP BY など)、および結合は、フェデレーテッド・サーバーがデータ・ソースに送信するどの SQL にも含まれません。

表 35. DB2 データ・ソースのサーバー・オプション (続き)

名前	説明
SAME_DECFLT_ROUNDING	フェデレーテッド・サーバーとデータ・ソース・サーバーの丸めモードが、いずれも同じ DECFLOAT 丸めモード設定値を使用するかどうかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合、フェデレーテッド・サーバーとリモート・サーバーの DECFLOAT 丸めモード設定値は異なります。 **重要:** フェデレーテッド・サーバーとデータ・ソース・サーバーで丸めモードが異なる場合に、このオプションを Y に設定すると、間違った DECFLOAT 丸め結果を受け取ることがあります。 同じ DECFLOAT 丸めモード設定値を使用する既存のフェデレーテッド・サーバーおよびデータ・ソース・サーバーを構成する場合は、ALTER SERVER ステートメントを使用します。 **制約事項:** このサーバー・オプションは、DB2 Database for Linux, UNIX, and Windows バージョン 9.5 以降でのみ有効です。 InfoSphere Federation Server バージョン 9.7 フィックスパック 2 以降では、CREATE SERVER ステートメントを実行すると、このサーバー・オプションはデータ・ソースの構成に基づいて自動的に構成されます。このサーバー・オプションを手動で構成しようとすると、SQL1841N メッセージを受け取ります。

表 35. DB2 データ・ソースのサーバー・オプション (続き)

名前	説明
VARCHAR2_COMPAT	リモート・データ・ソースが VARCHAR2 互換であるかどうかを指定します。有効な値は Y と N です。デフォルト値は、リモート・データ・ソースによって異なります。リモート Oracle データ・ソースの場合、デフォルトは Y であり、データ・ソースは VARCHAR2 互換です。その他すべてのリモート・データ・ソースの場合、デフォルトは N であり、データ・ソースは VARCHAR2 互換モードに設定されません。 DB2 Database for Linux, UNIX, and Windows、ODBC、または JDBC データ・ソースが VARCHAR2 互換モードで構成されている場合、このサーバー・オプションを Y に設定する必要があります。 InfoSphere Federation Server バージョン 9.7 フィックスパック 2 以降では、CREATE SERVER ステートメントを実行すると、このサーバー・オプションはデータ・ソースの構成に基づいて自動的に構成されます。このサーバー・オプションを手動で構成しようとすると、SQL1841N メッセージを受け取ります。

ユーザー・マッピング・オプション

表 36. DB2 データ・ソースのユーザー・マッピング・オプション

オプション	説明
FED_PROXY_USER	インバウンド接続が非トラステッド接続である場合に、すべてのアウトバウンド・トラステッド接続を確立するために使用する許可 ID を指定します。このオプションで指定された ID を持つユーザーには、REMOTE_AUTHID と REMOTE_PASSWORD の両方を指定するユーザー・マッピングが必要です。FED_PROXY_USER ユーザー・マッピング・オプションを指定する場合は、FED_PROXY_USER サーバー・オプションも指定する必要があります。 **制約事項:** このサーバー・オプションは、DB2 Database for Linux, UNIX, and Windows バージョン 9.5 以降、および DB2 for z/OS バージョン 9 以降でのみ有効です。

表 36. DB2 データ・ソースのユーザー・マッピング・オプション (続き)

オプション	説明
ACCOUNTING_STRING	アカウンティング情報を受け渡す必要がある場合は必須。 DRDA 会計情報ストリングを指定します。有効な値は、255 文字以下のストリングです。
REMOTE_AUTHID	ローカル・ユーザー ID のマップ先となるリモート・ユーザー ID を指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用する ID が使用されます。
REMOTE_PASSWORD	リモート・ユーザー ID のリモート・パスワードを指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用したパスワードが使用されます。
USE_TRUSTED_CONTEXT	ユーザー・マッピングがトラステッドかどうかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合ユーザー・マッピングはトラステッドとはならず、非トラステッドのフェデレーテッド・アウトバウンド接続でのみ使用できます。 Y が指定されている場合、ユーザー・マッピングはトラステッドであり、トラステッドと非トラステッドのどちらのアウトバウンド・フェデレーテッド接続でも使用できます。 **制約事項:** このサーバー・オプションは、DB2 Database for Linux, UNIX, and Windows バージョン 9.5 以降、および DB2 for z/OS バージョン 9 以降でのみ有効です。

列オプション

表 37. DB2 データ・ソースの列オプション

オプション	説明
NUMERIC_STRING	数値ストリングの処理方法を指定します。デフォルトは N です。データ・ソース・ストリング列に数値ストリングのみが含まれ、ブランクを含め他の文字が含まれていない場合、NUMERIC_STRING オプションを Y に設定します。列に対して NUMERIC_STRING が Y に設定されている場合、照会オプティマイザーは、列のデータをソートする場合に支障となり得るブランクがこの列には含まれないことを認識します。データ・ソースの照合シーケンスが、フェデレーテッド・サーバーが使用する照合シーケンスとは異なる場合に、このオプションを使用してください。このオプションを使用する列は、照合シーケンスが異なるためにリモートでの評価から除かれるということはありません。
NO_EMPTY_STRING	リモート・データ・ソース・サーバーに、空ストリングを格納できるかどうかを指定します。有効な値は Y と N です。デフォルト値は、ご使用のリモート・データ・ソースによって異なります。リモート Oracle データ・ソースの場合、デフォルトは 1 であり、すべての空ストリング値は NULL 値に変換されます。その他すべてのリモート・データ・ソースの場合、デフォルトは N であり、データ・ソースに空ストリングを格納できます。
XML_ROOT	XML シーケンスを参照する XML 列の値に追加する XML ルート要素を指定します。このオプションを指定した場合、XML 列の値が整形式 XML 文書であることが保証されます。

Excel オプション・リファレンス

フェデレーテッド・サーバーとそのユーザーがどのようにデータ・ソースと対話するかを構成するには、ラッパー、サーバー、およびニックネームの各オプションを設定および変更します。

ラッパー・オプション

次の表は、このデータ・ソースに適用されるオプションをリストし、指定する必要がある必須選択のオプションを示しています。

表 38. Excel のラッパー・オプション

名前	説明
DB2_FENCED	必須。ラッパーを fenced モードとトラステッド・モードのどちらで実行するかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合、ラッパーはトラステッド・モードで実行されます。

サーバー・オプション

表 39. Excel のサーバー・オプション

名前	説明
DB2_MAX_ASYNC_REQUESTS_PER_QUERY	照会からの並行非同期要求の最大数を指定します。有効な値は -1 から 64000 です。デフォルトは 1 です。 -1 を指定すると、要求の数はフェデレーテッド照会オプティマイザーが決定します。 0 を指定すると、データ・ソースは付加的な非同期要求を受け入れることができません。

ニックネーム・オプション

表 40. Excel のニックネーム・オプション

オプション	説明
FILE_PATH	必須。アクセスする Excel スプレッドシートの完全修飾ディレクトリー・パスおよびファイル名を指定します。
RANGE	使用するセルの範囲を指定します (例: A1:C100)。コロンの前の値は、範囲の左上隅のセルを指定します。コロンの後の値は、範囲の右下隅のセルを指定します。

Informix オプション・リファレンス

フェデレーテッド・サーバーおよびそのユーザーがデータ・ソースと対話する方法を構成するには、ラッパー、サーバー、ユーザー・マッピング、および列のオプションを設定および変更します。

ラッパー・オプション

次の表は、このデータ・ソースに適用されるオプションをリストし、指定する必要がある必須選択のオプションを示しています。

表 41. Informix のラッパー・オプション

名前	説明
DB2_FENCED	必須。ラッパーを fenced モードとトラステッド・モードのどちらで実行するかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合、ラッパーはトラステッド・モードで実行されます。
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「UserMappingRepositoryLDAP」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は、Java および C です。デフォルトは Java です。

サーバー・オプション

表 42. Informix のサーバー・オプション

名前	説明
COLLATING_SEQUENCE	データ・ソースがフェデレーテッド・データベースと同じデフォルト照合シーケンスを使用するかどうかを指定します。有効な値は Y、N、および I です。I は大/小文字を区別しないことを指定します。デフォルト値は Y です。フェデレーテッド・サーバーに指定された照合シーケンスは、リモート・データ・ソースの照合シーケンスと一致している必要があります。
COMM_RATE	フェデレーテッド・サーバーとデータ・ソース・サーバーとの間の通信速度を指定します (MB/秒)。有効な値は 0 より大きく、2147483648 より小さい整数です。デフォルトは 2 です。

表 42. *Informix* のサーバー・オプション (続き)

名前	説明
CPU_RATIO	データ・ソースの CPU がフェデレーテッド・サーバーの CPU と比較してどれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。 1 を設定すると、フェデレーテッド・サーバーとデータ・ソース・サーバーの CPU 速度が同じで、1:1 の比率であるということです。 0.5 を設定すると、フェデレーテッド・サーバーの CPU の速度がデータ・ソースの CPU の半分であることを示します。 2 を設定すると、フェデレーテッド CPU がデータ・ソース CPU の 2 倍高速であるということです。
DBNAME	必須。アクセスする Informix データベースの名前を指定します。
DB2_MAXIMAL_PUSHDOWN	照会オプティマイザーがアクセス・プランの選択に使用する 1 次基準を指定します。有効な値は Y および N です。デフォルトは N であり、その場合、照会オプティマイザーは見積コストが最小になるプランを選択します。 Y は、照会オプティマイザーが、照会操作を最も多くデータ・ソースにプッシュダウンするアクセス・プランを選択するということを指定します。この基準を満たすアクセス・プランが複数ある場合、最もコストの小さいプランが選択されます。
DB2_MAX_ASYNC_REQUESTS_ PER_QUERY	照会からの並行非同期要求の最大数を指定します。有効な値は -1 から 64000 です。デフォルトは 1 です。 -1 を指定すると、要求の数はフェデレーテッド照会オプティマイザーが決定します。 0 を指定すると、データ・ソースは付加的な非同期要求を受け入れることができません。
DB2_TWO_PHASE_COMMIT	フェデレーテッド・サーバーがデータ・ソースに 2 フェーズ・コミット・プロトコルと 1 フェーズ・コミット・プロトコルのどちらで接続するかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合、フェデレーテッド・サーバーは 1 フェーズ・コミット・プロトコルを使用して接続します。 Y を指定すると、フェデレーテッド・サーバーは 2 フェーズ・コミット・プロトコルを使用して接続します。

表 42. Informix のサーバー・オプション (続き)

名前	説明
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「UserMappingRepositoryLDAP」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は、Java および C です。デフォルトは Java です。
FOLD_ID	データ・ソースに送信されるユーザー ID で大/小文字のどちらを使用するかを指定します。デフォルト値はありません。フェデレーテッド・サーバーは、ユーザー ID を大文字で送信し、この ID が失敗した場合は、ユーザー ID を小文字で送信します。有効な値は U (大文字)、L (小文字)、および N (NULL) です。NULL に設定すると、パフォーマンスが低下する場合があるため、NULL 値の使用は避けてください。
FOLD_PW	データ・ソースに送信されるパスワードで大/小文字のどちらを使用するかを指定します。デフォルト値はありません。フェデレーテッド・サーバーは、パスワードを大文字で送信し、このパスワードが失敗した場合は、パスワードを小文字で送信します。有効な値は U (大文字)、L (小文字)、および N (NULL) です。NULL に設定すると、パフォーマンスが低下する場合があるため、NULL 値の使用は避けてください。
INFORMIX_CLIENT_LOCALE	フェデレーテッド・サーバーとデータ・ソース・サーバー間の接続に使用する CLIENT_LOCALE を指定します。この値は、任意の有効な Informix ロケールです。このオプションを指定しない場合、CLIENT_LOCALE 環境変数は、db2dj.ini ファイルの中で指定されている値に設定されます。db2dj.ini で CLIENT_LOCALE 環境変数が指定されていない場合、INFORMIX_CLIENT_LOCALE は、フェデレーテッド・データベースのコード・ページとテリトリーに最も近い Informix ロケールに設定されます。

表 42. *Informix* のサーバー・オプション *(続き)*

名前	説明
INFORMIX_DB_LOCALE	フェデレーテッド・サーバーとデータ・ソース・サーバーとの間の接続に使用する Informix DB_LOCALE を指定します。INFORMIX_DB_LOCALE オプションを指定しない場合、Informix DB_LOCALE 環境変数は、db2dj.ini ファイルで指定された値に設定されます。db2dj.ini ファイルに値が指定されていない場合、Informix DB_LOCALE 環境変数は設定されません。

INFORMIX_LOCK_MODE	Informix データ・ソースに設定するロック・モードを指定します。Informix ラッパーは、Informix データ・ソースに接続した直後に、SET LOCK MODE コマンドを発行します。有効な値は W、N、および数値です。デフォルトは W であり、その場合ラッパーは、ロックの解放を無制限に待機します。 N が指定されている場合は待機せず、ただちにエラーが戻されます。待機時間の最大値の秒数を表す数値を使用して指定してください。デッドロックまたはタイムアウトが発生した場合には、ALTER SERVER ステートメントを使用して INFORMIX_LOCK_MODE オプションの値を変更します。例:
	`ALTER SERVER TYPE informix`
	`VERSION 9`
	`WRAPPER informix`
	`OPTIONS`
	`(ADD informix_lock_mode '60')`

IO_RATIO	データ・ソース入出力システムがフェデレーテッド・サーバー入出力システムと比較して、どれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。 1 を設定すると、フェデレーテッド・サーバーとデータ・ソース・サーバーの入出力速度が同じで、1:1 の比率であるということです。 0.5 を設定すると、フェデレーテッド・サーバーの速度がデータ・ソースの速度の半分であることを示します。 2 を設定すると、フェデレーテッドの速度がデータ・ソースの速度の 2 倍高速であるということです。

表 42. Informix のサーバー・オプション (続き)

名前	説明
IUD_APP_SVPT_ENFORCE	フェデレーテッド・サーバーでアプリケーション・セーブポイント・ステートメントの使用が必須かどうかを指定します。有効な値は Y と N です。デフォルトは Y であり、この場合、データ・ソースでアプリケーション・セーブポイント・ステートメントが必須ではない状態で挿入、更新、または削除の操作中にエラーが発生した場合には、フェデレーテッド・サーバーはトランザクションをロールバックし、SQL エラー・コード SQL1476N が戻されます。デフォルト設定を使用することが推奨されています。
NODE	必須。データ・ソースをそのリレーショナル・データベース管理システムに対してインスタンスとして定義する名前を指定します。
OLD_NAME_GEN	データ・ソースの列名と索引名を、フェデレーテッド・サーバーのニックネーム列名とローカル索引名に変換する方法を指定します。有効な値は Y と N です。デフォルトは N であり、その場合、生成される名前とデータ・ソース中の名前には密接な対応関係があります。 Y を指定すると、生成される名前は IBM WebSphere Federation Server バージョン 9 以前で作成された名前と同じになります。したがって、それらの名前はデータ・ソースの名前と密接に対応しないことがあります。
PUSHDOWN	フェデレーテッド・サーバーが、データ・ソースによる操作の評価を許可するかどうかを指定します。有効な値は Y および N です。デフォルトは Y であり、その場合、データ・ソースは操作を評価します。 N は、列名を指定した SELECT のみを含む SQL ステートメントをフェデレーテッド・サーバーが送信するということを指定します。述部 (WHERE= など)、列およびスカラー関数 (MAX や MIN など)、ソート (ORDER BY または GROUP BY など)、および結合は、フェデレーテッド・サーバーがデータ・ソースに送信するどの SQL にも含まれません。

ユーザー・マッピング・オプション

表 43. Informix のユーザー・マッピング・オプション

名前	説明
REMOTE_AUTHID	ローカル・ユーザー ID のマップ先となるリモート・ユーザー ID を指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用する ID が使用されます。
REMOTE_PASSWORD	リモート・ユーザー ID のリモート・パスワードを指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用したパスワードが使用されます。

列オプション

表 44. Informix の列オプション

名前	説明
BINARY_REP	特定のバイナリー・データ・タイプの列を識別します。 BINARY_REP 列オプションの値を Y に設定すると、フェデレーテッド・サーバーに SQL_BINARY データ・タイプのプッシュダウンを強制することができます。リモート・データ・ソースで照会オプティマイザーがバイナリー形式の列の比較を実行できるようにすると、照会パフォーマンスを改善できます。
NUMERIC_STRING	数値ストリングの処理方法を指定します。デフォルトは N です。データ・ソース・ストリング列に数値ストリングのみが含まれ、ブランクを含め他の文字が含まれていない場合、NUMERIC_STRING オプションを Y に設定します。列に対して NUMERIC_STRING が Y に設定されている場合、照会オプティマイザーは、列のデータをソートする場合に支障となり得るブランクがこの列には含まれないことを認識します。データ・ソースの照合シーケンスが、フェデレーテッド・サーバーが使用する照合シーケンスとは異なる場合に、このオプションを使用してください。このオプションを使用する列は、照合シーケンスが異なるためにリモートでの評価から除かれるということはありません。

JDBC オプション・リファレンス

フェデレーテッド・サーバーおよびそのユーザーがデータ・ソースと対話する方法を構成するには、ラッパー、サーバー、ユーザー・マッピング、および列のオプションを設定および変更します。

ラッパー・オプション

次の表は、このデータ・ソースに適用されるオプションをリストし、指定する必要がある必須選択のオプションを示しています。

表 45. JDBC のラッパー・オプション

名前	説明
DB2_FENCED	必須。ラッパーを fenced モードとトラステッド・モードのどちらで実行するかを指定します。DB2 サーバーは fenced モードでの JVM のロードのみをサポートしているため、有効な値は Y だけです。デフォルトは Y であり、ラッパーは fenced モードで実行されます。
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「UserMappingRepositoryLDAP」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は、Java および C です。デフォルトは Java です。

サーバー・オプション

表 46. JDBC のサーバー・オプション

名前	説明
COLLATING_SEQUENCE	データ・ソースがフェデレーテッド・データベースと同じデフォルト照合シーケンスを使用するかどうかを指定します。有効な値は Y、N、および I です。I は大/小文字を区別しないことを指定します。デフォルト値は Y です。フェデレーテッド・サーバーに指定された照合シーケンスは、リモート・データ・ソースの照合シーケンスと一致している必要があります。

表 46. JDBC のサーバー・オプション (続き)

名前	説明
COMM_RATE	フェデレーテッド・サーバーとデータ・ソース・サーバーとの間の通信速度を指定します (MB/秒)。有効な値は 0 より大きく、2147483648 より小さい整数です。デフォルトは 2 です。
CPU_RATIO	データ・ソースの CPU がフェデレーテッド・サーバーの CPU と比較してどれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。 1 を設定すると、フェデレーテッド・サーバーとデータ・ソース・サーバーの CPU 速度が同じで、1:1 の比率であるということです。 0.5 を設定すると、フェデレーテッド・サーバーの CPU の速度がデータ・ソースの CPU の半分であることを示します。 2 を設定すると、フェデレーテッド CPU がデータ・ソース CPU の 2 倍高速であるということです。
DATEFORMAT	データ・ソースが使用する日付形式を指定します。日付形式は、'DD'、'MM'、および 'YY' または 'YYYY' を使用して指定します。区切り文字にはスペース、ハイフン、コンマなどを指定できます。例えば `'YYYY-MM-DD'` という形式は、日付を 1958-10-01 などとして指定します。値には NULL 値を含めることができます。
DB2_MAXIMAL_PUSHDOWN	照会オプティマイザーがアクセス・プランの選択に使用する 1 次基準を指定します。有効な値は Y および N です。デフォルトは N であり、その場合、照会オプティマイザーは見積コストが最小になるプランを選択します。 Y は、照会オプティマイザーが、照会操作を最も多くデータ・ソースにプッシュダウンするアクセス・プランを選択するということを指定します。
DB2_MAX_ASYNC_REQUESTS_PER_QUERY	照会からの並行非同期要求の最大数を指定します。有効な値は -1 から 64000 です。デフォルトは 0 です。-1 を指定すると、要求の数はフェデレーテッド照会オプティマイザーが決定します。 0 を指定すると、データ・ソースは付加的な非同期要求を受け入れることができません。

表 46. JDBC のサーバー・オプション (続き)

名前	説明
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「UserMappingRepositoryLDAP」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は、Java および C です。デフォルトは Java です。
DRIVER_CLASS	JDBC ドライバー・ライブラリーを指定します。 JDBC ドライバーが JDBC 仕様のバージョン 3.0 以降に準拠している場合、サーバーを複数の JDBC データ・ソースに対して登録できます。 JDBC 仕様と、DRIVER_CLASS サーバー・オプションの設定方法については、JDBC ドライバーの資料を参照してください。 **例** 以下の例では、com.ibm.db2.jcc.DB2Driver JDBC ドライバー・ライブラリーが指定されています。 `DRIVER_CLASS 'com.ibm.db2.jcc.DB2Driver'` **重要:** このオプションを指定する場合は、URL サーバー・オプションも指定する必要があります。
DRIVER_PACKAGE	JDBC ドライバー・パッケージを指定します。複数のドライバー・クラス・パッケージは、パス分離文字を使用して指定します。Windows オペレーティング・システムではセミコロンを、 Linux および Unix オペレーティング・システムではコロンを使用してください。 **例** 以下は、Linux オペレーティング・システムで複数のドライバー・パッケージをコロンを使用して指定する例です。 `DRIVER_PACKAGE '/path1/file1.jar: /path2/file2.jar'`

表 46. JDBC のサーバー・オプション (続き)

名前	説明
FOLD_ID	データ・ソースに送信されるユーザー ID で大/小文字のどちらを使用するかを指定します。デフォルト値はありません。フェデレーテッド・サーバーは、ユーザー ID を大文字で送信し、この ID が失敗した場合は、ユーザー ID を小文字で送信します。有効な値は U (大文字)、L (小文字)、および N (NULL) です。NULL に設定すると、パフォーマンスが低下する場合があるため、NULL 値の使用は避けてください。
FOLD_PW	データ・ソースに送信されるパスワードで大/小文字のどちらを使用するかを指定します。デフォルト値はありません。フェデレーテッド・サーバーは、パスワードを大文字で送信し、このパスワードが失敗した場合は、パスワードを小文字で送信します。有効な値は U (大文字)、L (小文字)、および N (NULL) です。NULL に設定すると、パフォーマンスが低下する場合があるため、NULL 値の使用は避けてください。
IO_RATIO	データ・ソース入出力システムがフェデレーテッド・サーバー入出力システムと比較して、どれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。 1 を設定すると、フェデレーテッド・サーバーとデータ・ソース・サーバーの入出力速度が同じで、1:1 の比率であるということです。 0.5 を設定すると、フェデレーテッド・サーバーの速度がデータ・ソースの速度の半分であることを示します。 2 を設定すると、フェデレーテッドの速度がデータ・ソースの速度の 2 倍高速であるということです。

