



Informatica®
10.4.0

パフォーマンスのチューニングの概要

Informatica パフォーマンスのチューニングの概要

10.4.0

2019 年 12 月

© 著作権 Informatica LLC 2009, 2020

本ソフトウェアおよびマニュアルは、使用および開示の制限を定めた個別の使用許諾契約のもとでのみ提供されています。本マニュアルのいかなる部分も、いかなる手段（電子的複製、写真複製、録音など）によっても、Informatica LLC の事前の承諾なしに複製または転載することは禁じられています。

Informatica および Informatica ロゴは、米国およびその他の国における Informatica LLC の商標または登録商標です。Informatica の商標の最新リストは、Web (<https://www.informatica.com/trademarks.html>) にあります。その他の企業名および製品名は、それぞれの企業の商標または登録商標です。

米政府の権利プログラム、ソフトウェア、データベース、および関連文書や技術データは、米国政府の顧客に配信され、「商用コンピュータソフトウェア」または「商業技術データ」は、該当する連邦政府の取得規制と代理店固有の補足規定に基づきます。このように、使用、複製、開示、変更、および適応は、適用される政府の契約に規定されている制限およびライセンス条項に従うものとし、政府契約の条項によって適当な範囲において、FAR 52.227-19、商用コンピュータソフトウェアライセンスの追加権利を規定します。

本ソフトウェアまたはドキュメンテーション（あるいはその両方）の一部は、第三者が保有する著作権の対象となります。必要な第三者の通知は、製品に含まれています。

本マニュアルの情報は、予告なしに変更されることがあります。このドキュメントで問題が見つかった場合は、infa_documentation@informatica.com までご報告ください。

Informatica 製品は、それらが提供される契約の条件に従って保証されます。Informatica は、商品性、特定目的への適合性、非侵害性の保証等を含めて、明示的または黙示的ないかなる種類の保証をせず、本マニュアルの情報を「現状のまま」提供するものとします。

発行日: 2020-02-04

目次

序文	7
Informatica のリソース	7
Informatica Network	7
Informatica ナレッジベース	7
Informatica マニュアル	7
Informatica 製品可用性マトリックス	8
Informatica Velocity	8
Informatica Marketplace	8
Informatica グローバルカスタマサポート	8
第 1 章 : パフォーマンスのチューニングの概要	9
パフォーマンスのチューニングの概要	9
パフォーマンスのチューニングプロセス	10
ターゲットのボトルネック	10
ソースのボトルネック	11
マッピングのボトルネック	11
コンピュータシステムのボトルネック	11
Windows におけるシステムのボトルネックの特定	11
UNIX におけるシステムのボトルネックの特定	12
ランタイムのボトルネック	12
SQL データサービスの最適化のボトルネック	13
Web サービスの最適化のボトルネック	13
接続ボトルネック	13
第 2 章 : ターゲットの最適化	14
ターゲットの最適化の概要	14
フラットファイルターゲットの最適化	14
データベースのチェックポイント間隔	15
バルクロード	15
データベースターゲットの最適化	15
第 3 章 : ソースの最適化	17
ソースの最適化の概要	17
フラットファイルソースの最適化	18
クエリの最適化	18
条件フィルタ	19
個別に選択	19
ヒント	19
ヒントに関するルールとガイドライン	20
ヒントの作成	20

制約.	21
制約の設定.	21
カスタマイズデータオブジェクトの最適化.	22
データベースソースの最適化.	23
第 4 章: トランスフォーメーションの最適化.	24
トランスフォーメーションの最適化.	24
アグリゲータトランスフォーメーションの最適化.	25
式の最適化.	25
Java トランスフォーメーションの最適化.	27
Java トランスフォーメーションによる初期選択の最適化.	27
Java トランスフォーメーションによるプッシュイン最適化.	29
ジョイナトランスフォーメーションの最適化.	30
ルックアップトランスフォーメーションの最適化.	30
ソータトランスフォーメーションの最適化.	33
SQL トランスフォーメーションの最適化.	33
SQL トランスフォーメーションを使用した初期選択の最適化.	34
SQL トランスフォーメーションによるプッシュイン最適化.	34
トランスフォーメーションのキャッシュ.	35
トランスフォーメーションエラーの除去.	36
トランスフォーメーションの副次作用.	36
Web サービスコンシューマトランスフォーメーションの最適化.	37
Web サービスコンシューマトランスフォーメーションによる初期選択の最適化.	38
Web サービスコンシューマトランスフォーメーションによるプッシュイン最適化.	38
第 5 章: マッピングの最適化.	41
マッピングの最適化の概要.	41
最適化方式.	42
最適化レベル.	42
フィルタの最適化.	43
初期プロジェクション最適化方法.	43
述部最適化方式.	44
コストベースの最適化方式.	45
データシップ結合最適化方式.	45
準結合最適化方式.	46
初期選択最適化方式.	47
グローバル述部最適化方式.	47
ブランチ刈り込み最適化方式.	48
プッシュイン最適化方法.	48
プッシュダウンの最適化.	48
完全なプッシュダウンの最適化.	49
ソースプッシュダウン.	50
プッシュダウンの最適化に関するルールとガイドライン.	50