表 46. *JDBC* のサーバー・オプション *(続き)*

名前	説明
JDBC_LOG	JDBC ラッパーで、エラー・トレース用のログ・ファイルを作成するかどうかを指定します。有効な値は Y および N です。デフォルトは N であり、ログ・ファイルは作成されません。このサーバー・オプションを Y に設定すると、JDBC ラッパーは JDBC ログ・ファイルを `jdbc_wrapper_prod_id.log` ファイル (ここで、 *prod_id* は製品 ID) に書き込みます。このログ・ファイルは、DB2 データベース・マネージャー構成パラメーター DIAGPATH によって指定されるディレクトリー内に保管されます。 UNIX システムのデフォルト・ディレクトリーは、`inst_home/sqllib/db2dump` です。 **推奨:** このサーバー・オプションを YES に設定するとシステムのパフォーマンスに影響を与えるため、実動システムではロギングを有効にしないことをお勧めします。
OLD_NAME_GEN	データ・ソースの列名と索引名を、フェデレーテッド・サーバーのニックネーム列名とローカル索引名に変換する方法を指定します。有効な値は Y と N です。デフォルトは N であり、その場合、生成される名前とデータ・ソース中の名前には密接な対応関係があります。 Y を指定すると、生成される名前は IBM WebSphere Federation Server バージョン 9 以前で作成された名前と同じになります。したがって、それらの名前はデータ・ソースの名前と密接に対応しないことがあります。
PUSHDOWN	フェデレーテッド・サーバーが、データ・ソースによる操作の評価を許可するかどうかを指定します。有効な値は Y および N です。デフォルトは Y であり、その場合、データ・ソースは操作を評価します。 N は、列名を指定した SELECT のみを含む SQL ステートメントをフェデレーテッド・サーバーが送信するということを指定します。述部 (WHERE= など)、列およびスカラー関数 (MAX や MIN など)、ソート (ORDER BY または GROUP BY など)、および結合は、フェデレーテッド・サーバーがデータ・ソースに送信するどの SQL にも含まれません。

表 46. JDBC のサーバー・オプション (続き)

名前	説明
TIMEFORMAT	データ・ソースが使用する時刻形式を指定します。 'hh12'、'hh24'、'mm'、'ss'、'AM'、および 'A.M' を使用して、時刻形式を指定します。例えば、'hh24:mm:22' という形式は、時刻を 16:00:00 などとして指定します。'hh12:mm:ss AM' という形式は、時刻を 8:00:00 AM などとして指定します。値には NULL 値を含めることができます。
TIMESTAMPFORMAT	データ・ソースが使用するタイム・スタンプ形式を指定します。有効な値は、DATEFORMAT オプションおよび TIMEFORMAT オプションで使用される形式です。 10 分の 1 秒の場合は 'n'、100 分の 1 秒の場合は 'nn'、1000 分の 1 秒 (ミリ秒) の場合は 'nnn' を指定します。最大でマイクロ秒の 'nnnnnn' まで可能です。例えば、'YYY-MM-DD-hh24:mm:ss.nnnnnn' という形式は、タイム・スタンプを 1994-01-01-24:00:00.000000 などとして指定します。値には NULL 値を含めることができます。
URL	リモート・サーバーの JDBC 接続ストリングを指定します。 JDBC 接続ストリングは次の 3 つの部分から構成され、各部の間はいずれもコロンで区切られます。 • データベース・プロトコル • データベース・タイプ名、または接続ドライバー名 • 別名またはサブネームを使用したデータベース ID **例** 以下の例では、JDBC 接続ストリングは jdbc:db2://cn.ibm.com:50471/testdb です。 URL 'jdbc:db2://cn.ibm.com:50471/testdb' JDBC ドライバーが JDBC 仕様のバージョン 3.0 以降に準拠している場合、サーバーを複数の JDBC データ・ソースに対して登録できます。 JDBC 仕様と、URL サーバー・オプションの設定方法については、JDBC ドライバーの資料を参照してください。 **重要:** このオプションを指定する場合は、DRIVER_CLASS サーバー・オプションも指定する必要があります。

表 46. JDBC のサーバー・オプション (続き)

名前	説明
VARCHAR_NO_TRAILING_BLANKS	データ・ソースが、末尾ブランク文字を少なくとも 1 つ含む VARCHAR 列を持つかどうかを指定します。デフォルトは N です。この場合、VARCHAR 列には、末尾ブランク文字が少なくとも 1 つ含まれます。

ユーザー・マッピング・オプション

表 47. JDBC のユーザー・マッピング・オプション

名前	説明
REMOTE_AUTHID	ローカル・ユーザー ID のマップ先となるリモート・ユーザー ID を指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用する ID が使用されます。
REMOTE_PASSWORD	リモート・ユーザー ID のリモート・パスワードを指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用したパスワードが使用されます。

列オプション

表 48. JDBC の列オプション

名前	説明
NUMERIC_STRING	列にブランクを含む数字のストリングが含まれているかどうかを指定します。有効な値は Y と N です。デフォルトは N です。この場合、列には、ブランクを含む数字のストリングが含まれません。後ろに末尾ブランクが続く数字のストリングだけが列に含まれる場合は、Y と指定しないでください。列に対して NUMERIC_STRING を Y に設定すると、列のデータをソートする場合に支障となり得るブランクがこの列には含まれないことを、照会オプティマイザーが認識します。データ・ソースの照合シーケンスが、フェデレーテッド・サーバーが使用する照合シーケンスとは異なる場合に、このオプションを使用してください。このオプションを使用する列は、照合シーケンスが異なるためにリモートでの評価から除かれるということはありません。
VARCHAR_NO_TRAILING_BLANKS	VARCHAR 列に末尾ブランクが少なくとも 1 つあるかどうかを指定します。

Microsoft SQL Server オプション・リファレンス

フェデレーテッド・サーバーおよびそのユーザーがデータ・ソースと対話する方法を構成するには、ラッパー、サーバー、ユーザー・マッピング、および列のオプションを設定および変更します。

ラッパー・オプション

以下の表に、このデータ・ソースに適用されるオプションと、指定する必要のある必須指定のオプションをまとめました。

表 49. Microsoft SQL Server のラッパー・オプション

名前	説明
DB2_FENCED	必須。ラッパーを fenced モードとトラステッド・モードのどちらで実行するかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合、ラッパーはトラステッド・モードで実行されます。
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「`UserMappingRepositoryLDAP`」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は、Java および C です。デフォルトは Java です。

サーバー・オプション

表 50. Microsoft SQL Server のサーバー・オプション

名前	説明
CODEPAGE	データ・ソースのクライアント構成のコード化文字セットに対応するコード・ページを指定します。非 Unicode のフェデレーテッド・データベースを使用する UNIX システムおよび Microsoft Windows システムでは、デフォルトはフェデレーテッド・データベースのコード・ページです。 Unicode のフェデレーテッド・データベースを使用する UNIX システムでは、デフォルトは 1208 です。Unicode のフェデレーテッド・データベースを使用する Windows システムでは、デフォルトは 1202 です。

表 50. *Microsoft SQL Server* のサーバー・オプション (続き)

名前	説明
COLLATING_SEQUENCE	データ・ソースがフェデレーテッド・データベースと同じデフォルト照合シーケンスを使用するかどうかを指定します。有効な値は Y、N、および I です。I は大/小文字を区別しないことを指定します。デフォルト値は Y です。フェデレーテッド・サーバーに指定された照合シーケンスは、リモート・データ・ソースの照合シーケンスと一致している必要があります。
COMM_RATE	データ・ソースの CPU がフェデレーテッド・サーバーの CPU と比較してどれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。 1 を設定すると、フェデレーテッド・サーバーとデータ・ソース・サーバーの CPU 速度が同じで、1:1 の比率であるということです。 0.5 を設定すると、フェデレーテッド・サーバーの CPU の速度がデータ・ソースの CPU の半分であることを示します。 2 を設定すると、フェデレーテッド CPU がデータ・ソース CPU の 2 倍高速であるということです。
CPU_RATIO	データ・ソースの CPU の実行速度がフェデレーテッド・サーバーの CPU の速度と比較してどれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。
DBNAME	必須。アクセスしようとするデータベースの別名を指定します。この値は大文字小文字が区別されます。
DB2_MAXIMAL_PUSHDOWN	照会オプティマイザーがアクセス・プランの選択に使用する 1 次基準を指定します。有効な値は Y および N です。デフォルトは N であり、その場合、照会オプティマイザーは見積コストが最小になるプランを選択します。 Y は、照会オプティマイザーが、照会操作を最も多くデータ・ソースにプッシュダウンするアクセス・プランを選択するということを指定します。この基準を満たすアクセス・プランが複数ある場合、最もコストの小さいプランが選択されます。

表 50. Microsoft SQL Server のサーバー・オプション (続き)

名前	説明
DB2_MAX_ASYNC_REQUESTS_PER_QUERY	照会からの並行非同期要求の最大数を指定します。有効な値は -1 から 64000 です。デフォルトは 1 です。 -1 を指定すると、要求の数はフェデレーテッド照会オプティマイザーが決定します。 0 を指定すると、データ・ソースは付加的な非同期要求を受け入れることができません。
DB2_TWO_PHASE_COMMIT	フェデレーテッド・サーバーがデータ・ソースに 2 フェーズ・コミット・プロトコルと 1 フェーズ・コミット・プロトコルのどちらで接続するかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合、フェデレーテッド・サーバーは 1 フェーズ・コミット・プロトコルを使用して接続します。 Y を指定すると、フェデレーテッド・サーバーは 2 フェーズ・コミット・プロトコルを使用して接続します。 **重要:** このオプションを Y に設定した場合は、XA_OPEN_STRING_OPTION も指定する必要があります。
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「`UserMappingRepositoryLDAP`」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は、Java および C です。デフォルトは Java です。
FOLD_ID	データ・ソースに送信されるユーザー ID で大/小文字のどちらを使用するかを指定します。デフォルト値はありません。フェデレーテッド・サーバーは、ユーザー ID を大文字で送信し、この ID が失敗した場合は、ユーザー ID を小文字で送信します。有効な値は U (大文字)、L (小文字)、および N (NULL) です。NULL に設定すると、パフォーマンスが低下する場合があるため、NULL 値の使用は避けてください。

表 50. *Microsoft SQL Server* のサーバー・オプション *(続き)*

名前	説明
FOLD_PW	データ・ソースに送信されるパスワードで大/小文字のどちらを使用するかを指定します。デフォルト値はありません。フェデレーテッド・サーバーは、パスワードを大文字で送信し、このパスワードが失敗した場合は、パスワードを小文字で送信します。有効な値は U (大文字)、L (小文字)、および N (NULL) です。NULL に設定すると、パフォーマンスが低下する場合があるため、NULL 値の使用は避けてください。
IO_RATIO	データ・ソース入出力システムがフェデレーテッド・サーバー入出力システムと比較して、どれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。 1 を設定すると、フェデレーテッド・サーバーとデータ・ソース・サーバーの入出力速度が同じで、1:1 の比率であるということです。 0.5 を設定すると、フェデレーテッド・サーバーの速度がデータ・ソースの速度の半分であることを示します。 2 を設定すると、フェデレーテッドの速度がデータ・ソースの速度の 2 倍高速であるということです。
NODE	必須。フェデレーテッド・サーバーが Microsoft Windows を使用している場合、NODE の値は、Microsoft SQL Server 用に指定したシステムの DSN 名になります。フェデレーテッド・サーバーが UNIX または Linux を使用している場合、NODE の値は `odbc.ini` ファイルで定義されます。この値は大文字小文字が区別されます。
OLD_NAME_GEN	データ・ソースの列名と索引名を、フェデレーテッド・サーバーのニックネーム列名とローカル索引名に変換する方法を指定します。有効な値は Y と N です。デフォルトは N であり、その場合、生成される名前とデータ・ソース中の名前には密接な対応関係があります。 Y を指定すると、生成される名前は IBM WebSphere Federation Server バージョン 9 以前で作成された名前と同じになります。したがって、それらの名前はデータ・ソースの名前と密接に対応しないことがあります。

表 50. *Microsoft SQL Server* のサーバー・オプション *(続き)*

名前	説明
PASSWORD	パスワードがデータ・ソースに送信されるかどうかを指定します。デフォルト値は Y です。
PUSHDOWN	フェデレーテッド・サーバーが、データ・ソースによる操作の評価を許可するかどうかを指定します。有効な値は Y および N です。デフォルトは Y であり、その場合、データ・ソースは操作を評価します。 N は、列名を指定した SELECT のみを含む SQL ステートメントをフェデレーテッド・サーバーが送信するということを指定します。述部 (WHERE= など)、列およびスカラー関数 (MAX や MIN など)、ソート (ORDER BY または GROUP BY など)、および結合は、フェデレーテッド・サーバーがデータ・ソースに送信するどの SQL にも含まれません。
XA_OPEN_STRING_OPTIONS	DB2_TWO_PHASE_COMMIT が Y に設定されている場合に必須です。Microsoft SQL Server レジストリーのリソース・マネージャー ID を指定します。

ユーザー・マッピング・オプション

表 51. *Microsoft SQL Server* のユーザー・マッピング・オプション

名前	説明
REMOTE_AUTHID	ローカル・ユーザー ID のマップ先となるリモート・ユーザー ID を指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用する ID が使用されます。
REMOTE_PASSWORD	リモート・ユーザー ID のリモート・パスワードを指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用したパスワードが使用されます。

列オプション

表 52. Microsoft SQL Server の列オプション

名前	説明
BINARY_REP	特定のバイナリー・データ・タイプの列を識別します。 BINARY_REP 列オプションの値を Y に設定すると、フェデレーテッド・サーバーに SQL_BINARY データ・タイプのプッシュダウンを強制することができます。リモート・データ・ソースで照会オプティマイザーがバイナリー形式の列の比較を実行できるようにすると、照会パフォーマンスを改善できます。
NUMERIC_STRING	数値ストリングの処理方法を指定します。デフォルトは N です。データ・ソース・ストリング列に数値ストリングのみが含まれ、ブランクを含め他の文字が含まれていない場合、NUMERIC_STRING オプションを Y に設定します。列に対して NUMERIC_STRING が Y に設定されている場合、照会オプティマイザーは、列のデータをソートする場合に支障となり得るブランクがこの列には含まれないことを認識します。データ・ソースの照合シーケンスが、フェデレーテッド・サーバーが使用する照合シーケンスとは異なる場合に、このオプションを使用してください。このオプションを使用する列は、照合シーケンスが異なるためにリモートでの評価から除かれるということはありません。

ODBC オプション・リファレンス

フェデレーテッド・サーバーおよびそのユーザーがデータ・ソースと対話する方法を構成するには、ラッパー、サーバー、ユーザー・マッピング、および列のオプションを設定および変更します。

ラッパー・オプション

次の表は、このデータ・ソースに適用されるオプションをリストし、指定する必要がある必須選択のオプションを示しています。

表 53. ODBC のラッパー・オプション

名前	説明
DB2_FENCED	必須。ラッパーを fenced モードとトラステッド・モードのどちらで実行するかを指定します。有効な値は Y と N です。デフォルトは N です。この場合、ラッパーはトラステッド・モードで実行されます。 **重要:** UNIX システムでこのオプションを Y に設定した場合は、DB2_SOURCE_CLIENT_MODE ラッパー・オプションも設定する必要があります。
DB2_SOURCE_CLIENT_MODE	データ・ソースのクライアントが 32 ビットであり、フェデレーテッド・サーバー上のデータベース・インスタンスが 64 ビットであることを指定します。有効値は 32 ビットのみです。このオプションは、UNIX のみで有効です。 **重要:** このオプションを設定する場合、DB2_FENCED ラッパー・オプションを Y に設定する必要もあります。
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「`UserMappingRepositoryLDAP`」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は、Java および C です。デフォルトは Java です。
MODULE	UNIX システム上で稼働するフェデレーテッド・サーバーの場合は必須です。ODBC Driver Manager インプリメンテーションまたは SQL/CLI インプリメンテーションを含むライブラリーの絶対パスを指定します。 UNIX の場合、デフォルトはありません。 Microsoft Windows システムの場合、デフォルトは `odbc32.dll` です。

サーバー・オプション

表 54. ODBC のサーバー・オプション

名前	説明
CODEPAGE	データ・ソースのクライアント構成のコード化文字セットに対応するコード・ページを指定します。非 Unicode のフェデレーテッド・データベースを使用する UNIX システムおよび Windows システムでは、デフォルトはフェデレーテッド・データベースが使用するコード・ページです。Unicode のフェデレーテッド・データベースを使用する UNIX システムでは、デフォルトは 1208 です。Unicode のフェデレーテッド・データベースを使用する Windows システムでは、デフォルトは 1202 です。
COLLATING_SEQUENCE	データ・ソースがフェデレーテッド・データベースと同じデフォルト照合シーケンスを使用するかどうかを指定します。有効な値は Y、N、および I です。I は大/小文字を区別しないことを指定します。デフォルト値は Y です。フェデレーテッド・サーバーに指定された照合シーケンスは、リモート・データ・ソースの照合シーケンスと一致している必要があります。
COMM_RATE	フェデレーテッド・サーバーとデータ・ソース・サーバーとの間の通信速度を指定します (MB/秒)。有効な値は 0 より大きく、2147483648 より小さい整数です。デフォルトは 2 です。
CPU_RATIO	データ・ソースの CPU がフェデレーテッド・サーバーの CPU と比較してどれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。 1 を設定すると、フェデレーテッド・サーバーとデータ・ソース・サーバーの CPU 速度が同じで、1:1 の比率であるということです。 0.5 を設定すると、フェデレーテッド・サーバーの CPU の速度がデータ・ソースの CPU の半分であることを示します。 2 を設定すると、フェデレーテッド CPU がデータ・ソース CPU の 2 倍高速であるということです。

表 54. ODBC のサーバー・オプション (続き)

名前	説明
DATEFORMAT	データ・ソースが使用する日付形式を指定します。日付形式は、'DD'、'MM'、および 'YY' または 'YYYY' を使用して指定します。区切り文字にはスペース、ハイフン、コンマなどを指定できます。例えば `'YYYY-MM-DD'` という形式は、日付を 1958-10-01 などとして指定します。値には NULL 値を含めることができます。 値を単一引用符で囲んで指定する場合、その単一引用符を保持する必要があります。単一引用符を保持するには、追加の単一引用符 `''` を指定します。これらは二重引用符ではなく、単一引用符を 2 つ重ねたものです。例えば `'''YYYY-MM-DD'''` のように指定します。
DBNAME	アクセスしようとするデータ・ソース・データベースの名前を指定します。
DB2_AUTHID_QUOTE_CHAR	スキーマ名やユーザー名など authid 名に使用する引用文字を指定します。このオプションを指定しない場合は、二重引用符がデフォルトで使用されます。
DB2_ID_QUOTE_CHAR	カラム名など区切り ID に使用する引用文字を指定します。このオプションを指定しない場合は、二重引用符がデフォルトで使用されます。
DB2_MAX_ASYNC_REQUESTS_PER_QUERY	照会からの並行非同期要求の最大数を指定します。有効な値は -1 から 64000 です。デフォルトは 0 です。-1 を指定すると、要求の数はフェデレーテッド照会オプティマイザーが決定します。 0 を指定すると、データ・ソースは付加的な非同期要求を受け入れることができません。
DB2_MAXIMAL_PUSHDOWN	照会オプティマイザーがアクセス・プランの選択に使用する 1 次基準を指定します。有効な値は Y および N です。デフォルトは N であり、その場合、照会オプティマイザーは見積コストが最小になるプランを選択します。 Y は、照会オプティマイザーが、照会操作を最も多くデータ・ソースにプッシュダウンするアクセス・プランを選択するということを指定します。
DB2_TABLE_QUOTE_CHAR	表名に使用する引用文字を指定します。このオプションを指定しない場合は、二重引用符がデフォルトで使用されます。

表 54. ODBC のサーバー・オプション (続き)

名前	説明
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「UserMappingRepositoryLDAP」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は、Java および C です。デフォルトは Java です。
ENABLE_BULK_INSERT	バルク挿入処理を Netezza データ・ソースに対して有効にするかどうかを指定します。有効な値は Y と N です。デフォルトは N です。
FOLD_ID	データ・ソースに送信されるユーザー ID で大/小文字のどちらを使用するかを指定します。デフォルト値はありません。フェデレーテッド・サーバーは、ユーザー ID を大文字で送信し、この ID が失敗した場合は、ユーザー ID を小文字で送信します。有効な値は U (大文字)、L (小文字)、および N (NULL) です。NULL に設定すると、パフォーマンスが低下する場合があるため、NULL 値の使用は避けてください。
FOLD_PW	データ・ソースに送信されるパスワードで大/小文字のどちらを使用するかを指定します。デフォルト値はありません。フェデレーテッド・サーバーは、パスワードを大文字で送信し、このパスワードが失敗した場合は、パスワードを小文字で送信します。有効な値は U (大文字)、L (小文字)、および N (NULL) です。NULL に設定すると、パフォーマンスが低下する場合があるため、NULL 値の使用は避けてください。

表 54. ODBC のサーバー・オプション (続き)

名前	説明
IO_RATIO	データ・ソース入出力システムがフェデレーテッド・サーバー入出力システムと比較して、どれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。 1 を設定すると、フェデレーテッド・サーバーとデータ・ソース・サーバーの入出力速度が同じで、1:1 の比率であるということです。 0.5 を設定すると、フェデレーテッド・サーバーの速度がデータ・ソースの速度の半分であることを示します。 2 を設定すると、フェデレーテッドの速度がデータ・ソースの速度の 2 倍高速であるということです。
NODE	必須。DSN の定義時に、ODBC データ・ソースに割り当てられるノード名またはシステム DSN 名を指定します。この値は大文字小文字が区別されます。
OLD_NAME_GEN	データ・ソースの列名と索引名を、フェデレーテッド・サーバーのニックネーム列名とローカル索引名に変換する方法を指定します。有効な値は Y と N です。デフォルトは N であり、その場合、生成される名前とデータ・ソース中の名前には密接な対応関係があります。 Y を指定すると、生成される名前は IBM WebSphere Federation Server バージョン 9 以前で作成された名前と同じになります。したがって、それらの名前はデータ・ソースの名前と密接に対応しないことがあります。
PUSHDOWN	フェデレーテッド・サーバーが、データ・ソースによる操作の評価を許可するかどうかを指定します。有効な値は Y および N です。デフォルトは Y であり、その場合、データ・ソースは操作を評価します。 N は、列名を指定した SELECT のみを含む SQL ステートメントをフェデレーテッド・サーバーが送信するということを指定します。述部 (WHERE= など)、列およびスカラー関数 (MAX や MIN など)、ソート (ORDER BY または GROUP BY など)、および結合は、フェデレーテッド・サーバーがデータ・ソースに送信するどの SQL にも含まれません。

表 54. ODBC のサーバー・オプション (続き)

名前	説明
TIMEFORMAT	データ・ソースが使用する時刻形式を指定します。 'hh12'、'hh24'、'mm'、'ss'、'AM'、および 'A.M' を使用して、時刻形式を指定します。例えば、'hh24:mm:22' という形式は、時刻を 16:00:00 などとして指定します。'hh12:mm:ss AM' という形式は、時刻を 8:00:00 AM などとして指定します。値には NULL 値を含めることができます。
TIMESTAMPFORMAT	データ・ソースが使用するタイム・スタンプ形式を指定します。有効な値は、DATEFORMAT オプションおよび TIMEFORMAT オプションで使用される形式です。 10 分の 1 秒の場合は 'n'、100 分の 1 秒の場合は 'nn'、1000 分の 1 秒 (ミリ秒) の場合は 'nnn' を指定します。最大でマイクロ秒の 'nnnnnn' まで可能です。例えば、'YYY-MM-DD-hh24:mm:ss.nnnnnn' という形式は、タイム・スタンプを 1994-01-01-24:00:00.000000 などとして指定します。値には NULL 値を含めることができます。
VARCHAR_NO_TRAILING_BLANKS	データ・ソースが、末尾ブランク文字を少なくとも 1 つ含む VARCHAR 列を持つかどうかを指定します。デフォルトは N です。この場合、VARCHAR 列には、末尾ブランク文字が少なくとも 1 つ含まれます。

ユーザー・マッピング・オプション

表 55. ODBC のユーザー・マッピング・オプション

名前	説明
REMOTE_AUTHID	ローカル・ユーザー ID のマップ先となるリモート・ユーザー ID を指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用する ID が使用されます。
REMOTE_PASSWORD	リモート・ユーザー ID のリモート・パスワードを指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用したパスワードが使用されます。

列オプション

表 56. ODBC の列オプション

名前	説明
NUMERIC_STRING	列にブランクを含む数字のストリングが含まれているかどうかを指定します。有効な値は Y と N です。デフォルトは N です。この場合、列には、ブランクを含む数字のストリングが含まれません。後ろに末尾ブランクが続く数字のストリングだけが列に含まれる場合は、Y と指定しないでください。列に対して NUMERIC_STRING を Y に設定すると、列のデータをソートする場合に支障となり得るブランクがこの列には含まれないことを、照会オプティマイザーが認識します。データ・ソースの照合シーケンスが、フェデレーテッド・サーバーが使用する照合シーケンスとは異なる場合に、このオプションを使用してください。このオプションを使用する列は、照合シーケンスが異なるためにリモートでの評価から除かれるということはありません。
VARCHAR_NO_TRAILING_BLANKS	VARCHAR 列に末尾ブランクが少なくとも 1 つあるかどうかを指定します。

Oracle オプション・リファレンス

フェデレーテッド・サーバーおよびそのユーザーがデータ・ソースと対話する方法を構成するには、ラッパー、サーバー、ユーザー・マッピング、および列のオプションを設定および変更します。

ラッパー・オプション

次の表は、このデータ・ソースに適用されるオプションをリストし、指定する必要がある必須選択のオプションを示しています。

表 57. Oracle のラッパー・オプション

名前	説明
DB2_FENCED	必須。ラッパーを fenced モードとトラステッド・モードのどちらで実行するかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合、ラッパーはトラステッド・モードで実行されます。

表 57. *Oracle* のラッパー・オプション (続き)

名前	説明
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「UserMappingRepositoryLDAP」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は、Java および C です。デフォルトは Java です。

サーバー・オプション

表 58. *Oracle* のサーバー・オプション

名前	説明
COLLATING_SEQUENCE	データ・ソースがフェデレーテッド・データベースと同じデフォルト照合シーケンスを使用するかどうかを指定します。有効な値は Y、N、および I です。I は大/小文字を区別しないことを指定します。デフォルト値は Y です。フェデレーテッド・サーバーに指定された照合シーケンスは、リモート・データ・ソースの照合シーケンスと一致している必要があります。
COMM_RATE	フェデレーテッド・サーバーとデータ・ソース・サーバーとの間の通信速度を指定します (MB/秒)。有効な値は 0 より大きく、2147483648 より小さい整数です。デフォルトは 2 です。
CPU_RATIO	データ・ソースの CPU がフェデレーテッド・サーバーの CPU と比較してどれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。 1 を設定すると、フェデレーテッド・サーバーとデータ・ソース・サーバーの CPU 速度が同じで、1:1 の比率であるということです。 0.5 を設定すると、フェデレーテッド・サーバーの CPU の速度がデータ・ソースの CPU の半分であることを示します。 2 を設定すると、フェデレーテッド CPU がデータ・ソース CPU の 2 倍高速であるということです。

表 58. *Oracle* のサーバー・オプション *(続き)*

名前	説明
DB2_MAXIMAL_PUSHDOWN	照会オプティマイザーがアクセス・プランの選択に使用する 1 次基準を指定します。有効な値は Y および N です。デフォルトは N であり、その場合、照会オプティマイザーは見積コストが最小になるプランを選択します。 Y は、照会オプティマイザーが、照会操作を最も多くデータ・ソースにプッシュダウンするアクセス・プランを選択するということを指定します。
DB2_MAX_ASYNC_REQUESTS_PER_QUERY	照会からの並行非同期要求の最大数を指定します。有効な値は -1 から 64000 です。デフォルトは 1 です。 -1 を指定すると、要求の数はフェデレーテッド照会オプティマイザーが決定します。 0 を指定すると、データ・ソースは付加的な非同期要求を受け入れることができません。
DB2_TWO_PHASE_COMMIT	フェデレーテッド・サーバーがデータ・ソースに 2 フェーズ・コミット・プロトコルと 1 フェーズ・コミット・プロトコルのどちらで接続するかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合、フェデレーテッド・サーバーは 1 フェーズ・コミット・プロトコルを使用して接続します。 Y を指定すると、フェデレーテッド・サーバーは 2 フェーズ・コミット・プロトコルを使用して接続します。
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「`UserMappingRepositoryLDAP`」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は、Java および C です。デフォルトは Java です。
FED_PROXY_USER	インバウンド接続が非トラステッド接続である場合に、すべてのアウトバウンド・トラステッド接続を確立するために使用する許可 ID を指定します。 **重要:** このオプションで指定された ID を持つユーザーには、REMOTE_AUTHID と REMOTE_PASSWORD の両方を指定するユーザー・マッピングが必要です。

表 58. Oracle のサーバー・オプション (続き)

名前	説明
FOLD_ID	データ・ソースに送信されるユーザー ID で大/小文字のどちらを使用するかを指定します。デフォルト値はありません。フェデレーテッド・サーバーは、ユーザー ID を大文字で送信し、この ID が失敗した場合は、ユーザー ID を小文字で送信します。有効な値は U (大文字)、L (小文字)、および N (NULL) です。NULL に設定すると、パフォーマンスが低下する場合があるため、NULL 値の使用は避けてください。
FOLD_PW	データ・ソースに送信されるパスワードで大/小文字のどちらを使用するかを指定します。デフォルト値はありません。フェデレーテッド・サーバーは、パスワードを大文字で送信し、このパスワードが失敗した場合は、パスワードを小文字で送信します。有効な値は U (大文字)、L (小文字)、および N (NULL) です。NULL に設定すると、パフォーマンスが低下する場合があるため、NULL 値の使用は避けてください。
IO_RATIO	データ・ソース入出力システムがフェデレーテッド・サーバー入出力システムと比較して、どれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。 1 を設定すると、フェデレーテッド・サーバーとデータ・ソース・サーバーの入出力速度が同じで、1:1 の比率であるということです。 0.5 を設定すると、フェデレーテッド・サーバーの速度がデータ・ソースの速度の半分であることを示します。 2 を設定すると、フェデレーテッドの速度がデータ・ソースの速度の 2 倍高速であるということです。
NODE	必須。Oracle データベース・サーバーが置かれているノードの名前を指定します。ノードの名前は、`tnsnames.ora` ファイルから取得します。

表 58. *Oracle* のサーバー・オプション *(続き)*

名前	説明
OLD_NAME_GEN	データ・ソースの列名と索引名を、フェデレーテッド・サーバーのニックネーム列名とローカル索引名に変換する方法を指定します。有効な値は Y と N です。デフォルトは N であり、その場合、生成される名前とデータ・ソース中の名前には密接な対応関係があります。 Y を指定すると、生成される名前は IBM WebSphere Federation Server バージョン 9 以前で作成された名前と同じになります。したがって、それらの名前はデータ・ソースの名前と密接に対応しないことがあります。
PLAN_HINTS	プラン・ヒントを使用可能にするかどうかを指定します。プラン・ヒントはステートメントの一部分であり、データ・ソース・オプティマイザーが照会のパフォーマンスを向上させるために使用する追加情報を提供します。データ・ソース・オプティマイザーはこのプラン・ヒントを使用して、索引を使用するかどうか、およびどの索引を使用するか、またはどの表結合順序を使用するかを決定します。有効な値は Y および N です。デフォルトは N であり、プラン・ヒントは使用可能になりません。 Y は、データ・ソースがプラン・ヒントをサポートしている場合に、データ・ソースでプラン・ヒントを使用可能にするということを指定します。
PUSHDOWN	フェデレーテッド・サーバーが、データ・ソースによる操作の評価を許可するかどうかを指定します。有効な値は Y および N です。デフォルトは Y であり、フェデレーテッド・サーバーはデータ・ソースによる操作の評価を許可します。 N は、フェデレーテッド・サーバーがリモート・データ・ソースから列を取り出すということを指定します。
VARCHAR_NO_TRAILING_BLANKS	特定のサーバーについて、VARCHAR 列に末尾ブランクが含まれるかどうかを指定します。このオプションを 1 つの列に適用するには、VARCHAR_NO_TRAILING_BLANKS 列オプションを使用します。
XA_OPEN_STRING_OPTIONS	ストリングに付加する追加情報を指定します。例えば、この情報はトレース・ファイルのディレクトリーなどです。

ユーザー・マッピング・オプション

表 59. Oracle のユーザー・マッピング・オプション

名前	説明
FED_PROXY_USER	インバウンド接続が非トラステッド接続である場合に、すべてのアウトバウンド・トラステッド接続を確立するために使用する許可 ID を指定します。 **重要:** このオプションで指定された ID を持つユーザーには、REMOTE_AUTHID と REMOTE_PASSWORD の両方を指定するユーザー・マッピングが必要です。 FED_PROXY_USER ユーザー・マッピング・オプションを指定する場合は、FED_PROXY_USER サーバー・オプションも指定する必要があります。
REMOTE_AUTHID	ローカル・ユーザー ID のマップ先となるリモート・ユーザー ID を指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用する ID が使用されます。
REMOTE_PASSWORD	リモート・ユーザー ID のリモート・パスワードを指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用したパスワードが使用されます。
USE_TRUSTED_CONTEXT	ユーザー・マッピングがトラステッドかどうかを指定します。有効な値は Y と N です。デフォルトは N です。この場合、ユーザー・マッピングはトラステッドではなく、非トラステッド・アウトバウンド接続でしか使用できません。 Y は、ユーザー・マッピングがトラステッドであり、トラステッドおよび非トラステッド両方のアウトバウンド接続で使用できるということを指定します。

列オプション

表 60. Oracle の列オプション

名前	説明
NUMERIC_STRING	列にブランクを含む数字のストリングが含まれているかどうかを指定します。有効な値は Y と N です。デフォルトは N です。この場合、列には、ブランクを含む数字のストリングが含まれません。後ろに末尾ブランクが続く数字のストリングだけが列に含まれる場合は、Y と指定しないでください。列に対して NUMERIC_STRING を Y に設定すると、列のデータをソートする場合に支障となり得るブランクがこの列には含まれないことを、照会オプティマイザーが認識します。データ・ソースの照合シーケンスが、フェデレーテッド・サーバーが使用する照合シーケンスとは異なる場合に、このオプションを使用してください。このオプションを使用する列は、照合シーケンスが異なるためにリモートでの評価から除かれるということはありません。
VARCHAR_NO_TRAILING_BLANKS	列に末尾ブランクが含まれるかどうかを指定します。

スクリプト・オプション・リファレンス

フェデレーテッド・サーバーとそのユーザーがどのようにデータ・ソースと対話するかを構成するには、ラッパー、サーバー、ユーザー・マッピング、ニックネーム、および列の各オプションを設定および変更します。

ラッパー・オプション

次の表は、このデータ・ソースに適用されるオプションをリストし、指定する必要がある必須選択のオプションを示しています。

表 61. スクリプトのラッパー・オプション

名前	説明
DB2_FENCED	必須。ラッパーを fenced モードとトラステッド・モードのどちらで実行するかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合、ラッパーはトラステッド・モードで実行されます。
PROXY_TYPE	フェデレーテッド・サーバーがファイアウォールの背後にある場合に、インターネットへのアクセスに使用するプロキシー・タイプを指定します。有効な値は NONE および SOCKS です。デフォルト値は NONE です。

表 61. スクリプトのラッパー・オプション (続き)

名前	説明
PROXY_SERVER_NAME	プロキシー・サーバーの名前または IP アドレスを指定します。有効な値は、1 から 32760 までの 10 進数のポート番号、またはサービス名です。 IPv6 形式は、IPv6 を構成する場合のみ使用します。
PROXY_SERVER_PORT	プロキシー・サーバー上のプロキシー・サービスのポートまたはサービス名を指定します。有効な値は、10 進数の 1 から 32760 までのポート番号か、サービス名です。

サーバー・オプション

表 62. スクリプトのサーバー・オプション

名前	説明
DAEMON_PORT	SCRIPT デーモンが SCRIPT ジョブ要求を listen するときに使用するポート番号を指定します。デフォルトは 4099 です。このポート番号は、デーモン構成ファイルの DAEMON_PORT オプションと同じでなければなりません。サービス名が SCRIPT デーモン用に構成されている場合は、TCP/IP サービス名を使用します。
DB2_MAX_ASYNC_REQUESTS_PER_QUERY	照会からの並行非同期要求の最大数を指定します。有効な値は -1 から 64000 です。デフォルトは 1 です。 -1 を指定すると、要求の数はフェデレーテッド照会オプティマイザーが決定します。 0 を指定すると、データ・ソースは付加的な非同期要求を受け入れることができません。
NODE	必須。SCRIPT デーモンが実行されるシステムの DNS ホスト名または IP アドレスを指定します。有効な IP アドレスは IPv4 (ドット区切り) 形式または IPv6 (コロン区切り) 形式です。 IPv6 形式は、IPv6 を構成する場合のみ使用します。
PROXY_AUTHID	プロキシー・サーバー認証のユーザー名を指定します。
PROXY_PASSWORD	プロキシー・サーバー認証のパスワードを指定します。
PROXY_SERVER_NAME	プロキシー・サーバーの名前または IP アドレスを指定します。有効な値は、1 から 32760 までの 10 進数のポート番号、またはサービス名です。 IPv6 形式は、IPv6 を構成する場合のみ使用します。

表 62. スクリプトのサーバー・オプション (続き)

名前	説明
PROXY_SERVER_PORT	プロキシー・サーバー上のプロキシー・サービスのポートまたはサービス名を指定します。有効な値は、10 進数の 1 から 32760 までのポート番号か、サービス名です。
PROXY_TYPE	フェデレーテッド・サーバーがファイアウォールの背後にある場合に、インターネットへのアクセスに使用するプロキシー・タイプを指定します。有効な値は NONE および SOCKS です。デフォルト値は NONE です。

ユーザー・マッピング・オプション

表 63. スクリプトのユーザー・マッピング・オプション

名前	説明
PROXY_AUTHID	プロキシー・サーバー認証のユーザー名を指定します。
PROXY_PASSWORD	プロキシー・サーバー認証のパスワードを指定します。このパスワードは、フェデレーテッド・データベース・カタログに保管されるときに暗号化されます。

ニックネーム・オプション

表 64. スクリプトのニックネーム・オプション

名前	説明
DATASOURCE	ルート・ニックネームの場合は必須です。呼び出すスクリプトの名前を指定します。このオプションの値として指定するスクリプトは、SCRIPT デーモン構成ファイルでも指定されていなければなりません。 **重要:** このオプションは、ルート・ニックネームのみで有効です。

NAMESPACES	各列の XPATH および TEMPLATE オプションで使用される、ネーム・スペース接頭部に関連付けられているネーム・スペースを指定します。次の構文を使用します。
	`NAMESPACES'prefix1= "actual_namespace1", prefix2="actual_namespace2"'`
	複数のネーム・スペースを区切るには、コンマを使用します。例:
	`NAMESPACES='http://www.myweb.com/cust", i='http://www.myweb.com/cust/id", n='http://www.myweb.com/cust/name"'`

表 64. スクリプトのニックネーム・オプション (続き)

名前	説明
STREAMING	ソース文書を処理するために、論理フラグメントに分ける必要があるかどうかを指定します。このフラグメントは、ニックネームの XPath 式に一致するノードに対応しています。このラッパーは、ソース・データの構文解析と処理をフラグメントごとに行います。このタイプの構文解析では、メモリーの使用量が最小限度で済みます。有効な値は Y と N です。デフォルトは N です。この場合、文書は構文解析されません。このオプションは、ルート・ニックネームのみで有効です。STREAMING と VALIDATE の両方のオプションを Y に設定することは避けてください。
TIMEOUT	デーモンからの結果を待つ最大時間 (分単位) を指定します。デフォルトは 60 です。このオプションは、ルート・ニックネームのみで有効です。
VALIDATE	ソース文書からデータが抽出される前に、その文書が XML スキーマまたは文書タイプ定義 (DTD) に準拠しているかを検証するかどうかを指定します。デフォルトは N です。この場合、検証は行われません。この値を Y に設定する前に、ソース文書が指定する場所にスキーマ・ファイルまたは DTD ファイルを用意します。このオプションは、ルート・ニックネームのみで有効です。 STREAMING と VALIDATE の両方のオプションを Y に設定することは避けてください。
XPATH	個々のタプルを表す XML エレメントを識別する XPath 式を指定します。子ニックネームの XPATH ニックネーム・オプションは、その親の XPATH ニックネーム・オプションによって指定されるパスのコンテキストで評価されます。この XPath 式は、XPath 列オプションが存在することによって識別される列値を評価するためのコンテキストとして使用されます。

列オプション

表 65. スクリプトの列オプション

名前	説明
DEFAULT	スクリプト入力列のデフォルト値を指定します。 SQL 照会で値が指定されない場合は、このデフォルト値が使用されます。

表 65. スクリプトの列オプション (続き)

名前	説明
FOREIGN_KEY	このニックネームが子ニックネームであることを示し、対応する親ニックネームの名前を指定します。ニックネームには、FOREIGN_KEY 列オプションを 1 つしか指定できません。このオプションの値は、大文字小文字を区別します。この列には XPATH オプションを指定しないでください。列は、親のニックネームと子のニックネームを結合するためだけに使用できます。親ニックネームに別のスキーマ名が付けられている場合、FOREIGN_KEY オプションを含む CREATE NICKNAME ステートメントは失敗します。FOREIGN_KEY 文節で参照されるニックネームが CREATE NICKNAME ステートメントで小文字または大/小文字混合として明示的に定義されているのでない限り、このニックネームを FOREIGN_KEY 文節で参照するときは大文字で指定する必要があります。このオプションが列に設定されている場合、その列に他のオプションを設定することはできません。
INPUT_MODE	列の入力モードを指定します。有効な値は、CONFIG および FILE_INPUT です。CONFIG は値を列の入力モードとして処理します。 FILE_INPUT は、値を保管するファイルを指定します。指定した値は、ラッパーによって SCRIPT デーモンに渡されます。
POSITION	定位置パラメーターの整数値を指定します。定位置値を整数に設定すると、この入力はコマンド行のその位置になければなりません。このオプションを設定すると、照会の実行時に該当位置にスイッチが挿入されます。POSITION を -1 に設定すると、オプションはコマンド行の最後のオプションとして追加されます。 POSITION 整数値を同じニックネームの中で 2 回使用することはできません。このオプションは、入力列にのみ適用されます。

表 65. スクリプトの列オプション (続き)

名前	説明
PRIMARY_KEY	1 つ以上の子ニックネームを持つ親ニックネームの場合に必須です。このニックネームを親ニックネームとするということを指定します。列データ・タイプは VARCHAR(16) でなければなりません。ニックネームには、PRIMARY_KEY 列オプションを 1 つしか指定できません。 Yes が唯一の有効な値です。この列には XPATH オプションを指定しないでください。列は、親のニックネームと子のニックネームを結合するためだけに使用できます。このオプションが列に設定されている場合、その列に他のオプションを設定することはできません。
SWITCH	コマンド行のスクリプトのフラグを指定します。このオプションの値は、WSSCRIPT.ARGS またはデフォルト値 (存在する場合) によって指定される列の値の前に置かれます。このオプションに値を指定せず、列にデフォルト値が存在する場合には、スイッチ情報なしでデフォルト値が追加されます。このオプションは、入力列の場合に必須です。
SWITCH_ONLY	コマンド行引数なしでのスイッチの使用を可能にします。このオプションは Y に設定します。有効な入力値は Y および N です。入力値が Y の場合、コマンド行にスイッチのみが追加されます。入力値が N の場合、コマンド行に値は追加されません。
VALID_VALUES	列の有効な値のセットを指定します。複数の値を区切るには、セミコロンを使用します。
XPATH	XML 文書の中でこの列に対応するデータを含む Xpath 式を指定します。 CREATE NICKNAME ステートメントが XPATH ニックネーム・オプションから XPath 式を適用した後で、ラッパーはその XPath 式を評価します。

Sybase オプション・リファレンス

フェデレーテッド・サーバーおよびそのユーザーがデータ・ソースと対話する方法を構成するには、ラッパー、サーバー、ユーザー・マッピング、および列のオプションを設定および変更します。

ラッパー・オプション

次の表は、このデータ・ソースに適用されるオプションをリストし、指定する必要がある必須選択のオプションを示しています。

表 66. *Sybase* のラッパー・オプション

名前	説明
DB2_FENCED	必須。ラッパーを fenced モードとトラステッド・モードのどちらで実行するかを指定します。Microsoft Windows では、有効な値は Y と N です。デフォルトは N です。この場合、ラッパーはトラステッド・モードで実行されます。 UNIX では、デフォルトで、かつ唯一の有効な値は Y です。ラッパーは、fenced モードで実行しなければなりません。
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「`UserMappingRepositoryLDAP`」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は Java および C です。デフォルトは Java です。

サーバー・オプション

表 67. *Sybase* のサーバー・オプション

名前	説明
COLLATING_SEQUENCE	データ・ソースがフェデレーテッド・データベースと同じデフォルト照合シーケンスを使用するかどうかを指定します。有効な値は Y、N、および I です。I は大/小文字を区別しないことを指定します。デフォルト値は Y です。フェデレーテッド・サーバーに指定された照合シーケンスは、リモート・データ・ソースの照合シーケンスと一致している必要があります。
COMM_RATE	フェデレーテッド・サーバーとデータ・ソース・サーバーとの間の通信速度を指定します (MB/秒)。有効な値は 0 より大きく、2147483648 より小さい整数です。デフォルトは 2 です。

表 67. *Sybase* のサーバー・オプション (続き)