Single-Pass 読み込み.	51
フィルタの最適化.	51
データ型変換の最適化.	52
エラートレース.	52
第 6 章 : パーティション化したマッピングの最適化.	54
パーティション化したマッピングの最適化の概要.	54
複数の CPU の使用.	55
最大並行処理の値の増加.	55
パーティション化に対応するためのフラットファイルの最適化.	56
パーティション化に対応するためのフラットファイルソースの最適化.	56
パーティション化に対応するためのフラットファイルターゲットの最適化.	56
パーティション化に対応するためのリレーショナルデータベースの最適化.	57
パーティション化に対応するためのソースデータベースの最適化.	57
パーティション化に対応するためのターゲットデータベースの最適化.	57
パーティション化に対応するためのトランスフォーメーションの最適化.	58
第 7 章 : 実行時の最適化.	60
実行時の最適化の概要.	60
アプリケーションサービスの最適化.	60
アナリストサービスの最適化.	60
データ統合サービスの最適化.	61
モデルリポジトリサービスの最適化.	62
監視統計.	62
メモリ割り当て.	64
データオブジェクトのキャッシュ.	65
キャッシュテーブルのデータ型.	66
データオブジェクトキャッシュの最適化.	67
システムの最適化.	68
第 8 章 : SQL データサービスの最適化.	69
SQL データサービスの最適化の概要.	69
サードパーティのクライアントツールの最適化.	70
SQL データサービス最適化レベル.	70
データプレビューの SQL データサービス最適化レベルの設定.	71
デプロイされている SQL データサービスの最適化レベルの設定.	71
SQL データサービスのクエリプラン.	72
SQL クエリプランの表示.	73
SQL データサービスのメモリおよび同時要求のプロパティ.	73
SQL データサービスの結果セットキャッシュ.	75
SQL データサービスの結果セットキャッシュのプロパティ.	75
SQL データサービスの結果セットキャッシュ処理の有効化.	76
一時テーブルにおける仮想データの維持.	76

一時テーブルの実装.	76
第 9 章 : Web サービスの最適化.	77
Web サービスの最適化の概要.	77
HTTP 要求の最適化.	78
Web サービスメッセージの圧縮.	78
Web サービス最適化レベル.	78
データプレビューの Web サービス最適化レベルの設定.	79
デプロイされている Web サービスの最適化レベルの設定.	79
Web サービスのメモリおよび同時要求のプロパティ.	80
データ統合サービスの同時 Web サービス要求の設定例.	82
アクティブな DTM インスタンスを設定する Web サービスプロパティ.	82
Web サービスの結果セットキャッシュ処理.	83
Web サービスの結果セットキャッシュ処理の有効化.	83
Web サービスのログ管理.	83
第 10 章 : 接続の最適化.	85
接続の最適化の概要.	85
接続プール.	85
接続オブジェクトのプールのプロパティ.	85
データベースのネットワークパケットサイズ.	86
索引.	88

序文

マッピングパフォーマンスを最適化する方法を学習するには、『*Informatica(R)*パフォーマンスチューニングガイド』を参照してください。各マッピングコンポーネント内のパフォーマンスのボトルネックを特定して除去する方法を確認します。

Informatica のリソース

Informatica は、Informatica Network やその他のオンラインポータルを通じてさまざまな製品リソースを提供しています。リソースを使用して Informatica 製品とソリューションを最大限に活用し、その他の Informatica ユーザーや各分野の専門家から知見を得ることができます。

Informatica Network

Informatica Network は、Informatica ナレッジベースや Informatica グローバルカスタマサポートなど、多くのリソースへの入口です。Informatica Network を利用するには、<https://network.informatica.com> にアクセスしてください。

Informatica Network メンバーは、次のオプションを利用できます。

- ナレッジベースで製品リソースを検索できます。
- 製品の提供情報を表示できます。
- サポートケースを作成して確認できます。
- 最寄りの Informatica ユーザーグループネットワークを検索して、他のユーザーと共同作業を行えます。

Informatica ナレッジベース

Informatica ナレッジベースを使用して、ハウツー記事、ベストプラクティス、よくある質問に対する回答など、製品リソースを見つけることができます。

ナレッジベースを検索するには、<https://search.informatica.com> にアクセスしてください。ナレッジベースに関する質問、コメント、ご意見の連絡先は、Informatica ナレッジベースチーム (KB_Feedback@informatica.com) です。