名前	説明
CPU_RATIO	データ・ソースの CPU がフェデレーテッド・サーバーの CPU と比較してどれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。 1 を設定すると、フェデレーテッド・サーバーとデータ・ソース・サーバーの CPU 速度が同じで、1:1 の比率であるということです。 0.5 を設定すると、フェデレーテッド・サーバーの CPU の速度がデータ・ソースの CPU の半分であることを示します。 2 を設定すると、フェデレーテッド CPU がデータ・ソース CPU の 2 倍高速であるということです。
CONV_EMPTY_STRING	フェデレーテッド・サーバーが、レプリケーション・タスク中に空ストリングをスペースに変換するかどうかを指定します。有効な値は Y と N です。デフォルトは N です。この場合、フェデレーテッド・サーバーは空ストリングを変換しません。空ストリングが保管される非 NULL 可能文字列をデータ・ソースが持っているときは、このオプションを Y に設定してください。
DBNAME	必須。アクセスしようとするデータベースの名前を指定します。データベースの名前は、Sybase サーバーから取得します。
DB2_ID_QUOTE_CHAR	カラム名など区切り ID に使用する引用文字を指定します。このオプションを指定しない場合は、二重引用符がデフォルトで使用されます。
DB2_MAXIMAL_PUSHDOWN	照会オプティマイザーがアクセス・プランの選択に使用する 1 次基準を指定します。有効な値は Y および N です。デフォルトは N であり、その場合、照会オプティマイザーは見積コストが最小になるプランを選択します。 Y は、照会オプティマイザーが、照会操作を最も多くデータ・ソースにプッシュダウンするアクセス・プランを選択するということを指定します。
DB2_MAX_ASYNC_REQUESTS_PER_QUERY	照会からの並行非同期要求の最大数を指定します。有効な値は -1 から 64000 です。デフォルトは 1 です。 -1 を指定すると、要求の数はフェデレーテッド照会オプティマイザーが決定します。 0 を指定すると、データ・ソースは付加的な非同期要求を受け入れることができません。

表 67. *Sybase* のサーバー・オプション *(続き)*

名前	説明
DB2_TWO_PHASE_COMMIT	フェデレーテッド・サーバーがデータ・ソースに 2 フェーズ・コミット・プロトコルと 1 フェーズ・コミット・プロトコルのどちらで接続するかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合、フェデレーテッド・サーバーは 1 フェーズ・コミット・プロトコルを使用して接続します。 Y を指定すると、フェデレーテッド・サーバーは 2 フェーズ・コミット・プロトコルを使用して接続します。
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「`UserMappingRepositoryLDAP`」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は Java および C です。デフォルトは Java です。
FOLD_ID	データ・ソースに送信されるユーザー ID で大/小文字のどちらを使用するかを指定します。デフォルト値はありません。フェデレーテッド・サーバーは、ユーザー ID を大文字で送信し、この ID が失敗した場合は、ユーザー ID を小文字で送信します。有効な値は U (大文字)、L (小文字)、および N (NULL) です。NULL に設定すると、パフォーマンスが低下する場合があるため、NULL 値の使用は避けてください。
FOLD_PW	データ・ソースに送信されるパスワードで大/小文字のどちらを使用するかを指定します。デフォルト値はありません。フェデレーテッド・サーバーは、パスワードを大文字で送信し、このパスワードが失敗した場合は、パスワードを小文字で送信します。有効な値は U (大文字)、L (小文字)、および N (NULL) です。NULL に設定すると、パフォーマンスが低下する場合があるため、NULL 値の使用は避けてください。

表 67. *Sybase* のサーバー・オプション (続き)

名前	説明
IFILE	デフォルトのインターフェース・ファイルの代わりに使用する Sybase インターフェース・ファイルのパスと名前を指定します。Sybase ラッパーは、インターフェース・ファイルを、次の場所から指定された順序で検索します。Microsoft Windows の場合は、IFILE サーバー・オプション、次に %DB2PATH%¥interfaces ディレクトリー、最後に %SYBASE%¥ini¥sql.ini ディレクトリー。UNIX の場合は、IFILE サーバー・オプション、次に sqllib/interfaces ディレクトリー、最後に $SYBASE/interfaces ディレクトリー。
IO_RATIO	データ・ソース入出力システムがフェデレーテッド・サーバー入出力システムと比較して、どれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。 1 を設定すると、フェデレーテッド・サーバーとデータ・ソース・サーバーの入出力速度が同じで、1:1 の比率であるということです。 0.5 を設定すると、フェデレーテッド・サーバーの速度がデータ・ソースの速度の半分であることを示します。 2 を設定すると、フェデレーテッドの速度がデータ・ソースの速度の 2 倍高速であるということです。
LOGIN_TIMEOUT	フェデレーテッド・サーバーがログイン要求を中止する前に待機する時間 (秒単位) を指定します。デフォルトは 0 です。この場合、フェデレーテッド・サーバーが待機する時間は無制限です。
NODE	必須。Sybase サーバーが置かれているノードの名前を指定します。ノード名は Sybase インターフェース・ファイルにあります。

表 67. *Sybase* のサーバー・オプション *(続き)*

名前	説明
OLD_NAME_GEN	データ・ソースの列名と索引名を、フェデレーテッド・サーバーのニックネーム列名とローカル索引名に変換する方法を指定します。有効な値は Y と N です。デフォルトは N であり、その場合、生成される名前とデータ・ソース中の名前には密接な対応関係があります。 Y を指定すると、生成される名前は IBM WebSphere Federation Server バージョン 9 以前で作成された名前と同じになります。したがって、それらの名前はデータ・ソースの名前と密接に対応しないことがあります。
PACKET_SIZE	クライアント・ライブラリーが表データ・ストリーム (TDS) パケットの送信に使用するパケット・サイズ (バイト単位) を指定します。Sybase ラッパーが大量のテキスト・データおよびイメージ・データを送信または受信する必要がある場合は、PACKET_SIZE を大きくします。
PLAN_HINTS	プラン・ヒントを使用可能にするかどうかを指定します。プラン・ヒントはステートメントの一部分であり、データ・ソース・オプティマイザーが照会のパフォーマンスを向上させるために使用する追加情報を提供します。データ・ソース・オプティマイザーはこのプラン・ヒントを使用して、索引を使用するかどうか、およびどの索引を使用するか、またはどの表結合順序を使用するかを決定します。有効な値は Y および N です。デフォルトは N であり、プラン・ヒントは使用可能になりません。 Y は、データ・ソースがプラン・ヒントをサポートしている場合に、データ・ソースでプラン・ヒントを使用可能にするということを指定します。
PUSHDOWN	フェデレーテッド・サーバーが、データ・ソースによる操作の評価を許可するかどうかを指定します。有効な値は Y および N です。デフォルトは Y であり、その場合、データ・ソースは操作を評価します。 N は、列名を指定した SELECT のみを含む SQL ステートメントをフェデレーテッド・サーバーが送信するということを指定します。述部 (WHERE= など)、列およびスカラー関数 (MAX や MIN など)、ソート (ORDER BY または GROUP BY など)、および結合は、フェデレーテッド・サーバーがデータ・ソースに送信するどの SQL にも含まれません。

表 67. *Sybase* のサーバー・オプション (続き)

名前	説明
SERVER_PRINCIPAL_NAME	プリンシパル名がノード名と異なる場合は、Sybase サーバーのプリンシパル名を指定します。デフォルト値はノード名です。
TIMEOUT	リモート・サーバーがコマンドに応答するのをフェデレーテッド・サーバーが待つ最大時間 (秒単位) を指定します。デフォルトは 0 です。これは時間の制限がないということを指定します。
XA_OPEN_STRING_OPTIONS	Sybase DTM XA インターフェースのオープン・ストリングを指定します。これらのストリングは、LRM 名、ユーザー名、およびパスワードに追加されるものです。

ユーザー・マッピング・オプション

表 68. *Sybase* のユーザー・マッピング・オプション

オプション	説明
ENABLE_KERBEROS_CONNECTION	Kerberos セキュリティー・メカニズムを指定します。デフォルト値 N を指定すると、デフォルトのセキュリティー・メカニズムが使用されます。値 Y を指定すると、Kerberos セキュリティー・メカニズムが有効になります。
REMOTE_AUTHID	ローカル・ユーザー ID のマップ先となるリモート・ユーザー ID を指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用する ID が使用されます。
REMOTE_PASSWORD	リモート・ユーザー ID のリモート・パスワードを指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用したパスワードが使用されます。

列オプション

表 69. Sybase の列オプション

オプション	説明
NUMERIC_STRING	数値ストリングの処理方法を指定します。デフォルトは N です。データ・ソース・ストリング列に数値ストリングのみが含まれ、ブランクを含め他の文字が含まれていない場合、NUMERIC_STRING オプションを Y に設定します。列に対して NUMERIC_STRING が Y に設定されている場合、照会オプティマイザーは、列のデータをソートする場合に支障となり得るブランクがこの列には含まれないことを認識します。データ・ソースの照合シーケンスが、フェデレーテッド・サーバーが使用する照合シーケンスとは異なる場合に、このオプションを使用してください。このオプションを使用する列は、照合シーケンスが異なるためにリモートでの評価から除かれるということはありません。

Teradata オプション・リファレンス

フェデレーテッド・サーバーおよびそのユーザーがデータ・ソースと対話する方法を構成するには、ラッパー、サーバー、ユーザー・マッピング、および列のオプションを設定および変更します。

ラッパー・オプション

次の表は、このデータ・ソースに適用されるオプションをリストし、指定する必要がある必須選択のオプションを示しています。

表 70. Teradata のラッパー・オプション

名前	説明
DB2_FENCED	必須。ラッパーを fenced モードとトラステッド・モードのどちらで実行するかを指定します。Microsoft Windows では、有効な値は Y と N です。デフォルトは N です。この場合、ラッパーはトラステッド・モードで実行されます。 UNIX では、デフォルト値は Y です。この場合、ラッパーは、fenced モードで実行しなければなりません。

表 70. Teradata のラッパー・オプション (続き)

名前	説明
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「UserMappingRepositoryLDAP」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は、Java および C です。デフォルトは Java です。

サーバー・オプション

表 71. Teradata のサーバー・オプション

名前	説明
COLLATING_SEQUENCE	データ・ソースがフェデレーテッド・データベースと同じデフォルト照合シーケンスを使用するかどうかを指定します。有効な値は Y、N、および I です。I は大/小文字を区別しないことを指定します。デフォルト値は Y です。フェデレーテッド・サーバーに指定された照合シーケンスは、リモート・データ・ソースの照合シーケンスと一致している必要があります。
COMM_RATE	フェデレーテッド・サーバーとデータ・ソース・サーバーとの間の通信速度を指定します (MB/秒)。有効な値は 0 より大きく、2147483648 より小さい整数です。デフォルトは 2 です。
CPU_RATIO	データ・ソースの CPU がフェデレーテッド・サーバーの CPU と比較してどれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。 1 を設定すると、フェデレーテッド・サーバーとデータ・ソース・サーバーの CPU 速度が同じで、1:1 の比率であるということです。 0.5 を設定すると、フェデレーテッド・サーバーの CPU の速度がデータ・ソースの CPU の半分であることを示します。 2 を設定すると、フェデレーテッド CPU がデータ・ソース CPU の 2 倍高速であるということです。

表 71. Teradata のサーバー・オプション (続き)

名前	説明
DB2_MAXIMAL_PUSHDOWN	照会オプティマイザーがアクセス・プランの選択に使用する 1 次基準を指定します。有効な値は Y および N です。デフォルトは N であり、その場合、照会オプティマイザーは見積コストが最小になるプランを選択します。 Y は、照会オプティマイザーが、照会操作を最も多くデータ・ソースにプッシュダウンするアクセス・プランを選択するということを指定します。
DB2_MAX_ASYNC_REQUESTS_PER_QUERY	照会からの並行非同期要求の最大数を指定します。有効な値は -1 から 64000 です。デフォルトは 1 です。 -1 を指定すると、要求の数はフェデレーテッド照会オプティマイザーが決定します。 0 を指定すると、データ・ソースは付加的な非同期要求を受け入れることができません。
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「`UserMappingRepositoryLDAP`」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は、Java および C です。デフォルトは Java です。
IO_RATIO	データ・ソース入出力システムがフェデレーテッド・サーバー入出力システムと比較して、どれほど速いかまたは遅いかを指定します。有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。デフォルトは 1.0 です。値は、任意の有効な倍精度表記 (例えば 123E10、123、または 1.21E4) で表現できます。 1 を設定すると、フェデレーテッド・サーバーとデータ・ソース・サーバーの入出力速度が同じで、1:1 の比率であるということです。 0.5 を設定すると、フェデレーテッド・サーバーの速度がデータ・ソースの速度の半分であることを示します。 2 を設定すると、フェデレーテッドの速度がデータ・ソースの速度の 2 倍高速であるということです。
NODE	必須。Teradata サーバーの別名または IP アドレスを指定します。

表 71. Teradata のサーバー・オプション (続き)

名前	説明
OLD_NAME_GEN	データ・ソースの列名と索引名を、フェデレーテッド・サーバーのニックネーム列名とローカル索引名に変換する方法を指定します。有効な値は Y と N です。デフォルトは N であり、その場合、生成される名前とデータ・ソース中の名前には密接な対応関係があります。 Y を指定すると、生成される名前は IBM WebSphere Federation Server バージョン 9 以前で作成された名前と同じになります。したがって、それらの名前はデータ・ソースの名前と密接に対応しないことがあります。
PUSHDOWN	フェデレーテッド・サーバーが、データ・ソースによる操作の評価を許可するかどうかを指定します。有効な値は Y および N です。デフォルトは Y であり、その場合、データ・ソースは操作を評価します。 N は、列名を指定した SELECT のみを含む SQL ステートメントをフェデレーテッド・サーバーが送信するということを指定します。述部 (WHERE= など)、列およびスカラー関数 (MAX や MIN など)、ソート (ORDER BY または GROUP BY など)、および結合は、フェデレーテッド・サーバーがデータ・ソースに送信するどの SQL にも含まれません。

ユーザー・マッピング・オプション

表 72. Teradata のユーザー・マッピング・オプション

名前	説明
REMOTE_AUTHID	ローカル・ユーザー ID のマップ先となるリモート・ユーザー ID を指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用する ID が使用されます。
REMOTE_PASSWORD	リモート・ユーザー ID のリモート・パスワードを指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用したパスワードが使用されます。

列オプション

表 73. Teradata の列オプション

名前	説明
NUMERIC_STRING	数値ストリングの処理方法を指定します。デフォルトは N です。データ・ソース・ストリング列に数値ストリングのみが含まれ、ブランクを含め他の文字が含まれていない場合、NUMERIC_STRING オプションを Y に設定します。列に対して NUMERIC_STRING が Y に設定されている場合、照会オプティマイザーは、列のデータをソートする場合に支障となり得るブランクがこの列には含まれないことを認識します。データ・ソースの照合シーケンスが、フェデレーテッド・サーバーが使用する照合シーケンスとは異なる場合に、このオプションを使用してください。このオプションを使用する列は、照合シーケンスが異なるためにリモートでの評価から除かれるということはありません。

表構造ファイル・オプション・リファレンス

フェデレーテッド・サーバーとそのユーザーがどのようにデータ・ソースと対話するかを構成するには、ラッパー、サーバー、ニックネーム、および列の各オプションを設定および変更します。

次の表は、このデータ・ソースに適用されるオプションをリストし、指定する必要がある必須選択のオプションを示しています。

ラッパー・オプション

表 74. 表構造ファイルのラッパー・オプション

名前	説明
DB2_FENCED	必須。ラッパーを fenced モードとトラステッド・モードのどちらで実行するかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合、ラッパーはトラステッド・モードで実行されます。

サーバー・オプション

表 75. 表構造ファイルのサーバー・オプション

名前	説明
DB2_MAX_ASYNC_REQUESTS_PER_QUERY	照会からの並行非同期要求の最大数を指定します。有効な値は -1 から 64000 です。デフォルトは 1 です。 -1 を指定すると、要求の数はフェデレーテッド照会オプティマイザーが決定します。 0 を指定すると、データ・ソースは付加的な非同期要求を受け入れることができません。

ニックネーム・オプション

表 76. 表構造ファイルのニックネーム・オプション

名前	説明
COLUMN_DELIMITER	表構造ファイルの列を区切る文字として使用する単一文字を指定します。デフォルトはコンマ (,) です。単一引用符は区切り文字としては使用できません。列区切り文字は、ファイル全体で一貫性がなければなりません。NULL 値は 2 つの区切り文字が続くことによって表されます。 NULL フィールドが行の最後にある場合、区切り文字に行の終止符を続けることによって表されます。列区切り文字は、列の有効なデータとして存在することはできません。
CODEPAGE	データ・ソースのファイルのコード・ページを指定します。このオプションは、Unicode を使用するフェデレーテッド・データベースに対してのみ有効です。ソース・データは、指定されたコード・ページから Unicode に変換されます。
FILE_PATH	表構造ファイルまでの絶対パスを指定します。ファイル名は単一引用符で囲みます。データ・ファイルは標準ファイルまたはシンボリック・リンクでなければならず、パイプや別の非標準ファイル・タイプにすることはできません。 **重要:** FILE_PATH オプションを指定する場合、DOCUMENT 列を指定しないでください。

表 76. 表構造ファイルのニックネーム・オプション (続き)

名前	説明
KEY_COLUMN	ファイルをソートするときの基準となる列の名前を指定します。 DOCUMENT 列オプションがある列は、キー列にはできません。単一列のキーのみがサポートされています。値は、CREATE NICKNAME ステートメントに定義されている列の名前にしなければなりません。列は必ず昇順でソートされます。NOT NULL オプションをニックネーム・ステートメントの中の列の定義に追加することにより、キー列が NULL 可能ではないということを指定しなければなりません。この値は大文字小文字の区別があります。
SORTED	データ・ソースにあるファイルが昇順にソートされるか否かを指定します。有効な値は Y および N です。デフォルトは N であり、データ・ソースにあるファイルは昇順にソートされません。ソートされるデータ・ソースは、 LC_COLLATE 各国語サポート・カテゴリーの設定によって定義されている現在のロケールの照合順序に従って、昇順でソートされなければなりません。データ・ソースがソートされるように指定する場合、VALIDATE_DATA_FILE オプションを Y に設定してください。
VALIDATE_DATA_FILE	ソートされるファイルの場合、このオプションは、キー列が昇順でソートされていることの検証と NULL キーの検査を、ラッパーが行うかどうかを指定します。この検証は、ニックネームが最初に作成されたときに一度だけ行われます。デフォルトは N です。この場合、ソート順は検証されません。このオプションは、SORTED オプションが Y に設定され、DOCUMENT オプションが指定されていない場合にのみ有効です。

列オプション

表 77. 表構造ファイルの列オプション

オプション	説明
DOCUMENT	ニックネームの作成時にではなく、照会の実行時にファイル・パスを指定できるようにします。有効値は FILE のみです。DOCUMENT オプションで指定できるのは、ニックネームあたり 1 つの列のみです。DOCUMENT オプションに関連付けられる列は、データ・タイプ VARCHAR または CHAR の列でなければなりません。FILE_PATH ニックネーム・オプションではなく、DOCUMENT ニックネーム列オプションを使用するということは、照会の実行中に、このニックネームに対応するファイルが提供されることを意味します。 DOCUMENT オプションに FILE 値が指定されている場合、照会の実行時に提供される値は、このニックネームのニックネーム定義に一致するスキーマを持つファイルの絶対パスです。

Web サービス・オプション・リファレンス

フェデレーテッド・サーバーおよびそのユーザーがデータ・ソースと対話する方法を構成するには、ラッパー、サーバー、ユーザー・マッピング、ニックネーム、および列のオプションを設定および変更します。

ラッパー・オプション

次の表は、このデータ・ソースに適用されるオプションをリストし、指定する必要がある必須選択のオプションを示しています。

表 78. Web サービスのラッパー・オプション

名前	説明
DB2_FENCED	必須。ラッパーを fenced モードとトラステッド・モードのどちらで実行するかを指定します。有効な値は Y と N です。デフォルトは N であり、その場合、ラッパーはトラステッド・モードで実行されます。
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「UserMappingRepositoryLDAP」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。

表 78. *Web* サービスのラッパー・オプション *(続き)*

名前	説明
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は、Java および C です。デフォルトは Java です。
PROXY_TYPE	フェデレーテッド・サーバーがファイアウォールの背後にある場合に、インターネットへのアクセスに使用するプロキシー・タイプを指定します。有効な値は NONE、HTTP、および SOCKS です。デフォルト値は NONE です。
PROXY_SERVER_NAME	プロキシー・サーバーの名前または IP アドレスを指定します。有効な値は、1 から 32760 までの 10 進数のポート番号、またはサービス名です。 IPv6 形式は、IPv6 を構成する場合のみ使用します。
PROXY_SERVER_PORT	プロキシー・サーバー上のプロキシー・サービスのポートまたはサービス名を指定します。有効な値は、10 進数の 1 から 32760 までのポート番号か、サービス名です。
SSL_KEYSTORE_FILE	SSL または TSL を使用する通信用の証明書ストレージ・ファイルを指定します。有効な値は、フェデレーテッド・データベース・エージェントまたは fenced モード・プロセスによってアクセス可能な絶対パス名です。デフォルトは *install path*/cfg/WSWrapperKeystore.kdb です。
SSL_KEYSTORE_PASSWORD	SSL_KEYSTORE_FILE オプションで、ファイルにアクセスするために使用するパスワードを指定します。有効な値はパスワード (フェデレーテッド・データベース・カタログに保管されるときに暗号化される) および file:*file_name* (*file_name* は、スタッシュ・ファイルへの絶対パス) です。
SSL_VERIFY_SERVER_CERTIFICATE	SSL 認証の際にサーバー証明書を検証するかどうかを指定します。有効な値は Y と N です。デフォルトは N です。この場合、証明書は検証されません。

サーバー・オプション

表 79. Web サービスのサーバー・オプション

名前	説明
DB2_MAX_ASYNC_REQUESTS_PER_ QUERY	照会からの並行非同期要求の最大数を指定します。有効な値は -1 から 64000 です。デフォルトは 1 です。 -1 を指定すると、要求の数はフェデレーテッド照会オプティマイザーが決定します。 0 を指定すると、データ・ソースは付加的な非同期要求を受け入れることができません。
DB2_UM_PLUGIN	ユーザー・マッピング・プラグインのインプリメンテーションを指定します。 Java で作成されたプラグインの場合、ユーザー・マッピング・リポジトリー・クラスに対応するクラス名として、大/小文字の区別があるストリングを指定します。例えば、「UserMappingRepositoryLDAP」のように指定します。C で作成されたプラグインの場合、任意の有効な C ライブラリー名を指定します。
DB2_UM_PLUGIN_LANG	ユーザー・マッピング・プラグインの言語を指定します。有効な値は、Java および C です。デフォルトは Java です。
PROXY_AUTHID	プロキシー・サーバー認証のユーザー名を指定します。
PROXY_PASSWORD	プロキシー・サーバー認証のパスワードを指定します。
PROXY_SERVER_NAME	プロキシー・サーバーの名前または IP アドレスを指定します。有効な値は、1 から 32760 までの 10 進数のポート番号、またはサービス名です。 IPv6 形式は、IPv6 を構成する場合のみ使用します。
PROXY_SERVER_PORT	プロキシー・サーバー上のプロキシー・サービスのポートまたはサービス名を指定します。有効な値は、10 進数の 1 から 32760 までのポート番号か、サービス名です。
PROXY_TYPE	フェデレーテッド・サーバーがファイアウォールの背後にある場合に、インターネットへのアクセスに使用するプロキシー・タイプを指定します。有効な値は NONE、HTTP、および SOCKS です。デフォルト値は NONE です。
SSL_CLIENT_CERTIFICATE_ LABEL	SSL 認証の際に送信するクライアント証明書を指定します。値を指定しない場合は、フェデレーテッド・データベースの現在の許可 ID を使用して、証明書を見つけます。

表 79. *Web* サービスのサーバー・オプション *(続き)*

名前	説明
SSL_KEYSTORE_FILE	SSL または TSL を使用する通信用の証明書ストレージ・ファイルを指定します。有効な値は、フェデレーテッド・データベース・エージェントまたは fenced モード・プロセスによってアクセス可能な絶対パス名です。デフォルトは *install path*/cfg/WSWrapperKeystore.kdb です。
SSL_KEYSTORE_PASSWORD	SSL_KEYSTORE_FILE オプションで、ファイルにアクセスするために使用するパスワードを指定します。有効な値はパスワード (フェデレーテッド・データベース・カタログに保管されるときに暗号化される) および file:*file_name* (*file_name* は、スタッシュ・ファイルへの絶対パス) です。
SSL_VERIFY_SERVER_CERTIFICATE	SSL 認証の際にサーバー証明書を検証するかどうかを指定します。有効な値は Y と N です。デフォルトは N です。この場合、証明書は検証されません。

ユーザー・マッピング・オプション

表 80. *Web* サービスのユーザー・マッピング・オプション

名前	説明
PROXY_AUTHID	プロキシー・サーバー認証のユーザー名を指定します。
PROXY_PASSWORD	プロキシー・サーバー認証のパスワードを指定します。このパスワードは、フェデレーテッド・データベース・カタログに保管されるときに暗号化されます。
REMOTE_AUTHID	ローカル・ユーザー ID のマップ先となるリモート・ユーザー ID を指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用する ID が使用されます。
REMOTE_PASSWORD	リモート・ユーザー ID のリモート・パスワードを指定します。このオプションを指定しない場合は、フェデレーテッド・データベースへの接続に使用したパスワードが使用されます。
SSL_CLIENT_CERTIFICATE_ LABEL	SSL 認証の際に送信するクライアント証明書を指定します。値を指定しない場合は、フェデレーテッド・データベースの現在の許可 ID を使用して、証明書を見つけます。

ニックネーム・オプション

表 81. Web サービスのニックネーム・オプション

名前	説明

NAMESPACES	各列の XPATH および TEMPLATE オプションで使用される、ネーム・スペース接頭部に関連付けられているネーム・スペースを指定します。次の構文を使用します。
	`NAMESPACES'prefix1= "actual_namespace1", prefix2="actual_namespace2"'`
	複数のネーム・スペースを区切るには、コンマを使用します。例:
	`NAMESPACES='http://www.myweb.com/cust", i='http://www.myweb.com/cust/id", n='http://www.myweb.com/cust/name"'`

SOAPACTION	ルート・ニックネームの場合は必須です。Web サービス記述言語 (WSDL) 形式からの URI SOAPACTION 属性を指定します。URL には、コロン区切り IPv6 アドレスを、大括弧で囲んで含めることができます (例えば `http://[1080:0:0:0:8:800:200C:417A]`)。**注:** このオプションは、非ルート・ニックネームでは無効です。
STREAMING	ソース文書を処理するために、論理フラグメントに分ける必要があるかどうかを指定します。このフラグメントは、ニックネームの XPath 式に一致するノードに対応しています。このラッパーは、ソース・データの構文解析と処理をフラグメントごとに行います。このタイプの構文解析では、メモリーの使用量が最小限度で済みます。有効な値は Y と N です。デフォルトは N です。この場合、文書は構文解析されません。このオプションは、ルート・ニックネームのみで有効です。
TEMPLATE	SOAP 要求を構成するために使用する、ニックネーム・テンプレートのフラグメントを指定します。フラグメントは、指定されたテンプレートの構文に準拠していなければなりません。このオプションは、ルート・ニックネームのみで有効です。
URL	ルート・ニックネームの場合は必須です。Web サービス・エンドポイントの URL を指定します。サポートされているプロトコルは HTTP および HTTPS です。 URL には、コロン区切り IPv6 アドレスを、大括弧で囲んで含めることができます (例えば `http://[1080:0:0:0:8:800:200C:417A]`)。

表 81. Web サービスのニックネーム・オプション (続き)

名前	説明
XML_CODESET	XML データの送受信に使用するエンコード方式を指定します。このオプションは、内部エンコード方式をオーバーライドします。
XPATH	必須。個々のタプルを表す SOAP 応答エレメントを識別する XPath 式を指定します。この XPath 式は、XPATH ニックネーム列オプションによって識別される列値を評価するためのコンテキストとして使用されます。

列オプション

表 82. Web サービスの列オプション

名前	説明
ESCAPE_INPUT	XML 特殊文字が XML 入力値で置換されるかどうかを指定します。このオプションは、繰り返しエレメントを含む XML フラグメントを含めるなど、XML フラグメントを入力として含めるために使用します。有効な値は Y と N です。N がデフォルトです。この場合、XML 入力値は保持されます。列データ・タイプは、VARCHAR または CHAR でなければなりません。 ESCAPE_INPUT が Y に設定されている場合は、TEMPLATE 列オプションも指定する必要があります。

表 82. Web サービスの列オプション (続き)

名前	説明
FOREIGN_KEY	このニックネームが子ニックネームであることを示し、対応する親ニックネームの名前を指定します。ニックネームには、FOREIGN_KEY 列オプションを 1 つしか指定できません。このオプションの値は、大文字小文字を区別します。この列には XPATH オプションを指定しないでください。列は、親のニックネームと子のニックネームを結合するためだけに使用できます。親ニックネームに別のスキーマ名が付けられている場合、FOREIGN_KEY オプションを含む CREATE NICKNAME ステートメントは失敗します。FOREIGN_KEY 文節で参照されるニックネームが CREATE NICKNAME ステートメントで小文字または大/小文字混合として明示的に定義されているのでない限り、このニックネームを FOREIGN_KEY 文節で参照するときは大文字で指定する必要があります。 **注:** • このオプションが列に設定されている場合、その列に他のオプションを設定することはできません。 • この列オプションを設定した場合は、後から ALTER NICKNAME ステートメントを使用してこのオプションをドロップすることはできません。代わりに、ニックネームをドロップしてから、この列オプションを付けずにニックネームを再作成する必要があります。

表 82. *Web* サービスの列オプション (続き)

名前	説明
PRIMARY_KEY	1 つ以上の子ニックネームを持つ親ニックネームの場合に必須です。このニックネームを親ニックネームとするということを指定します。列データ・タイプは VARCHAR(16) でなければなりません。ニックネームには、PRIMARY_KEY 列オプションを 1 つしか指定できません。 Yes が唯一の有効な値です。この列には XPATH オプションを指定しないでください。列は、親のニックネームと子のニックネームを結合するためだけに使用できます。 **注:** • このオプションが列に設定されている場合、その列に他のオプションを設定することはできません。 • この列オプションを設定した場合は、後から ALTER NICKNAME ステートメントを使用してこのオプションをドロップすることはできません。代わりに、ニックネームをドロップしてから、この列オプションを付けずにニックネームを再作成する必要があります。
SOAPACTIONCOLUMN	照会を実行するときに Web サービス・エンドポイントの SOAP アクションを動的に指定する列を指定します。このオプションは、ルート・ニックネームのみで有効です。ホスト名が IPv6 (コロン区切り) アドレスの場合、ホスト名を大括弧で囲みます。例: 「`http://[1080:0:0:0:8:800:200C:417A]:99/soap`」。このオプションが列に設定されている場合、その列に他のオプションを設定することはできません。
TEMPLATE	XML 入力文書を構成するために使用する、列テンプレートのフラグメントを指定します。フラグメントは、指定されたテンプレートの構文に準拠していなければなりません。 **注:** この列オプションを設定した場合は、後から ALTER NICKNAME ステートメントを使用してこのオプションをドロップすることはできません。代わりに、ニックネームをドロップしてから、この列オプションを付けずにニックネームを再作成する必要があります。

表 82. Web サービスの列オプション (続き)

名前	説明
URLCOLUMN	照会を実行するときに Web サービス・エンドポイントの SOAP アクションを動的に指定する列を指定します。このオプションは、ルート・ニックネームのみで有効です。ホスト名が IPv6 (コロン区切り) アドレスの場合、ホスト名を大括弧で囲みます。例: 「`http://[1080:0:0:0:8:800:200C:417A]:99/soap`」。このオプションが列に設定されている場合、その列に他のオプションを設定することはできません。
XPATH	XML 文書の中でこの列に対応するデータを含む Xpath 式を指定します。 CREATE NICKNAME ステートメントが XPATH ニックネーム・オプションから XPath 式を適用した後で、ラッパーはその XPath 式を評価します。 **注:** この列オプションを設定した場合は、後から ALTER NICKNAME ステートメントを使用してこのオプションをドロップすることはできません。代わりに、ニックネームをドロップしてから、この列オプションを付けずにニックネームを再作成する必要があります。

XML オプション・リファレンス

フェデレーテッド・サーバーとそのユーザーがどのようにデータ・ソースと対話するかを構成するには、ラッパー、サーバー、ユーザー・マッピング、ニックネーム、および列の各オプションを設定および変更します。

ラッパー・オプション

次の表は、このデータ・ソースに適用されるオプションをリストし、指定する必要がある必須選択のオプションを示しています。

表 83. XML のラッパー・オプション

名前	説明
DB2_FENCED	必須。ラッパーを fenced モードとトラステッド・モードのどちらで実行するかを指定します。有効な値は Y と N です。N がデフォルトです。この場合、ラッパーはトラステッド・モードで実行されます。

表 83. XML のラッパー・オプション (続き)

名前	説明
PROXY_TYPE	フェデレーテッド・サーバーがファイアウォールの背後にある場合に、インターネットへのアクセスに使用するプロキシー・タイプを指定します。有効な値は NONE、HTTP、および SOCKS です。デフォルト値は NONE です。
PROXY_SERVER_NAME	プロキシー・サーバーの名前または IP アドレスを指定します。有効な値は、1 から 32760 までの 10 進数のポート番号、またはサービス名です。 IPv6 形式は、IPv6 を構成する場合のみ使用します。
PROXY_SERVER_PORT	プロキシー・サーバー上のプロキシー・サービスのポートまたはサービス名を指定します。有効な値は、10 進数の 1 から 32760 までのポート番号か、サービス名です。
SSL_KEYSTORE_FILE	SSL または TSL を使用する通信用の証明書ストレージ・ファイルを指定します。有効な値は、フェデレーテッド・データベース・エージェントまたは fenced モード・プロセスによってアクセス可能な絶対パス名です。
SSL_KEYSTORE_PASSWORD	SSL_KEYSTORE_FILE オプションで、ファイルにアクセスするために使用するパスワードを指定します。有効な値はパスワード (フェデレーテッド・データベース・カタログに保管されるときに暗号化される) および `file:`*`file_name`* (*file_name* は、スタッシュ・ファイルへの絶対パス) です。
SSL_VERIFY_SERVER_CERTIFICATE	SSL 認証の際にサーバー証明書を検証するかどうかを指定します。有効な値は Y と N です。デフォルトは N です。この場合、証明書は検証されません。

サーバー・オプション

表 84. XML のサーバー・オプション

名前	説明
DB2_MAX_ASYNC_REQUESTS_PER_QUERY	照会からの並行非同期要求の最大数を指定します。有効な値は -1 から 64000 です。デフォルトは 1 です。 -1 を指定すると、要求の数はフェデレーテッド照会オプティマイザーが決定します。 0 は、データ・ソースが、追加の非同期要求に対応できないということを指定します。
PROXY_AUTHID	プロキシー・サーバー認証のユーザー名を指定します。

表 84. XML のサーバー・オプション (続き)

名前	説明
PROXY_PASSWORD	プロキシー・サーバー認証のパスワードを指定します。
PROXY_SERVER_NAME	プロキシー・サーバーの名前または IP アドレスを指定します。有効な値は、1 から 32760 までの 10 進数のポート番号、またはサービス名です。 IPv6 形式は、IPv6 を構成する場合のみ使用します。
PROXY_SERVER_PORT	プロキシー・サーバー上のプロキシー・サービスのポートまたはサービス名を指定します。有効な値は、10 進数の 1 から 32760 までのポート番号か、サービス名です。
PROXY_TYPE	フェデレーテッド・サーバーがファイアウォールの背後にある場合に、インターネットへのアクセスに使用するプロキシー・タイプを指定します。有効な値は NONE、HTTP、および SOCKS です。デフォルト値は NONE です。
SOCKET_TIMEOUT	プロキシー・サーバーからの結果をフェデレーテッド・サーバーが待つ最大時間 (分単位) を指定します。有効な値は 0 以上の任意の数字です。デフォルトは 0 です。この場合、サーバーが待機する時間は無制限です。
SSL_CLIENT_CERTIFICATE_ LABEL	SSL 認証の際に送信するクライアント証明書を指定します。値を指定しない場合は、フェデレーテッド・データベースの現在の許可 ID を使用して、証明書を見つけます。
SSL_KEYSTORE_FILE	SSL または TSL を使用する通信用の証明書ストレージ・ファイルを指定します。有効な値は、フェデレーテッド・データベース・エージェントまたは fenced モード・プロセスによってアクセス可能な絶対パス名です。
SSL_KEYSTORE_PASSWORD	SSL_KEYSTORE_FILE オプションで、ファイルにアクセスするために使用するパスワードを指定します。有効な値はパスワード (フェデレーテッド・データベース・カタログに保管されるときに暗号化される) および `file:`*`file_name`* (*file_name* は、スタッシュ・ファイルへの絶対パス) です。
SSL_VERIFY_SERVER_CERTIFICATE	SSL 認証の際にサーバー証明書を検証するかどうかを指定します。デフォルトは N です。この場合、証明書は検証されません。

ユーザー・マッピング・オプション

表 85. XML のユーザー・マッピング・オプション

名前	説明
PROXY_AUTHID	プロキシー・サーバー認証のユーザー名を指定します。
PROXY_PASSWORD	プロキシー・サーバー認証のパスワードを指定します。
SSL_CLIENT_CERTIFICATE_ LABEL	SSL 認証の際に送信するクライアント証明書を指定します。値を指定しない場合は、フェデレーテッド・データベースの現在の許可 ID を使用して、証明書を見つけます。

ニックネーム・オプション

表 86. XML のニックネーム・オプション

名前	説明
DIRECTORY_PATH	1 つ以上の XML ファイルを含むディレクトリーのパス名を指定します。このオプションは、複数の XML ソース・ファイルに単一のニックネームを作成するために使用します。XML ラッパーは、指定するディレクトリー内にある、.xml 拡張子を持つファイルだけを使用します。 XML ラッパーは、このディレクトリーにある他のすべてのファイルを無視します。このニックネーム・オプションを指定する場合、DOCUMENT 列を指定しないでください。このオプションは、ルート・ニックネームのみで有効です。
FILE_PATH	XML 文書のファイル・パスを指定します。FILE_PATH を指定する場合、DOCUMENT 列を指定しないでください。このオプションは、ルート・ニックネームのみで有効です。
INSTANCE_PARSE_TIME	XML ソース文書の 1 行を構文解析するのに必要な時間 (ミリ秒単位) を指定します。有効な値は、整数または 10 進数値です。デフォルトは 7 です。このオプションは、ルート・ニックネームの列のみで有効です。大規模または複雑な XML ソース構造の照会を最適化するには、INSTANCE_PARSE_TIME、XPATH_EVAL_TIME、および NEXT_TIME の各オプションを変更します。

表 86. XML のニックネーム・オプション (続き)

名前	説明

NAMESPACES	各列の XPATH および TEMPLATE オプションで使用される、ネーム・スペース接頭部に関連付けられているネーム・スペースを指定します。次の構文を使用します。
	`NAMESPACES'prefix1= "actual_namespace1", prefix2="actual_namespace2"'`
	複数のネーム・スペースを区切るには、コンマを使用します。例:
	`NAMESPACES='http://www.myweb.com/cust", i='http://www.myweb.com/cust/id", n='http://www.myweb.com/cust/name"'`

NEXT_TIME	XPath 式から後続のソース・エレメントを位置指定するために必要な時間 (ミリ秒単位) を指定します。デフォルトは 1 です。このオプションはルート・ニックネームおよび非ルート・ニックネームで有効です。大規模または複雑な XML ソース構造の照会を最適化するには、INSTANCE_PARSE_TIME、XPATH_EVAL_TIME、および NEXT_TIME の各オプションを変更します。
STREAMING	ソース文書を処理するために、論理フラグメントに分ける必要があるかどうかを指定します。このフラグメントは、ニックネームの XPath 式に一致するノードに対応しています。このラッパーは、ソース・データの構文解析と処理をフラグメントごとに行います。このタイプの構文解析では、メモリーの使用量が最小限度で済みます。有効な値は Y と N です。デフォルトは N です。この場合、文書は構文解析されません。このオプションは、ルート・ニックネームのみで有効です。
VALIDATE	ソース文書からデータが抽出される前に、その文書が XML スキーマまたは文書タイプ定義 (DTD) に準拠しているかを検証するかどうかを指定します。デフォルトは N です。この場合、検証は行われません。この値を Y に設定する前に、ソース文書が指定する場所にスキーマ・ファイルまたは DTD ファイルを用意します。このオプションは、ルート・ニックネームのみで有効です。 STREAMING と VALIDATE の両方のオプションを Y に設定することは避けてください。
XPATH	必須。個々のタプルを表すエレメントを識別する XPath 式を指定します。この XPath 式は、XPATH 列オプションによって識別される列値を評価するためのコンテキストとして使用されます。

表 86. XML のニックネーム・オプション (続き)

名前	説明
XPATH_EVAL_TIME	ニックネームの XPath 式を評価して最初のエレメントを位置指定するために必要な時間 (ミリ秒単位) を指定します。可能な値は、整数または 10 進数値です。デフォルトは 1 です。このオプションはルート・ニックネームおよび非ルート・ニックネームで有効です。大規模または複雑な XML ソース構造の照会を最適化するには、INSTANCE_PARSE_TIME、XPATH_EVAL_TIME、および NEXT_TIME の各オプションを変更します。

列オプション

表 87. XML の列オプション

名前	説明
DOCUMENT	この列が DOCUMENT 列であることを指定します。 DOCUMENT 列の値は、照会の実行時にニックネームに提供される XML ソース・データのタイプを示します。このオプションは、ルート・ニックネーム (XML 文書の最上位にあるエレメントを識別するニックネーム) の列に対してのみ有効です。DOCUMENT オプションで指定できるのは、ニックネームあたり 1 つの列のみです。FILE_PATH または DIRECTORY_PATH ニックネーム・オプションの代わりに DOCUMENT 列オプションを使用する場合、このニックネームに対応する文書が照会の実行時に提供されます。有効な値は FILE、DIRECTORY、URI、および COLUMN です。デフォルトは FILE です。列は、VARCHAR データ・タイプでなければなりません。

表 87. XML の列オプション (続き)

名前	説明
FOREIGN_KEY	このニックネームが子ニックネームであることを示し、対応する親ニックネームの名前を指定します。ニックネームには、FOREIGN_KEY 列オプションを 1 つしか指定できません。このオプションの値は、大文字小文字を区別します。この列には XPATH オプションを指定しないでください。列は、親のニックネームと子のニックネームを結合するためだけに使用できます。親ニックネームに別のスキーマ名が付けられている場合、FOREIGN_KEY オプションを含む CREATE NICKNAME ステートメントは失敗します。FOREIGN_KEY 文節で参照されるニックネームが CREATE NICKNAME ステートメントで小文字または大/小文字混合として明示的に定義されているのでない限り、このニックネームを FOREIGN_KEY 文節で参照するときは大文字で指定する必要があります。 **注:** • このオプションが列に設定されている場合、その列に他のオプションを設定することはできません。 • この列オプションを設定した場合は、後から ALTER NICKNAME ステートメントを使用してこのオプションをドロップすることはできません。代わりに、ニックネームをドロップしてから、この列オプションを付けずにニックネームを再作成する必要があります。

表 87. XML の列オプション (続き)

名前	説明
PRIMARY_KEY	1 つ以上の子ニックネームを持つ親ニックネームの場合に必須です。このニックネームを親ニックネームとするということを指定します。列データ・タイプは VARCHAR(16) でなければなりません。ニックネームには、PRIMARY_KEY 列オプションを 1 つしか指定できません。 Yes が唯一の有効な値です。この列には XPATH オプションを指定しないでください。列は、親のニックネームと子のニックネームを結合するためだけに使用できます。 **注:** • このオプションが列に設定されている場合、その列に他のオプションを設定することはできません。 • この列オプションを設定した場合は、後から ALTER NICKNAME ステートメントを使用してこのオプションをドロップすることはできません。代わりに、ニックネームをドロップしてから、この列オプションを付けずにニックネームを再作成する必要があります。
XPATH	XML 文書の中でこの列に対応するデータを含む Xpath 式を指定します。 CREATE NICKNAME ステートメントが XPATH ニックネーム・オプションから XPath 式を適用した後で、ラッパーはその XPath 式を評価します。 **注:** この列オプションを設定した場合は、後から ALTER NICKNAME ステートメントを使用してこのオプションをドロップすることはできません。代わりに、ニックネームをドロップしてから、この列オプションを付けずにニックネームを再作成する必要があります。

第 42 章 フェデレーテッド情報を含むグローバル・カタログ表内のビュー

フェデレーテッド・データベース内のカタログ・ビューのほとんどは、他の DB2 for Linux, UNIX, and Windows データベース内のカタログ・ビューと同じです。

いくつかの独自のビューとして、フェデレーテッド・システムに関係のある情報を含むビュー (SYSCAT.WRAPPERS ビューなど) があります。

SYSCAT ビューは、読み取り専用です。SYSCAT スキーマ内のビューに対して更新または挿入操作を発行することはできません。システム・カタログを更新する場合は SYSSTAT ビューを使用することをお勧めします。 SYSCAT ビューを参照するアプリケーションは、SYSCAT ではなく更新可能な SYSSTAT ビューを参照するように変更してください。