Informatica マニュアル

Informatica マニュアルポータルでは、最新および最近の製品リリースに関するドキュメントの膨大なライブラリを参照できます。マニュアルポータルを利用するには、<https://docs.informatica.com> にアクセスしてください。

製品マニュアルに関する質問、コメント、ご意見については、Informatica マニュアルチーム (infa_documentation@informatica.com) までご連絡ください。

Informatica 製品可用性マトリックス

製品可用性マトリックス (PAM) には、製品リリースでサポートされるオペレーティングシステム、データベースなどのデータソースおよびターゲットが示されています。Informatica PAM は、<https://network.informatica.com/community/informatica-network/product-availability-matrices> で参照できます。

Informatica Velocity

Informatica Velocity は、Informatica プロフェッショナルサービスが開発したヒントとベストプラクティスのコレクションで、多数のデータ管理プロジェクトから得た実体験に基づいています。Informatica Velocity には、世界中の組織と連携してデータ管理ソリューションを計画、開発、デプロイ、管理する Informatica コンサルタントによる集合知を表しています。

Informatica Velocity リソースには、<http://velocity.informatica.com> からアクセスしてください。Informatica Velocity についての質問、コメント、またはアイデアがある場合は、ips@informatica.com から Informatica プロフェッショナルサービスにお問い合わせください。

Informatica Marketplace

Informatica Marketplace は、お使いの Informatica 製品を拡張したり強化したりするソリューションを検索できるフォーラムです。Marketplace で、Informatica デベロッパーやパートナーからの多数のソリューションを活用すれば、生産性を向上したり、プロジェクトでの実装時間を短縮したりできます。Informatica Marketplace は、<https://marketplace.informatica.com> からアクセスしてください。

Informatica グローバルカスタマサポート

電話または Informatica Network からグローバルサポートセンターに連絡できます。

各地域の Informatica グローバルカスタマサポートの電話番号は、Informatica Web サイト (<https://www.informatica.com/services-and-training/customer-success-services/contact-us.html>) を参照してください。

Informatica Network でオンラインサポートリソースを見つけるには、<https://network.informatica.com> にアクセスし、eSupport オプションを選択します。

第 1 章

パフォーマンスのチューニングの概要

この章では、以下の項目について説明します。

- [パフォーマンスのチューニングの概要, 9 ページ](#)
- [パフォーマンスのチューニングプロセス, 10 ページ](#)
- [ターゲットのボトルネック, 10 ページ](#)
- [ソースのボトルネック, 11 ページ](#)
- [マッピングのボトルネック, 11 ページ](#)
- [コンピュータシステムのボトルネック, 11 ページ](#)
- [ランタイムのボトルネック, 12 ページ](#)
- [SQL データサービスの最適化のボトルネック, 13 ページ](#)
- [Web サービスの最適化のボトルネック, 13 ページ](#)
- [接続ボトルネック, 13 ページ](#)

パフォーマンスのチューニングの概要

パフォーマンスのチューニングの目的は、パフォーマンスのボトルネックを除去することです。ボトルネックとは、スループットが最も低く最も頻繁に実行されるマッピングの領域です。ボトルネックはマッピング全体のパフォーマンスを低下させます。

マッピングを最適化するには、パフォーマンスのボトルネックを特定して除去する作業を繰り返し行います。一度に 1 つのマッピングコンポーネントを最適化します。変更前と変更後にマッピングの時間を計測することで、最適化によるパフォーマンスへの影響を確認することができます。

ソース、ターゲット、接続などのマッピングコンポーネントでボトルネックを特定して修正できます。データ統合サービスのボトルネック、およびデータ統合サービスが実行されているマシンを確認できます。Web サービスおよび SQL データサービスのプロパティを調整することもできます。

パフォーマンスのチューニングプロセス

一連の手順に従って、マッピングコンポーネントを調整し、パフォーマンスを向上させることができます。

次の順序でマッピングコンポーネントを最適化することができます。

1. ターゲット
2. ソース
3. マッピング
4. トランスフォーメーション
5. Administrator ツールの Informatica 環境
6. コンピュータシステム
7. データサービスまたは Web サービス

次の方法でパフォーマンスのボトルネックを特定します。

- テストマッピングを実行する。フラットファイルソースからの読み取りを行うテストマッピングまたはフラットファイルターゲットへの書き込みを行うテストマッピングの際に、ソースおよびターゲットのボトルネックが特定されるようにテストマッピングを設定できます。
- パフォーマンスの詳細を分析する。最適化方式などのパフォーマンスの詳細を分析して、マッピングのパフォーマンスが低下している箇所を特定します。
- システムパフォーマンスを監視する。システム監視ツールを使用して、CPU 使用率、入出力待ち、ページング、およびシステムリソース使用量を表示できます。