次の表は、フェデレーテッド情報が含まれている SYSCAT ビューをリストしています。これらのビューは読み取り専用です。

表 88. 通常、フェデレーテッド・システムで使用されるカタログ・ビュー

カタログ・ビュー	説明
SYSCAT.CHECKS	定義したチェック制約の情報が含まれている。
SYSCAT.COLCHECKS	チェック制約によって参照される列が含まれている。
SYSCAT.COLUMNS	ニックネームを作成したデータ・ソース・オブジェクト (表およびビュー) の列情報が含まれている。
SYSCAT.COLOPTIONS	ニックネーム用にセットした列のオプション値の情報が含まれている。
SYSCAT.CONSTDEP	定義した情報制約の従属関係が含まれている。
SYSCAT.DATATYPES	ローカルの組み込み DB2 データ・タイプおよびユーザー定義 DB2 データ・タイプについてのデータ・タイプ情報が含まれている。
SYSCAT.DBAUTH	個々のユーザーおよびグループが保持するデータベース権限が含まれている。
SYSCAT.FUNCMAPOPTIONS	関数マッピング用にセットしたオプション値の情報が含まれている。
SYSCAT.FUNCMAPPINGS	フェデレーテッド・データベースとデータ・ソース・オブジェクト間の関数マッピングが含まれている。
SYSCAT.INDEXCOLUSE	索引に含める列が含まれている。
SYSCAT.INDEXES	データ・ソース・オブジェクトの索引仕様が含まれている。

表 88. 通常、フェデレーテッド・システムで使用されるカタログ・ビュー (続き)

カタログ・ビュー	説明
SYSCAT.INDEXOPTIONS	索引オプションについての情報が含まれている。
SYSCAT.KEYCOLUSE	ユニーク・キー、主キー、または外部キーの制約によって定義されるキーに加わる列が含まれている。
SYSCAT.NICKNAMES	作成したニックネームについての情報が含まれている。
SYSCAT.REFERENCES	定義した参照制約についての情報が含まれている。
SYSCAT.ROUTINES	ローカル DB2 のユーザー定義関数または関数テンプレートが含まれている。関数テンプレートは、データ・ソース関数と対応付けるために使用されます。
SYSCAT.REVTYPEMAPPINGS	このビューは使用されない。すべてのデータ・タイプ・マッピングは、SYSCAT.TYPEMAPPINGS ビューに記録されます。
SYSCAT.ROUTINEOPTIONS	フェデレーテッド・ルーチンのオプション値についての情報が含まれている。
SYSCAT.ROUTINEPARMOPTIONS	フェデレーテッド・ルーチンのパラメーター・オプション値についての情報が含まれている。
SYSCAT.ROUTINEPARMS	SYSCAT.ROUTINES で定義されているルーチンのパラメーターまたは結果が含まれている。
SYSCAT.ROUTINESFEDERATED	定義したフェデレーテッド・ルーチンについての情報が含まれている。
SYSCAT.SERVERS	データ・ソース・サーバー用に作成したサーバー定義が含まれている。
SYSCAT.TABCONST	各行は、タイプ CHECK、 UNIQUE、PRIMARY KEY、または FOREIGN KEY の表およびニックネームの制約を表す。
SYSCAT.TABLES	作成したそれぞれのローカル DB2 表、フェデレーテッド・ビュー、およびニックネームについての情報が含まれている。
SYSCAT.TYPEMAPPINGS	フォワード・データ・タイプ・マッピングおよびリバース・データ・タイプ・マッピングが含まれている。マッピングは、データ・ソースのデータ・タイプからローカル DB2 データ・タイプへの対応付け。これらのマッピングは、データ・ソース・オブジェクトでニックネームを作成するときに使用されます。
SYSCAT.USEROPTIONS	フェデレーテッド・データベースとデータ・ソース・サーバー間にユーザー・マッピングを作成した時にセットした、ユーザー権限情報が含まれている。

表 88. 通常、フェデレーテッド・システムで使用されるカタログ・ビュー (続き)

カタログ・ビュー	説明
SYSCAT.VIEWS	作成したローカル・フェデレーテッド・ビューについての情報が含まれている。
SYSCAT.WRAPOPTIONS	ラッパーにセットしたオプション値についての情報が含まれている。
SYSCAT.WRAPPERS	ラッパーを作成したそれぞれのデータ・ソースの、ラッパーおよびライブラリー・ファイルの名前が含まれている。

次の表は、フェデレーテッド情報が含まれている SYSSTAT ビューをリストしています。これらのビューは読み取り/書き込みビューであり、そこに含まれている統計を更新することができます。

表 89. 更新可能なフェデレーテッド・グローバル・カタログ・ビュー

カタログ・ビュー	説明
SYSSTAT.COLUMNS	ニックネームを作成したデータ・ソース・オブジェクト (表およびビュー) の列についての統計情報が含まれている。タイプされた表の継承された列については統計は記録されません。
SYSSTAT.INDEXES	データ・ソース・オブジェクトの索引仕様についての統計情報が含まれている。
SYSSTAT.ROUTINES	ユーザー定義関数についての統計情報が含まれている。組み込み関数は含まれない。タイプされた表の継承された列については統計は記録されません。
SYSSTAT.TABLES	基本表についての情報が含まれている。このビューには、ビュー、シノニム、および別名の情報は含まれていません。タイプされた表の場合、表の階層のルート表だけがビューに含まれます。タイプされた表の継承された列については統計は記録されません。

第 43 章 フェデレーテッド・システムの関数マッピング・オプション

フェデレーテッド・サーバーは、DB2 関数とデータ・ソース関数の間のデフォルトのマッピングを提供します。ほとんどのデータ・ソースの場合、ラッパー内にデフォルトの関数マッピングがあります。フェデレーテッド・サーバーが認識しないデータ・ソース関数を使用したり、デフォルトのマッピングを変更したりする場合は、関数マッピングを作成する必要があります。

関数マッピングを作成する時、データ・ソース関数の名前を指定し、マップされる関数を使用可能にする必要があります。その後マップされた関数を使用する時、照会オプティマイザーは、その関数をデータ・ソースで実行する場合のコストとその関数をフェデレーテッド・サーバーで実行する場合のコストを比較します。

表 90. 関数マッピングのオプション

名前	説明
DISABLE	デフォルト関数マッピングを使用可能または使用不可にします。有効な値は Y と N です。デフォルトは N です。
REMOTE_NAME	データ・ソース関数の名前。デフォルトはローカル名です。

第 44 章 SQL ステートメントで有効なサーバーのタイプ

サーバー・タイプは、サーバー定義が表すデータ・ソースの種類を示します。

サーバー・タイプは、ベンダー、目的、およびオペレーティング・システムに応じて変わります。サポートされる値はデータ・ソースによって異なります。

ほとんどのデータ・ソースで、CREATE SERVER ステートメントに有効なサーバー・タイプを指定しなければなりません。

表 91. データ・ソースおよびサーバー・タイプ

データ・ソース	サーバー・タイプ
BioRS	サーバー・タイプは CREATE SERVER ステートメントでは必要ありません。
Excel	サーバー・タイプは CREATE SERVER ステートメントでは必要ありません。
IBM DB2 Universal Database for Linux, UNIX, and Windows	DB2/UDB
IBM DB2 Universal Database for System i and AS/400®	DB2/ISERIES
IBM DB2 Universal Database for z/OS	DB2/ZOS
IBM DB2 for VM	DB2/VM
Informix	INFORMIX
JDBC	JDBC (JDBC ドライバー 3.0 以降がサポートする JDBC データ・ソースでは必須。)
Microsoft SQL Server	MSSQLSERVER (DataDirect Connect ODBC 4.2 以降のドライバー、または Microsoft SQL Server ODBC 3.0 以降のドライバーがサポートするデータ・ソースでは必須。)
ODBC	ODBC (ODBC 3.x ドライバーがサポートする ODBC データ・ソースでは必須。)
OLE DB	サーバー・タイプは CREATE SERVER ステートメントでは必要ありません。
Oracle	ORACLE (Oracle NET8 クライアント・ソフトウェアがサポートする Oracle データ・ソースでは必須。)
Sybase (CTLIB)	SYBASE
表構造ファイル	サーバー・タイプは CREATE SERVER ステートメントでは必要ありません。
Teradata	TERADATA
Web サービス	サーバー・タイプは CREATE SERVER ステートメントでは必要ありません。
XML	サーバー・タイプは CREATE SERVER ステートメントでは必要ありません。

第 45 章 データ・タイプ・マッピング

リレーショナル・データ・ソースのデータ・タイプ・マッピングには、順方向タイプ・マッピング、逆方向タイプ・マッピング、および Unicode に固有のタイプ・マッピングがあります。非リレーショナル・データ・ソースは、それぞれ固有のデータ・タイプをサポートします。

デフォルトの順方向データ・タイプ・マッピング

データ・ソースのデータ・タイプとフェデレーテッド・データベースのデータ・タイプの間のマッピングには、順方向データ・タイプ・マッピングと逆方向データ・タイプ・マッピングの 2 種類があります。順方向タイプ・マッピングとは、リモート・データ・タイプから対応するローカル・タイプへのマッピングのことです。

デフォルトのタイプ・マッピングをオーバーライドすることも、CREATE TYPE MAPPING ステートメントを使用して新規タイプ・マッピングを作成することもできます。

これらのマッピングは、特に注記のない限り、すべてのサポートされるバージョンで有効です。

データ・ソースからフェデレーテッド・データベースへの、すべてのデフォルトの順方向データ・タイプ・マッピングでは、フェデレーテッド・スキーマは SYSIBM です。

以下の表に、フェデレーテッド・データベースのデータ・タイプとデータ・ソースのデータ・タイプとの間の、デフォルトの順方向マッピングを示しています。

DB2 Database for Linux, UNIX, and Windows データ・ソースのデフォルトの順方向データ・タイプ・マッピング

以下の表に、DB2 Database for Linux, UNIX, and Windows データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 92. DB2 Database for Linux, UNIX, and Windows のデフォルトの順方向データ・タイプ・マッピング (一部の列は示されていません)

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
BIGINT	-	-	-	-	-	-	BIGINT	-	0	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	-	-	-	-	-	-	CHAR	-	0	N
CHAR	-	-	-	-	Y	-	CHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DATE	-	-	-	-	-	-	TIMESTAMP[1]	-	0	-

表 92. DB2 Database for Linux, UNIX, and Windows のデフォルトの順方向データ・タイプ・マッピング (一部の列は示されていません) (続き)

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DECFLOAT [2]	-	-	-	-	-	-	DECFLOAT	-	0	-
DOUBLE	-	-	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
LONGVAR	-	-	-	-	N	-	CLOB	-	-	-
LONGVAR	-	-	-	-	Y	-	BLOB	-	-	-
LONGVARG	-	-	-	-	-	-	DBCLOB	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIME-STAMP(*p*)	-	-	*p*	*p*	-	-	TIME-STAMP(*p*)	-	*p*	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	0	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	0	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	0	N

注:

1. date_compat 構成パラメーターが ON に設定されている場合、フェデレーテッドのタイプは TIMESTAMP(0) になります。
2. SAME_DECFLT_ROUNDING サーバー・オプションはデフォルトでは N に設定され、 SAME_DECFLT_ROUNDING が Y に設定されていないと操作はリモート・データ・ソースにプッシュダウンされません。 SAME_DECFLT_ROUNDING サーバー・オプションについては、DB2 データベース・オプションの参照情報を参照してください。

DB2 for System i データ・ソースのデフォルトの順方向データ・タイプ・マッピング

以下の表に、DB2 for System i データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 93. DB2 for System i のデフォルトの順方向データ・タイプ・マッピング (一部の列は示されていません)

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y

表 93. DB2 for System i のデフォルトの順方向データ・タイプ・マッピング (一部の列は示されていません) (続き)

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
NUMERIC	-	-	-	-	-	-	DECIMAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP(6)	-	6	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

DB2 for VM and VSE データ・ソースのデフォルトの順方向データ・タイプ・マッピング

以下の表に、DB2 for VM and VSE データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 94. DB2 Server for VM and VSE のデフォルトの順方向データ・タイプ・マッピング (一部の列は示されていません)

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBAHW	-	-	-	-	-	-	SMALLINT	-	0	-
DBAINT	-	-	-	-	-	-	INTEGER	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-

表 94. DB2 Server for VM and VSE のデフォルトの順方向データ・タイプ・マッピング (一部の列は示されていません) (続き)

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP(6)	-	6	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPH	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

DB2 for z/OS データ・ソースのデフォルトの順方向データ・タイプ・マッピング

以下の表に、DB2 for z/OS データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 95. DB2 for z/OS のデフォルトの順方向データ・タイプ・マッピング (一部の列は示されていません)

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
BIGINT	0	8	0	0	" "	"/0"	BIGINT	0	0	N
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
ROWID	-	-	-	-	Y	-	VARCHAR	40	-	Y
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP(6)	-	6	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

Informix データ・ソースのデフォルトの順方向データ・タイプ・マッピング

以下の表に、Informix データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 96. Informix のデフォルトの順方向データ・タイプ・マッピング (一部の列は示されていません)

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
BLOB	-	-	-	-	-	-	BLOB	2147483647	-	-
BOOLEAN	-	-	-	-	-	-	CHARACTER	1	-	-
BYTE	-	-	-	-	-	-	BLOB	2147483647	-	-
CHAR	1	254	-	-	-	-	CHARACTER	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CLOB	-	-	-	-	-	-	CLOB	2147483647	-	-
DATE	-	-	-	-	-	-	DATE	4	-	-
DATE	-	-	-	-	-	-	TIMESTAMP[1]	-	0	-
DATETIME[2]	0	4	0	4	-	-	DATE	4	-	-
DATETIME	6	10	6	10	-	-	TIME	3	-	-
DATETIME	0	4	6	15	-	-	TIMESTAMP(6)	10	6	-
DATETIME	6	10	11	15	-	-	TIMESTAMP(6)	10	6	-
DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
DECIMAL	32	130	-	-	-	-	DOUBLE	8	-	-
DECIMAL	1	32	255	255	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
INTERVAL	-	-	-	-	-	-	VARCHAR	25	-	-
INT8	-	-	-	-	-	-	BIGINT	19	0	-
LVARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
MONEY	1	31	0	31	-	-	DECIMAL	-	-	-
MONEY	32	32	-	-	-	-	DOUBLE	8	-	-
NCHAR	1	254	-	-	-	-	CHARACTER	-	-	-
NCHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
NVARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
REAL	-	-	-	-	-	-	REAL	4	-	-
SERIAL	-	-	-	-	-	-	INTEGER	4	-	-
SERIAL8	-	-	-	-	-	-	BIGINT	-	-	-
SMALLFLOAT	-	-	-	-	-	-	REAL	4	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
TEXT	-	-	-	-	-	-	CLOB	2147483647	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-

表 96. Informix のデフォルトの順方向データ・タイプ・マッピング (一部の列は示されていません) (続き)

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ

注:

1. date_compat 構成パラメーターが ON に設定されている場合、フェデレーテッドのタイプは TIMESTAMP(0) になります。
2. Informix DATETIME データ・タイプについて、DB2 UNIX and Windows フェデレーテッド・サーバーは Informix 高位修飾子を REMOTE_LENGTH として、Informix 低位修飾子を REMOTE_SCALE として使用します。

Informix 修飾子は、Informix クライアント SDK の `datatime.h` ファイルで定義された「TU_」定数です。この定数は、以下のようになります。

0 = YEAR	8 = MINUTE	13 = FRACTION(3)
2 = MONTH	10 = SECOND	14 = FRACTION(4)
4 = DAY	11 = FRACTION(1)	15 = FRACTION(5)
6 = HOUR	12 = FRACTION(2)	

JDBC データ・ソースのデフォルトの順方向データ・タイプ・マッピング

以下の表に、JDBC データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 97. JDBC のデフォルトの順方向データ・タイプ・マッピング

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
BIGINT	-	-	-	-	-	-	BIGINT	8	-	-
BINARY	-	254	-	-	-	-	CHAR	-	-	Y
BINARY	255	32672	-	-	-	-	VARCHAR	-	-	Y
BINARY	32673	2147483647	-	-	-	-	BLOB	2147483647	-	-
BIT	-	-	-	-	-	-	SMALLINT	2	-	-
BLOB	-	-	-	-	-	-	BLOB	2147483647	-	-
BOOLEAN	-	-	-	-	-	-	SMALLINT	2	-	-
CHAR	-	254	-	-	-	-	CHAR	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CHAR	32673	2147483647	-	-	-	-	CLOB	2147483647	-	-
CLOB	-	-	-	-	-	-	CLOB	2147483647	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-
DATE	-	-	-	-	-	-	TIMESTAMP[1]	-	-	-
DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
FLOAT	-	-	-	-	-	-	FLOAT	4	-	-
INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
LONGVARCHAR	-	32672	-	-	-	-	VARCHAR	-	-	-
LONGVARCHAR	32673	2147483647	-	-	-	-	CLOB	2147483647	-	-

表 97. JDBC のデフォルトの順方向データ・タイプ・マッピング (続き)

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
LONGVARBINARY	-	32672	-	-	-	-	VARCHAR	-	-	Y
LONGVARBINARY	32673	2147483647	-	-	-	-	BLOB	2147483647	-	-
LONGNVARCHAR	-	16336	-	-	-	-	VARGRAPHIC	-	-	-
LONGNVARCHAR[2]	16337	1073741823	-	-	-	-	DBCLOB	-	-	-
NCHAR[2]	-	127	-	-	-	-	GRAPHIC	-	-	-
NCHAR[2]	128	16336	-	-	-	-	VARGRAPHIC	-	-	-
NCHAR[2]	16337	1073741823	-	-	-	-	DBCLOB	-	-	-
NCLOB[2]	-	-	-	-	-	-	DBCLOB	-	-	-
NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
NUMERIC	32	38	0	38	-	-	DOUBLE	8	-	-
NVARCHAR[2]	-	16336	-	-	-	-	VARGRAPHIC	-	-	-
NVARCHAR[2]	16337	1073741823	-	-	-	-	DBCLOB	-	-	-
REAL							REAL	4		
SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
TIME	-	-	-	-	-	-	TIME	3	-	-
TIMESTAMP	-	-	-	-	-	-	TIME-STAMP(6)	10	6	-
TIMESTAMP(*p*)	-	-	-	-	-	-	TIME-STAMP(6)	10	6	-
TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
VARBINARY	-	32672	-	-	-	-	VARCHAR	-	-	Y
VARBINARY	32673	2147483647	-	-	-	-	BLOB	2147483647	-	-
VARCHAR	-	32672	-	-	-	-	VARCHAR	-	-	-
VARCHAR	32673	2147483647	-	-	-	-	CLOB	2147483647	-	-

注:

1. date_compat 構成パラメーターが ON に設定されている場合、フェデレーテッドのタイプは TIMESTAMP(0) になります。
2. JDBC 4.0 ドライバーでのみサポートされるデータ・タイプは、NCHAR、NVARCHAR、LONGVARCHAR、および NCLOB です。

データ・タイプ
DATALINK、OTHER、JAVA_OBJECT、DISTINCT、STRUCT、ARRAY、および REF は JDBC ラッパーでサポートされません。

Microsoft SQL Server データ・ソースのデフォルトの順方向データ・タイプ・マッピング

以下の表に、Microsoft SQL Server データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 98. *Microsoft SQL Server* のデフォルトの順方向データ・タイプ・マッピング

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
bigint	-	-	-	-	-	-	BIGINT	-	-	-
binary	1	254	-	-	-	-	CHARACTER	-	-	Y
binary	255	8000	-	-	-	-	VARCHAR	-	-	Y
bit	-	-	-	-	-	-	SMALLINT	2	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	8000	-	-	-	-	VARCHAR	-	-	N
datetime	-	-	-	-	-	-	TIME-STAMP(6)	10	6	-
decimal	1	31	0	31	-	-	DECIMAL	-	-	-
decimal	32	38	0	38	-	-	DOUBLE	-	-	-
float	-	8	-	-	-	-	DOUBLE	8	-	-
float	-	4	-	-	-	-	REAL	4	-	-
image	-	-	-	-	-	-	BLOB	2147483647	-	Y
int	-	-	-	-	-	-	INTEGER	4	-	-
money	-	-	-	-	-	-	DECIMAL	19	4	-
nchar	1	127	-	-	-	-	CHAR	-	-	N
nchar	128	4000	-	-	-	-	VARCHAR	-	-	N
numeric	1	31	0	31	-	-	DECIMAL	-	-	-
numeric	32	38	0	38	-	-	DOUBLE	8	-	-
ntext	-	-	-	-	-	-	CLOB	2147483647	-	Y
nvarchar	1	4000	-	-	-	-	VARCHAR	-	-	N
real	-	-	-	-	-	-	REAL	4	-	-
smallint	-	-	-	-	-	-	SMALLINT	2	-	-
smalldatetime	-	-	-	-	-	-	TIME-STAMP(6)	10	6	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
SQL_BIGINT	-	-	-	-	-	-	BIGINT	-	-	-
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	8000	-	-	-	-	VARCHAR	-	-	N
SQL_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_GUID	-	-	-	-	-	-	VARCHAR	-	-	Y
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_ LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N
SQL_ LONGVARBINARY	-	-	-	-	-	-	BLOB	-	-	Y

表 98. *Microsoft SQL Server* のデフォルトの順方向データ・タイプ・マッピング *(続き)*

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_NUMERIC	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_REAL	-	-	-	-	-	-	REAL	8	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	6	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_VARBINARY	1	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	8000	-	-	-	-	VARCHAR	-	-	N
SQL_WCHAR	1	254	-	-	-	-	CHARACTER	-	-	N
SQL_WCHAR	255	8800	-	-	-	-	VARCHAR	-	-	N
SQL_WLONGVARCHAR	-	1073741823	-	-	-	-	CLOB	2147483647	-	N
SQL_WVARCHAR	1	16336	-	-	-	-	VARCHAR	-	-	N
text	-	-	-	-	-	-	CLOB	-	-	N
timestamp	-	-	-	-	-	-	VARCHAR	8		Y
tinyint	-	-	-	-	-	-	SMALLINT	2	-	-
uniqueidentifier	1	4000	-	-	Y	-	VARCHAR	16	-	Y
varbinary	1	8000	-	-	-	-	VARCHAR	-	-	Y
varchar	1	8000	-	-	-	-	VARCHAR	-	-	N

ODBC データ・ソースのデフォルトの順方向データ・タイプ・マッピング

以下の表に、ODBC データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 99. *ODBC* のデフォルトの順方向データ・タイプ・マッピング *(一部の列は示されていません)*

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
SQL_BIGINT	-	-	-	-	-	-	BIGINT	8	-	-
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	32672	-	-	-	-	VARCHAR	-	-	N
SQL_DATE	-	-	-	-	-	-	DATE	-	-	-
SQL_DATE	-	-	-	-	-	-	TIMESTAMP[1]	-	-	-
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-

表 99. ODBC のデフォルトの順方向データ・タイプ・マッピング (一部の列は示されていません) (続き)

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	8	-	-	-	-	FLOAT	8	-	-
SQL_FLOAT	-	4	-	-	-	-	FLOAT	4	-	-
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_ LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N
SQL_ LONGVARBINARY	-	-	-	-	-	-	BLOB	2147483647	-	Y
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_NUMERIC	32	32	0	31	-	-	DOUBLE	8	-	-
SQL_REAL	-	-	-	-	-	-	REAL	4	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TIMESTAMP	-	-	-	-	-	-	TIME-STAMP(6)	10	6	-
SQL_TIMESTAMP(*p*)	-	-	-	-	-	-	TIME-STAMP(6)	10	6	-
SQL_TYPE_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_TYPE_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TYPE_ TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	-	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_VARBINARY	1	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	N
SQL_WCHAR	1	127	-	-	-	-	CHAR	-	-	N
SQL_WCHAR	128	16336	-	-	-	-	VARCHAR	-	-	N
SQL_WVARCHAR	1	16336	-	-	-	-	VARCHAR	-	-	N
SQL_ WLONGVARCHAR	-	1073741823	-	-	-	-	CLOB	2147483647	-	N

注:

1. date_compat 構成パラメーターが ON に設定されている場合、フェデレーテッドのタイプは TIMESTAMP(0) になります。

Oracle NET8 データ・ソースのデフォルトの順方向データ・タイプ・マッピング

以下の表に、Oracle NET8 データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 100. Oracle NET8 のデフォルトの順方向データ・タイプ・マッピング

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
BLOB	0	0	0	0	-	\0	BLOB	2147483647	0	Y
CHAR	1	254	0	0	-	\0	CHAR	0	0	N

表 100. Oracle NET8 のデフォルトの順方向データ・タイプ・マッピング (続き)

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
CHAR	255	2000	0	0	-	\0	VARCHAR	0	0	N
CLOB	0	0	0	0	-	\0	CLOB	2147483647	0	N
DATE	0	0	0	0	-	\0	TIME-STAMP(6)	0	0	N
FLOAT	1	126	0	0	-	\0	DOUBLE	0	0	N
LONG	0	0	0	0	-	\0	CLOB	2147483647	0	N
LONG RAW	0	0	0	0	-	\0	BLOB	2147483647	0	Y
NUMBER	10	18	0	0	-	\0	BIGINT	0	0	N
NUMBER	1	38	-84	127	-	\0	DOUBLE	0	0	N
NUMBER	1	31	0	31	-	>=	DECIMAL	0	0	N
NUMBER	1	4	0	0	-	\0	SMALLINT	0	0	N
NUMBER	5	9	0	0	-	\0	INTEGER	0	0	N
NUMBER	-	10	0	0	-	\0	DECIMAL	0	0	N
RAW	1	2000	0	0	-	\0	VARCHAR	0	0	Y
ROWID	0	0	0	NULL	-	\0	CHAR	18	0	N
TIMESTAMP(*p*) [1]	-	-	-	-	-	\0	TIME-STAMP(6)	10	6	N
VARCHAR2	1	4000	0	0	-	\0	VARCHAR	0	0	N

注:

1.

- TIMESTAMP*(p)* は、可変の位取りが 0 から 9 までのタイム・スタンプを表します。 Oracle タイム・スタンプの位取りは、デフォルトで TIMESTAMP(6) にマップされます。ユーザー定義のタイプ・マッピングを使用すると、このデフォルト・タイプ・マッピングを変更し、Oracle TIMESTAMP を同じ位取りのフェデレーテッド TIMESTAMP にマップすることができます。
- このタイプ・マッピングは、Oracle 9i 以降のクライアント/サーバー構成だけで有効です。

Sybase データ・ソースのデフォルトの順方向データ・タイプ・マッピング

以下の表に、Sybase データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 101. Sybase CTLIB のデフォルトの順方向データ・タイプ・マッピング

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
binary	1	254	-	-	-	-	CHAR	-	-	Y
binary	255	32672	-	-	-	-	VARCHAR	-	-	Y
bit	-	-	-	-	-	-	SMALLINT	-	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	32672	-	-	-	-	VARCHAR	-	-	N

表 101. Sybase CTLIB のデフォルトの順方向データ・タイプ・マッピング (続き)

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
char null (varchar を参照)										
date	-	-	-	-	-	-	DATE	-	-	-
date	-	-	-	-	-	-	TIMESTAMP[1]	-	-	-
datetime	-	-	-	-	-	-	TIMESTAMP(6)	-	-	-
datetimn	-	-	-	-	-	-	TIMESTAMP	-	-	-
decimal	1	31	0	31	-	-	DECIMAL	-	-	-
decimal	32	38	0	38	-	-	DOUBLE	-	-	-
decimaln	1	31	0	31	-	-	DECIMAL	-	-	-
decimaln	32	38	0	38	-	-	DOUBLE	-	-	-
float	-	4	-	-	-	-	REAL	-	-	-
float	-	8	-	-	-	-	DOUBLE	-	-	-
floatn	-	4	-	-	-	-	REAL	-	-	-
floatn	-	8	-	-	-	-	DOUBLE	-	-	-
image	-	-	-	-	-	-	BLOB	-	-	-
int	-	-	-	-	-	-	INTEGER	-	-	-
intn	-	-	-	-	-	-	INTEGER	-	-	-
money	-	-	-	-	-	-	DECIMAL	19	4	-
moneyn	-	-	-	-	-	-	DECIMAL	19	4	-
nchar	1	254	-	-	-	-	CHAR	-	-	N
nchar	255	32672	-	-	-	-	VARCHAR	-	-	N
nchar null (nvarchar を参照)										
numeric	1	31	0	31	-	-	DECIMAL	-	-	-
numeric	32	38	0	38	-	-	DOUBLE	-	-	-
numericn	1	31	0	31	-	-	DECIMAL	-	-	-
numericn	32	38	0	38	-	-	DOUBLE	-	-	-
nvarchar	1	32672	-	-	-	-	VARCHAR	-	-	N
real	-	-	-	-	-	-	REAL	-	-	-
smalldatetime	-	-	-	-	-	-	TIMESTAMP(6)	-	-	-
smallint	-	-	-	-	-	-	SMALLINT	-	-	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
sysname	-	-	-	-	-	-	VARCHAR	30	-	N
text	-	-	-	-	-	-	CLOB	-	-	-
time	-	-	-	-	-	-	TIME	-	-	-
timestamp	-	-	-	-	-	-	VARCHAR	8	-	Y
tinyint	-	-	-	-	-	-	SMALLINT	-	-	-
unichar[2]	1	254	-	-	-	-	CHAR	-	-	N
unichar[2]	255	32672	-	-	-	-	VARCHAR	-	-	N

表 101. Sybase CTLIB のデフォルトの順方向データ・タイプ・マッピング (続き)

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
unichar null (univarchar を参照)										
univarchar[2]	1	32672	-	-	-	-	VARCHAR	-	-	N
varbinary	1	32672	-	-	-	-	VARCHAR	-	-	Y
varchar	1	32672	-	-	-	-	VARCHAR	-	-	N

注:

1. date_compat 構成パラメーターが ON に設定されている場合、フェデレーテッドのタイプは TIMESTAMP(0) になります。
2. 非 Unicode のフェデレーテッド・データベースに関して有効です。

Teradata データ・ソースのデフォルトの順方向データ・タイプ・マッピング

以下の表に、Teradata データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 102. Teradata のデフォルトの順方向データ・タイプ・マッピング (一部の列は示されていません)

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
BIGINT [1]	-	-	-	-	-	-	BIGINT [1]	-	-	-
BLOB	1	2097088000	-	-	-	-	BLOB	-	-	-
BYTE	1	254	-	-	-	-	CHAR	-	-	Y
BYTE	255	32672	-	-	-	-	VARCHAR	-	-	Y
BYTE	32673	64000	-	-	-	-	BLOB	-	-	-
BYTEINT	-	-	-	-	-	-	SMALLINT	-	-	-
CHAR	1	254	-	-	-	-	CHARACTER	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CHAR	32673	64000	-	-	-	-	CLOB	-	-	-
CLOB	1	2097088000 (Latin)	-	-	-	-	CLOB	-	-	-
CLOB	1	1048544000 (Unicode)	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-
DATE	-	-	-	-	-	-	TIMESTAMP [2]	-	-	-
DECIMAL	1	31[3]	0	31[3]	-	-	DECIMAL	-	-	-
DOUBLE PRECISION	-	-	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	-	-
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	-	-
GRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-

表 102. Teradata のデフォルトの順方向データ・タイプ・マッピング (一部の列は示されていません) (続き)

リモートのタイプ名	リモートの長さの下限	リモートの長さの上限	リモートの位取りの下限	リモートの位取りの上限	リモートのビット・データ	リモートのデータ演算子	フェデレーテッドのタイプ名	フェデレーテッドの長さ	フェデレーテッドの位取り	フェデレーテッドのビット・データ
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
INTERVAL	-	-	-	-	-	-	CHAR	-	-	-
NUMERIC	1	18	0	18	-	-	DECIMAL	-	-	-
REAL	-	-	-	-	-	-	DOUBLE	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	0	21	0	21	-	-	TIME	-	-	-
TIME-STAMP(*p*)	-	-	*p*	*p*	-	-	TIMESTAMP(6)	10	6	-
VARBYTE	1	32762	-	-	-	-	VARCHAR	-	-	Y
VARBYTE	32763	64000	-	-	-	-	BLOB	-	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
VARCHAR	32673	64000	-	-	-	-	CLOB	-	-	-
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	-	-
VARGRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-

注:

1. BIGINT データ・タイプは、Teradata バージョン 2 リリース 6 以降に適用されます。
2. date_compat 構成パラメーターが ON に設定されている場合、フェデレーテッドのタイプは TIMESTAMP(0) になります。
3. リモートの長さの上限とリモートの位取りの上限の値 31 は、Teradata バージョン 2 リリース 6 以降に適用されます。それ以前のバージョンの場合、この値は 18 になります。

順方向データ・タイプ・マッピングのサンプル

順方向タイプ・マッピングのサンプルを使用して、精度を持つ TIMESTAMP データ・タイプのサポートを活用することができます。

Informix データ・ソースの場合、これらのタイプ・マッピングはニックネーム列タイプ、フェデレーテッド・プロシージャー・パラメーター、パススルー、およびフェデレーテッド・プロシージャー結果セットに対して使用されます。

Informix 以外のデータ・ソースの場合、これらのタイプ・マッピングはニックネーム列タイプとフェデレーテッド・プロシージャー・パラメーターに対するマッピングにのみ影響を与えます。パススルーおよびフェデレーテッド・プロシージャー結果セットには影響しません。

順方向データ・タイプ・マッピング - Informix のサンプル

フェデレーテッド・オブジェクトを作成する際に、Informix 用に提供されている順方向タイプ・マッピングのサンプルを使用できます。

フェデレーテッド・オブジェクトを作成する前に、以下のマッピングを作成する必要があります。

```
CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(0)
   TO SERVER TYPE informix REMOTE TYPE datetime(0,10);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(1)
   TO SERVER TYPE informix REMOTE TYPE datetime(0,11);
```

```
CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(2)
   TO SERVER TYPE informix REMOTE TYPE datetime(0,12);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(3)
   TO SERVER TYPE informix REMOTE TYPE datetime(0,13);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(4)
   TO SERVER TYPE informix REMOTE TYPE datetime(0,14);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(5)
   TO SERVER TYPE informix REMOTE TYPE datetime(0,15);
```

順方向データ・タイプ・マッピング - Microsoft SQL Server のサンプル

フェデレーテッド・オブジェクトを作成する際に、Microsoft SQL Server 用に提供されている順方向タイプ・マッピングのサンプルを使用できます。

ニックネームまたはフェデレーテッド・プロシージャーを作成する前に、以下のマッピングを作成する必要があります。

```
CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(3)
   TO SERVER TYPE mssqlserver REMOTE TYPE "datetime";

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(0)
   TO SERVER TYPE mssqlserver REMOTE TYPE "smalldatetime";
```

順方向データ・タイプ・マッピング - Oracle のサンプル

フェデレーテッド・オブジェクトを作成する際に、Oracle 用に提供されている順方向タイプ・マッピングのサンプルを使用できます。

ニックネームまたはフェデレーテッド・プロシージャーを作成する前に、以下のマッピングを作成する必要があります。

```
CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(0)
   TO SERVER TYPE oracle REMOTE TYPE timestamp(0);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(1)
   TO SERVER TYPE oracle REMOTE TYPE timestamp(1);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(2)
   TO SERVER TYPE oracle REMOTE TYPE timestamp(2);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(3)
   TO SERVER TYPE oracle REMOTE TYPE timestamp(3);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(4)
   TO SERVER TYPE oracle REMOTE TYPE timestamp(4);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(5)
   TO SERVER TYPE oracle REMOTE TYPE timestamp(5);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(7)
   TO SERVER TYPE oracle REMOTE TYPE timestamp(7);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(8)
   TO SERVER TYPE oracle REMOTE TYPE timestamp(8);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(9)
   TO SERVER TYPE oracle REMOTE TYPE timestamp(9);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(0)
   TO SERVER TYPE oracle REMOTE TYPE date;
```

順方向データ・タイプ・マッピング - Sybase のサンプル

フェデレーテッド・オブジェクトを作成する際に、Sybase 用に提供されている順方向タイプ・マッピングのサンプルを使用できます。

ニックネームまたはフェデレーテッド・プロシージャーを作成する前に、以下のマッピングを作成する必要があります。

```
CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(3)
   TO SERVER TYPE sybase REMOTE TYPE datetime);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(0)
   TO SERVER TYPE sybase REMOTE TYPE smalldatetime);
```

順方向データ・タイプ・マッピング - Teradata のサンプル

フェデレーテッド・オブジェクトを作成する際に、Teradata 用に提供されている順方向タイプ・マッピングのサンプルを使用できます。

ニックネームまたはフェデレーテッド・プロシージャーを作成する前に、以下のマッピングを作成する必要があります。

```
CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(0)
   TO SERVER TYPE teradata REMOTE TYPE timestamp(0);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(1)
   TO SERVER TYPE teradata REMOTE TYPE timestamp(1);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(2)
   TO SERVER TYPE teradata REMOTE TYPE timestamp(2);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(3)
   TO SERVER TYPE teradata REMOTE TYPE timestamp(3);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(4)
   TO SERVER TYPE teradata REMOTE TYPE timestamp(4);

CREATE TYPE MAPPING FROM LOCAL TYPE timestamp(5)
   TO SERVER TYPE teradata REMOTE TYPE timestamp(5);
```

デフォルトの逆方向データ・タイプ・マッピング

ほとんどのデータ・ソースの場合、ラッパー内にデフォルトのタイプ・マッピングがあります。

データ・ソースのデータ・タイプとフェデレーテッド・データベースのデータ・タイプの間のマッピングには、順方向データ・タイプ・マッピングと逆方向データ・タイプ・マッピングの 2 種類があります。順方向タイプ・マッピングとは、リモート・データ・タイプから対応するローカル・タイプへのマッピングのことです。マッピングのもう 1 つのタイプは、逆方向タイプ・マッピングであり、透過 DDL でリモート表を作成または変更するために使用します。

DB2 ファミリー・データ・ソース用のデフォルトのタイプ・マッピングは、 DRDA ラッパー内にあります。 Informix 用のデフォルトのタイプ・マッピングは INFORMIX ラッパーにあります。その他のタイプ・マッピングについても同様です。

フェデレーテッド・データベースに対してリモートの表またはビューを定義する場合、その定義には逆方向タイプ・マッピングが含まれます。このマッピングは、各

列のローカルのフェデレーテッド・データベース・データ・タイプから、対応するリモートのデータ・タイプに対して行われます。例えば、ローカル・タイプ REAL が Informix タイプ SMALLFLOAT を指す、デフォルトの逆方向タイプ・マッピングがあります。

フェデレーテッド・データベースでは、LONG VARCHAR、LONG VARGRAPHIC、およびユーザー定義タイプへのマッピングはサポートされません。

CREATE TABLE ステートメントを使用してリモート表を作成する場合、そのリモート表に組み込みたいローカル・データ・タイプを指定します。これらのデフォルトの逆方向タイプ・マッピングは、これらの列に対応するリモート・タイプを割り当てます。例えば、CREATE TABLE ステートメントを使用して、列 C2 を持つ Informix 表を定義するとします。このステートメント内では、C2 のデータ・タイプとして BIGINT を指定します。 BIGINT のデフォルトの逆方向タイプ・マッピングは、どのバージョンの Informix で表を作成しているかによって異なります。Informix 表内の C2 のマッピングは、Informix バージョン 8 の場合は DECIMAL に、 Informix バージョン 9 の場合は INT8 に対して行われます。

デフォルトの逆方向タイプ・マッピングをオーバーライドすることも、CREATE TYPE MAPPING ステートメントを使用して新規の逆方向タイプ・マッピングを作成することもできます。

以下の表に、フェデレーテッド・データベースのローカル・データ・タイプとリモート・データ・ソースのデータ・タイプとの間の、デフォルトの逆方向マッピングを示しています。

これらのマッピングは、特に注記のない限り、すべてのサポートされるバージョンで有効です。

DB2 Database for Linux, UNIX, and Windows データ・ソースのデフォルトの逆方向データ・タイプ・マッピング

以下の表に、DB2 Database for Linux, UNIX, and Windows データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 103. DB2 Database for Linux, UNIX, and Windows のデフォルトの逆方向データ・タイプ・マッピング (一部の列は示されていません)

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	フェデレーテッドのビット・データ
BIGINT	-	8	-	-	-	-	BIGINT	-	-	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE[1]	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-

表 103. DB2 Database for Linux, UNIX, and Windows のデフォルトの逆方向データ・タイプ・マッピング (一部の列は示されていません) (続き)

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	フェデレーテッドのビット・データ
DECFLOAT [2]	-	8	-	-	-	-	DECFLOAT	-	0	-
DECFLOAT [2]	-	16	-	-	-	-	DECFLOAT	-	0	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIME-STAMP(*p*)	-	-	*p*	*p*	-	-	TIME-STAMP(*p*)	-	*p* [3]	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	-	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

注:

1. date_compat パラメーターが OFF に設定されている場合、フェデレーテッドの DATE は TIMESTAMP(0) にマップされます。
2. SAME_DECFLT_ROUNDING サーバー・オプションはデフォルトでは N に設定され、 SAME_DECFLT_ROUNDING が Y に設定されていないと操作はリモート・データ・ソースにプッシュダウンされません。 SAME_DECFLT_ROUNDING サーバー・オプションについては、DB2 データベース・オプションの参照情報を参照してください。
3. バージョン 9.5 では、TIMESTAMP のリモートの位取りは 6 です。

DB2 for System i データ・ソースのデフォルトの逆方向データ・タイプ・マッピング

以下の表に、DB2 for System i データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 104. DB2 for System i のデフォルトの逆方向データ・タイプ・マッピング (一部の列は示されていません)

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	リモートのビット・データ
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	N
CHARACTER	-	-	-	-	Y	-	CHARACTER	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	NUMERIC	-	-	-

表 104. DB2 for System i のデフォルトの逆方向データ・タイプ・マッピング (一部の列は示されていません) (続き)

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	リモートのビット・データ
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	FLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIME-STAMP*(p)*	-	-	*p*	*p*	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	VARG	-	-	N

DB2 for VM and VSE データ・ソースのデフォルトの逆方向データ・タイプ・マッピング

以下の表に、DB2 for VM and VSE データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 105. DB2 for VM and VSE のデフォルトの逆方向データ・タイプ・マッピング (一部の列は示されていません)

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	リモートのビット・データ
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	-
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIME-STAMP*(p)*	-	-	*p*	*p*	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPH	-	-	N

DB2 for z/OS データ・ソースのデフォルトの逆方向データ・タイプ・マッピング

以下の表に、DB2 for z/OS データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 106. DB2 for z/OS のデフォルトの逆方向データ・タイプ・マッピング (一部の列は示されていません)

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	リモートのビット・データ
BIGINT	0	8	0	0	" "	"/0"	BIGINT	0	0	N
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	–
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIME-STAMP*(p)*	-	-	*p*	*p*	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	N

Informix データ・ソースのデフォルトの逆方向データ・タイプ・マッピング

以下の表に、Informix データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 107. Informix のデフォルトの逆方向データ・タイプ・マッピング

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	リモートのビット・データ
BIGINT [1]	-	-	-	-	-	-	DECIMAL	19	-	-
BIGINT [2]	-	-	-	-	-	-	INT8	-	-	-
BLOB	1	2147483647	-	-	-	-	BYTE	-	-	-
CHARAC-TER	-	-	-	-	N	-	CHAR	-	-	-

表 107. *Informix* のデフォルトの逆方向データ・タイプ・マッピング (続き)

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	リモートのビット・データ
CHARAC-TER	-	-	-	-	Y	-	BYTE	-	-	-
CLOB	1	2147483647	-	-	-	-	TEXT	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	SMALLFLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	DATETIME	6	10	-
TIMESTAMP	-	10	-	-	-	-	DATETIME	0	15	-
VARCHAR	1	254	-	-	N	-	VARCHAR	-	-	-
VARCHAR [1]	255	32672	-	-	N	-	TEXT	-	-	-
VARCHAR	-	-	-	-	Y	-	BYTE	-	-	-
VARCHAR [2]	255	2048	-	-	N	-	LVARCHAR	-	-	-
VARCHAR [2]	2049	32672	-	-	N	-	TEXT	-	-	-

注:

1. このタイプ・マッピングは、バージョン 8 以前の Informix サーバーでのみ有効です。
2. このタイプ・マッピングは、バージョン 9 以降の Informix サーバーでのみ有効です。

Informix DATETIME データ・タイプについて、フェデレーテッド・サーバーは Informix 高位修飾子を REMOTE_LENGTH として、Informix 低位修飾子を REMOTE_SCALE として使用します。

Informix 修飾子は、Informix クライアント SDK の `datatime.h` ファイルで定義された「TU_」定数です。この定数は、以下のようになります。

0 = YEAR	8 = MINUTE	13 = FRACTION(3)
2 = MONTH	10 = SECOND	14 = FRACTION(4)
4 = DAY	11 = FRACTION(1)	15 = FRACTION(5)
6 = HOUR	12 = FRACTION(2)	

JDBC データ・ソースのデフォルトの逆方向データ・タイプ・マッピング

以下の表に、DB2 JDBC ドライバーのタイプ・マッピングに準拠する JDBC データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 108. *JDBC* のデフォルトの逆方向データ・タイプ・マッピング

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	リモートのビット・データ
BIGINT	-	-	-	-	-	-	BIGINT	-	-	-

表 108. JDBC のデフォルトの逆方向データ・タイプ・マッピング (続き)

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	リモートのビット・データ
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	-	-	-	-	Y	-	BINARY	-	-	-
CHAR	-	-	-	-	-	-	CHAR	-	-	-
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	NCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	NCHAR	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	-	-
TIMESTAMP(*p*)	-	-	*p*	*p*	-	-	TIMESTAMP(*p*)	-	min(9,*p*)	-
VARCHAR	-	-	-	-	Y	-	VARBINARY	-	-	-
VARCHAR	-	-	-	-	N	-	VARCHAR	-	-	-
VARGRAPHIC	-	-	-	-	-	-	NVARCHAR	-	-	-

Microsoft SQL Server データ・ソースのデフォルトの逆方向データ・タイプ・マッピング

以下の表には、Microsoft SQL Server データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 109. Microsoft SQL Server のデフォルトの逆方向データ・タイプ・マッピング (一部の列は示されていません)

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	リモートのビット・データ
BIGINT	-	-	-	-	-	-	bigint	-	-	-
BLOB	-	-	-	-	-	-	image	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
DATE	-	4	-	-	-	-	datetime	-	-	-
DECIMAL	-	-	-	-	-	-	decimal	-	-	-
DOUBLE	-	8	-	-	-	-	float	-	-	-
INTEGER	-	-	-	-	-	-	int	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
REAL	-	4	-	-	-	-	real	-	-	-
TIME	-	3	-	-	-	-	datetime	-	-	-

表 109. Microsoft SQL Server のデフォルトの逆方向データ・タイプ・マッピング (一部の列は示されていません) (続き)

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	リモートのビット・データ
TIMESTAMP	-	10	-	-	-	-	datetime	-	-	-
VARCHAR	1	8000	-	-	N	-	varchar	-	-	-
VARCHAR	8001	32672	-	-	N	-	text	-	-	-
VARCHAR	1	8000	-	-	Y	-	varbinary	-	-	-
VARCHAR	8001	32672	-	-	Y	-	image	-	-	-

ODBC データ・ソースのデフォルトの逆方向データ・タイプ・マッピング

以下の表に、ODBC データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 110. ODBC のデフォルトの逆方向データ・タイプ・マッピング (一部の列は示されていません)

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	リモートのビット・データ
BIGINT	-	-	-	-	-	-	SQL_BIGINT	-	-	-
BLOB	-	-	-	-	-	-	SQL_LONGVARBINARY	-	-	-
CHAR	-	-	-	-	Y	-	SQL_BINARY	-	-	-
CHAR	-	-	-	-	N	-	SQL_CHAR	-	-	-
CLOB	-	-	-	-	-	-	SQL_LONGVARCHAR	-	-	-
DATE	-	4	-	-	-	-	SQL_TYPE_DATE	-	-	-
DBCLOB	-	-	-	-	-	-	SQL_WLONGVARCHAR	-	-	-
DECIMAL	-	-	-	-	-	-	SQL_DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	SQL_DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	SQL_FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	SQL_WCHAR	-	-	-
INTEGER	-	-	-	-	-	-	SQL_INTEGER	-	-	-
REAL	-	4	-	-	-	-	SQL_REAL	-	-	-
SMALLINT	-	-	-	-	-	-	SQL_SMALLINT	-	-	-
TIME	-	3	-	-	-	-	SQL_TYPE_TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	SQL_TYPE_TIMESTAMP(*p*)	-	-	-
VARCHAR	-	-	-	-	Y	-	SQL_VARBINARY	-	-	-
VARCHAR	-	-	-	-	N	-	SQL_VARCHAR	-	-	-
VARGRAPHIC	-	-	-	-	Y	-	SQL_WVARCHAR	-	-	-

Oracle NET8 データ・ソースのデフォルトの逆方向データ・タイプ・マッピング

以下の表に、Oracle NET8 データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 111. Oracle NET8 のデフォルトの逆方向データ・タイプ・マッピング

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	リモートのビット・データ
BIGINT	0	8	0	0	N	\0	NUMBER	19	0	N
BLOB	0	2147483647	0	0	Y	\0	BLOB	0	0	Y
CHARACTER	1	254	0	0	N	\0	CHAR	0	0	N
CHARACTER	1	254	0	0	Y	\0	RAW	0	0	Y
CLOB	0	2147483647	0	0	N	\0	CLOB	0	0	N
DATE[1]	0	4	0	0	N	\0	DATE	0	0	N
DECIMAL	0	0	0	0	N	\0	NUMBER	0	0	N
DECFLOAT	0	8	0	0	N	\0	NUMBER	0	0	N
DECFLOAT	0	16	0	0	N	\0	NUMBER	0	0	N
DOUBLE	0	8	0	0	N	\0	FLOAT	126	0	N
FLOAT	0	8	0	0	N	\0	FLOAT	126	0	N
INTEGER	0	4	0	0	N	\0	NUMBER	10	0	N
REAL	0	4	0	0	N	\0	FLOAT	63	0	N
SMALLINT	0	2	0	0	N	\0	NUMBER	5	0	N
TIME	0	3	0	0	N	\0	DATE	0	0	N
TIMESTAMP [2]	0	10	0	0	N	\0	DATE	0	0	N
TIMESTAMP*(p)* [3]	-	-	-	-	N	\0	TIME-STAMP*(p)*	-	-	N
VARCHAR	1	4000	0	0	N	\0	VARCHAR2	0	0	N
VARCHAR	1	2000	0	0	Y	\0	RAW	0	0	Y

注:

1. date_compat パラメーターが OFF に設定されている場合、フェデレーテッドの DATE は Oracle の date にマップされます。date_compat パラメーターが ON に設定されている場合、フェデレーテッドの DATE (TIMESTAMP(0) と等価) は、Oracle の TIMESTAMP(0) にマップされます。
2. このタイプ・マッピングは、Oracle バージョン 8 でのみ有効です。
3.
   - TIMESTAMP*(p)* は、Oracle の場合は 0 から 9 まで、フェデレーションの場合は 0 から 12 までの可変の位取りを持つタイム・スタンプを表します。位取りが 0 から 9 までである場合、リモートの Oracle TIMESTAMP の位取りは、フェデレーテッドの TIMESTAMP と同じになります。フェデレーテッドの TIMESTAMP の位取りが 9 より大きい場合、対応する Oracle TIMESTAMP の位取りは、Oracle の最大の位取りである 9 になります。
   - このタイプ・マッピングは、Oracle バージョン 9、10、および 11 でのみ有効です。

Sybase データ・ソースのデフォルトの逆方向データ・タイプ・マッピング

以下の表に、Sybase データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 112. Sybase CTLIB のデフォルトの逆方向データ・タイプ・マッピング

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	リモートのビット・データ
BIGINT	-	-	-	-	-	-	decimal	19	0	-
BLOB	-	-	-	-	-	-	image	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
DATE	-	-	-	-	-	-	datetime	-	-	-
DECIMAL	-	-	-	-	-	-	decimal	-	-	-
DOUBLE	-	-	-	-	-	-	float	-	-	-
INTEGER	-	-	-	-	-	-	integer	-	-	-
REAL	-	-	-	-	-	-	real	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
TIME	-	-	-	-	-	-	datetime	-	-	-
TIMESTAMP	-	-	-	-	-	-	datetime	-	-	-
VARCHAR [1]	1	255	-	-	N	-	varchar	-	-	-
VARCHAR [1]	256	32672	-	-	N	-	text	-	-	-
VARCHAR [2]	1	16384	-	-	N	-	varchar	-	-	-
VARCHAR [2]	16385	32672	-	-	N	-	text	-	-	-
VARCHAR [1]	1	255	-	-	Y	-	varbinary	-	-	-
VARCHAR [1]	256	32672	-	-	Y	-	image	-	-	-
VARCHAR [2]	1	16384	-	-	Y	-	varbinary	-	-	-
VARCHAR [2]	16385	32672	-	-	Y	-	image	-	-	-

注:

1. このタイプ・マッピングは、Sybase サーバーのバージョン 12.0 以前の CTLIB に関してのみ有効です。
2. このタイプ・マッピングは、Sybase サーバーのバージョン 12.5 以降の CTLIB に関してのみ有効です。

Teradata データ・ソースのデフォルトの逆方向データ・タイプ・マッピング

以下の表に、Teradata データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 113. Teradata のデフォルトの逆方向データ・タイプ・マッピング (一部の列は示されていません)

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	リモートのビット・データ
BIGINT [1]	-	-	-	-	-	-	BIGINT	-	-	-
BLOB	1	2097088000	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	-
CHARACTER	-	-	-	-	Y	-	BYTE	-	-	-
CLOB	1	2097088000	-	-		-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-

表 113. Teradata のデフォルトの逆方向データ・タイプ・マッピング (一部の列は示されていません) (続き)

フェデレーテッドのタイプ名	フェデレーテッドの長さの下限	フェデレーテッドの長さの上限	フェデレーテッドの位取りの下限	フェデレーテッドの位取りの上限	フェデレーテッドのビット・データ	フェデレーテッドのデータ演算子	リモートのタイプ名	リモートの長さ	リモートの位取り	リモートのビット・データ
DBCLOB [2]	1	64000	-	-	-	-	VARGRAPHIC	-	-	-
DECIMAL	1	18/31[3]	0	18/31[3]	-	-	DECIMAL	-	-	-
DECIMAL[4]	19	31	0	31	-	-	FLOAT	8	-	-
DOUBLE	-	-	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	FLOAT	8	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	15	-	-
TIMESTAMP	10	10	6	6	-	-	TIMESTAMP	26	6	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARBYTE	-	-	-
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

注:

1. BIGINT データ・タイプは、Teradata バージョン 2 リリース 6 以降に適用されます。
2. Teradata VARGRAPHIC データ・タイプには、指定された長さ (1 から 32000) の DBCLOB データ・タイプしか格納できません。
3. 値 31 は、Teradata バージョン 2 リリース 6 以降に適用されます。値 18 は、それ以前のバージョンに適用されます。
4. この DECIMAL 行は、Teradata バージョン 2 リリース 5 以前に適用されます。

Unicode のデフォルトのデータ・タイプ・マッピング

特定のデータ・ソースは、Unicode データベースに対する順方向データ・タイプ・マッピングと逆方向データ・タイプ・マッピングをサポートしています。

JDBC データ・ソースのデフォルトの順方向データ・タイプ・マッピング (Unicode)

以下の表に、フェデレーテッド・データベースが Unicode データベースである場合の、JDBC データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 114. JDBC データ・ソースのデフォルトの順方向データ・タイプ・マッピング (Unicode)

UTF-8	JDBC	
データ・タイプ	データ・タイプ	長さ
CHAR	CHAR	1 から 254 バイト
VARCHAR	VARCHAR	1 から 32672 バイト
CLOB	CLOB	-
GRAPHIC	NCHAR	1 から 127 文字
VARGRAPHIC	NVARCHAR	1 から 16336 文字

表 114. JDBC データ・ソースのデフォルトの順方向データ・タイプ・マッピング (Unicode) (続き)

UTF-8	JDBC	
データ・タイプ	データ・タイプ	長さ
DBCLOB	NCLOB	-

JDBC データ・ソースのデフォルトの逆方向データ・タイプ・マッピング (Unicode)

以下の表に、フェデレーテッド・データベースが Unicode データベースである場合の、JDBC データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 115. JDBC データ・ソースのデフォルトの逆方向データ・タイプ・マッピング (Unicode)

UTF-8		JDBC
データ・タイプ	長さ	データ・タイプ
CHAR	1 から 254 バイト	CHAR
VARCHAR	1 から 32672 バイト	VARCHAR
CLOB	1 から 2147483647 バイト	CLOB
GRAPHIC	1 から 127 文字	NCHAR
VARGRAPHIC	1 から 16336 文字	NVARCHAR
DBCLOB	1 から 1073741823 文字	NCLOB

Microsoft SQL Server データ・ソースのデフォルトの順方向データ・タイプ・マッピング (Unicode)

以下の表に、フェデレーテッド・データベースが Unicode データベースである場合の、 Microsoft SQL Server データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 116. Microsoft SQL Server データ・ソースのデフォルトの順方向データ・タイプ・マッピング (Unicode)

UTF-8	Microsoft SQL Server	
データ・タイプ	データ・タイプ	長さ
CHAR	CHAR	1 から 254 バイト
VARCHAR	CHAR	255 から 8000 バイト
	VARCHAR	1 から 8000 バイト
CLOB	TEXT	-
GRAPHIC	NCHAR	1 から 127 文字
VARGRAPHIC	NCHAR	128 から 16336 文字
	NVARCHAR	1 から 16336 文字
DBCLOB	NTEXT	-

Microsoft SQL Server データ・ソースのデフォルトの逆方向データ・タイプ・マッピング (Unicode)

以下の表に、フェデレーテッド・データベースが Unicode データベースである場合の、 Microsoft SQL Server データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 117. Microsoft SQL Server データ・ソースのデフォルトの逆方向データ・タイプ・マッピング (Unicode)

UTF-8		**Microsoft SQL Server**
データ・タイプ	**長さ**	**データ・タイプ**
CHAR	1 から 254 バイト	CHAR
VARCHAR	1 から 32672 バイト	VARCHAR
CLOB	1 から 2 147 483 647 バイト	TEXT
GRAPHIC	1 から 127 文字	NCHAR
VARGRAPHIC	1 から 16336 文字	NVARCHAR
DBCLOB	1 から 1 073 741 823 文字	NTEXT

NET8 データ・ソースのデフォルトの順方向データ・タイプ・マッピング (Unicode)

以下の表に、フェデレーテッド・データベースが Unicode データベースである場合の、NET8 データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 118. NET8 データ・ソースのデフォルトの順方向データ・タイプ・マッピング (Unicode)

UTF-8	**Oracle**	
データ・タイプ	**データ・タイプ**	**長さ**
CHAR	CHAR	1 から 254 バイト
VARCHAR	CHAR	255 から 2000 バイト
	VARCHAR2	1 から 4000 バイト
DBCLOB	NCLOB	
GRAPHIC	NCHAR	1 から 127 文字
VARGRAPHIC	NCHAR	128 から 1000 文字
	NVARCHAR2	1 から 2000 文字

NET8 データ・ソースのデフォルトの逆方向データ・タイプ・マッピング (Unicode)

以下の表に、フェデレーテッド・データベースが Unicode データベースである場合の、NET8 データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 119. NET8 データ・ソースのデフォルトの逆方向データ・タイプ・マッピング (Unicode)

UTF-8		Oracle
データ・タイプ	長さ	データ・タイプ
CHAR	1 から 254 バイト	CHAR
VARCHAR	1 から 4000 バイト	VARCHAR2
CLOB	1 から 2 147 483 647 バイト	CLOB
GRAPHIC	1 から 127 文字	NCHAR
VARGRAPHIC	1 から 2000 文字	NVARCHAR2
DBCLOB	1 から 1 073 741 823 文字	NCLOB

ODBC データ・ソースのデフォルトの順方向データ・タイプ・マッピング (Unicode)

以下の表に、フェデレーテッド・データベースが Unicode データベースである場合の、ODBC データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 120. ODBC データ・ソースのデフォルトの順方向データ・タイプ・マッピング (Unicode)

UTF-8	ODBC	
データ・タイプ	データ・タイプ	長さ
CHAR	SQL_CHAR	1 から 254 バイト
VARCHAR	SQL_CHAR	255 から 32672 バイト
	SQL_VARCHAR	1 から 32672 バイト
CLOB	SQL_LONGVARCHAR	-
GRAPHIC	SQL_WCHAR	1 から 127 文字
VARGRAPHIC	SQL_WCHAR	128 から 16336 文字
	SQL_WVARCHAR	1 から 16336 文字
DBCLOB	SQL_WLONGVARCHAR	-

ODBC データ・ソースのデフォルトの逆方向データ・タイプ・マッピング (Unicode)

以下の表に、フェデレーテッド・データベースが Unicode データベースである場合の、ODBC データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 121. ODBC データ・ソースのデフォルトの逆方向データ・タイプ・マッピング (Unicode)

UTF-8		ODBC
データ・タイプ	長さ	データ・タイプ
CHAR	1 から 254 バイト	SQL_CHAR
VARCHAR	1 から 32672 バイト	SQL_VARCHAR
CLOB	1 から 2 147 483 647 バイト	SQL_LONGVARCHAR
GRAPHIC	1 から 127 文字	SQL_WCHAR
VARGRAPHIC	1 から 16336 文字	SQL_WVARCHAR

表 121. ODBC データ・ソースのデフォルトの逆方向データ・タイプ・マッピング (Unicode) (続き)

UTF-8		ODBC
データ・タイプ	長さ	データ・タイプ
DBCLOB	1 から 1 073 741 823 文字	SQL_WLONGVARCHAR

Sybase データ・ソースのデフォルトの順方向データ・タイプ・マッピング (Unicode)

以下の表に、フェデレーテッド・データベースが Unicode データベースである場合の、Sybase CTLIB データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 122. Sybase CTLIB データ・ソースのデフォルトの順方向データ・タイプ・マッピング (Unicode)

UTF-8	Sybase	
データ・タイプ	データ・タイプ	長さ
CHAR	char	1 から 254 バイト
	nchar	1 から 127 文字
VARCHAR	char	255 から 32672 バイト
	varchar	1 から 32672 バイト
	nchar	128 から 16336 文字
	nvarchar	1 から 16336 文字
CLOB	text	
GRAPHIC	unichar	1 から 127 文字
VARGRAPHIC	unichar	128 から 16336 文字
	univarchar	1 から 16336 文字

Sybase データ・ソースのデフォルトの逆方向データ・タイプ・マッピング (Unicode)

以下の表に、フェデレーテッド・データベースが Unicode データベースである場合の、Sybase CTLIB データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 123. Sybase CTLIB データ・ソースのデフォルトの逆方向データ・タイプ・マッピング (Unicode)

UTF-8		Sybase
データ・タイプ	長さ	データ・タイプ
CHAR	1 から 254 バイト	char
VARCHAR	1 から 32672 バイト	varchar
CLOB	1 から 2 147 483 647 バイト	text
GRAPHIC	1 から 127 文字	unichar
VARGRAPHIC	1 から 16336 文字	univarchar

非リレーショナル・データ・ソースでサポートされるデータ・タイプ

非リレーショナル・データ・ソースのほとんどの場合、そのデータ・ソースへのアクセスに使用するニックネームを作成する際に、データ・タイプなどの列情報を指定する必要があります。

一部の非リレーショナル・ラッパーは、データ・ソースへのアクセスに必要なすべての列を作成します。これらを*固定列* と呼びます。その他のラッパーでは、列のデータ・タイプの一部または全部を、ユーザーが CREATE NICKNAME ステートメントで指定します。

以下のセクションに、ユーザーがデータ・タイプを指定できるラッパーと、そのラッパーでサポートされるデータ・タイプをリストしています。

BioRS ラッパーでサポートされるデータ・タイプ

以下の表に、BioRS ラッパーでサポートされる DB2 データ・タイプをリストしています。

表 124. DB2 データ・タイプにマップする BioRS データ・タイプ

BioRS データ・タイプ	DB2 データ・タイプ
AUTHOR	CHARACTER, CLOB, VARCHAR
DATE	CHARACTER, CLOB, VARCHAR
NUMBER	CHARACTER, CLOB, VARCHAR
REFERENCE	CHARACTER, CLOB, VARCHAR
TEXT	CHARACTER, CLOB, VARCHAR

CLOB データ・タイプで許可される最大長は 5 メガバイトです。

Excel ラッパーでサポートされるデータ・タイプ

以下の表に、Excel ラッパーでサポートされる DB2 データ・タイプをリストしています。

表 125. DB2 データ・タイプにマップする Excel データ・タイプ

Excel データ・タイプ	DB2 データ・タイプ
文字	CHARACTER
日付	DATE
数値	DOUBLE
数値	FLOAT
整数	INTEGER
文字	VARCHAR

Script ラッパーでサポートされるデータ・タイプ

以下の表に、Script ラッパーでサポートされる DB2 データ・タイプをリストしています。

表 126. DB2 データ・タイプにマップする Script データ・タイプ

XML データ・タイプ	DB2 データ・タイプ
文字	BLOB
文字	CHARACTER
文字	CHARACTER FOR BIT DATA
文字	CLOB (最大長は 5 メガバイト)
日付	DATE
数値	DECIMAL
数値	DOUBLE
数値	FLOAT
整数	INTEGER
数値	REAL
整数	SMALLINT
文字	VARCHAR
文字	VARCHAR FOR BIT DATA

表構造ファイル・ラッパーでサポートされるデータ・タイプ

以下の表に、表構造ファイル・ラッパーでサポートされる DB2 データ・タイプをリストしています。

表 127. DB2 データ・タイプにマップする表構造ファイルのデータ・タイプ

表構造ファイルのデータ・タイプ	DB2 データ・タイプ
文字	CHARACTER
文字	CLOB (最大長は 5 メガバイト)
数値	DECIMAL
数値	DOUBLE
数値	FLOAT
整数	INTEGER
数値	REAL
整数	SMALLINT
文字	VARCHAR

Web サービス・ラッパーでサポートされるデータ・タイプ

以下の表に、Web サービス・ラッパーでサポートされる DB2 データ・タイプをリストしています。 Web サービス・ラッパーは、XML データ・タイプを使用します。

表 128. Web サービス・ラッパーの、DB2 データ・タイプにマップする XML データ・タイプ

XML データ・タイプ	DB2 データ・タイプ
文字	BLOB
文字	CHARACTER

表 128. Web サービス・ラッパーの、DB2 データ・タイプにマップする XML データ・タイプ (続き)

XML データ・タイプ	DB2 データ・タイプ
文字	CHARACTER FOR BIT DATA
文字	CLOB (最大長は 5 メガバイト)
日付	DATE
数値	DECIMAL
数値	DOUBLE
数値	FLOAT
整数	INTEGER
数値	REAL
整数	SMALLINT
文字	VARCHAR
文字	VARCHAR FOR BIT DATA

XML ラッパーでサポートされるデータ・タイプ

以下の表に、XML ラッパーでサポートされる DB2 データ・タイプをリストしています。

表 129. XML ラッパーの、DB2 データ・タイプにマップする XML データ・タイプ

XML データ・タイプ	DB2 データ・タイプ
文字	BLOB
文字	CHARACTER
文字	CHARACTER FOR BIT DATA
文字	CLOB (最大長は 5 メガバイト)
日付	DATE
数値	DECIMAL
数値	DOUBLE
数値	FLOAT
整数	INTEGER
数値	REAL
整数	SMALLINT
文字	VARCHAR
文字	VARCHAR FOR BIT DATA
XML 文書	XML

アクセシビリティー対応資料

資料は XHTML 形式で提供され、これは、ほとんどの Web ブラウザーで表示可能です。

XHTML では、ご使用のブラウザーで設定された表示設定に従って資料を表示できます。スクリーン・リーダーやその他の支援技術も使用できます。

構文図は小数点付き 10 進数の形式で提供されます。この形式は、スクリーン・リーダーを使用してオンライン文書にアクセスした場合にのみ、使用可能です。

特記事項および商標

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

特記事項

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することが

できます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年).このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _年を入れる_.All rights reserved.

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtml をご覧ください。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標または登録商標です。

Adobe は Adobe Systems Incorporated の米国およびその他の国における登録商標です。

IT Infrastructure Library は英国 Office of Government Commerce の一部である the Central Computer and Telecommunications Agency の登録商標です。

Intel、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Intel Centrino、Intel Centrino ロゴ、Celeron、Intel Xeon、Intel SpeedStep、Itanium、および Pentium は Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。

Linux は、Linus Torvalds の米国およびその他の国における登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

ITIL は英国 Office of Government Commerce の登録商標および共同体登録商標であって、米国特許商標庁にて登録されています。

UNIX は The Open Group の米国およびその他の国における登録商標です。

Cell Broadband Engine, Cell/B.E は、米国およびその他の国における Sony Computer Entertainment, Inc. の商標であり、同社の許諾を受けて使用しています。

Java およびすべてのJava 関連の商標およびロゴは Oracleやその関連会社の米国およびその他の国における商標または登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[ワ行]

[数字]

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Printed in Japan

SA88-4827-00

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21

Spine information:

IBM InfoSphere Federation Server バージョン 10.1 フェデレーテッド・システムの管理ガイド