ターゲットのボトルネック

ターゲットのボトルネックは、Data Integration Service でターゲットへの書き込みを行う際のパフォーマンスの低下です。ターゲットのボトルネックの原因としては、データベースで使用しているチェックポイント間隔が短い、データベースのネットワークパケットサイズが小さいなどが考えられます。

最も一般的なパフォーマンスのボトルネックは、Data Integration Service がターゲットデータベースへの書き込みを行うときに発生します。データベースのチェックポイント間隔が短いと、チェックポイントの書き込み回数が増え、データベース処理の速度が低下します。データベースのネットワークパケットサイズが小さいと、ボトルネックの原因となります。大きなデータパケットが一度にネットワークを流れるように調整することができます。

ターゲットのボトルネックを特定するには、データベースの代わりにフラットファイルをターゲットにしてマッピングのコピーを作成します。このマッピングでパフォーマンスが著しく向上する場合は、ターゲットにボトルネックがあります。マッピングで既にフラットファイルターゲットに書き込みしている場合、ターゲットにはボトルネックがないと考えられます。

ソースのボトルネック

ソースのボトルネックは、Data Integration Service でソースデータベースから読み取りを行う際のパフォーマンスの低下です。ソースのボトルネックの原因としては、ソースクエリの効率が低い、データベースのネットワークパケットサイズが小さいなどが考えられます。

マッピングでリレーショナルソースからの読み取りを行う場合、次の方法でソースのボトルネックを特定できます。

- マッピングにフィルタトランスフォーメーションを追加する。ソースの後にフィルタトランスフォーメーションを追加し、データが返されないようにフィルタ条件を false に設定します。マッピングの所要時間があまり変わらない場合、ソースにボトルネックがあると考えられます。
- 読み取りテストマッピングを作成する。マッピングのコピーを作成してトランスフォーメーション、結合、クエリをすべて削除し、ソースをターゲットに接続します。マッピングのパフォーマンスが元のマッピングのパフォーマンスとあまり変わらない場合、ソースにボトルネックがあると考えられます。
- ソースデータベースに対して読み取りクエリを直接実行する。マッピングログから読み取りクエリをコピーし、ISQL などのクエリツールを使用してソースデータベースに対してクエリを実行します。クエリの実行時間とクエリが行を返すまでにかかる時間を測定します。

マッピングのボトルネック

ソースまたはターゲットにボトルネックがないことを確認した場合は、マッピングにボトルネックが存在する可能性があります。キャッシュのサイズやバッファのメモリ容量の不足、およびコミットの間隔が短いことが原因で、マッピングのボトルネックが発生することがあります。

マッピングのボトルネックを特定するには、マッピングログでパフォーマンスの詳細を分析します。パフォーマンスの詳細には、入力行、出力行およびエラー行の数など、各トランスフォーメーションに関する情報が含まれます。

また、各ターゲット定義の前にフィルタトランスフォーメーションを追加する方法もあります。フィルタトランスフォーメーションでターゲットテーブルにデータがロードされないように、フィルタ条件を false に設定します。新しいマッピングの実行時間が元のマッピングの実行時間と同じ場合は、マッピングにボトルネックがあると考えられます。

コンピュータシステムのボトルネック

Windows や UNIX で Informatica サービスを実行するとき、リソース使用量を表示することができます。Windows の場合は、タスクマネージャを使用します。UNIX の場合は、いくつかのツールでパフォーマンスを確認できます。

Windows におけるシステムのボトルネックの特定

システムの情報は、タスク マネージャの[パフォーマンス]タブおよび[プロセス]タブで表示できます。タスク マネージャの [パフォーマンス] タブには、CPU の使用率とメモリの総使用量が表示されています。パフォーマンス モニタを使用すると、より詳細な情報が表示されます。

以下の表に、Windows のパフォーマンスモニタでチャートを作成するために使用できるシステム情報を示します。

プロパティ	説明
パーセントプロセッサ時間	複数の CPU がある場合は、各パーセントごとのプロセッサ時間を監視します。
ページ/秒	ページ/秒が 5 を超えている場合、メモリに過大な負担がかかっている可能性があります（この状態をスラッシングと呼びます）。
物理ディスクパーセント時間	読み取りまたは書き込みの要求を実行するときに、物理ディスクがビジーになるパーセント時間を示します。
物理ディスクキュー長	同一のディスクデバイスへのアクセスを待っているユーザーの数を示します。
秒あたりのサーバー処理バイト合計	サーバーとネットワークの間で送信および受信したバイト数を示します。

UNIX におけるシステムのボトルネックの特定

以下のツールを使用して、UNIX でシステムのボトルネックを特定します。

- top。システム全体のパフォーマンスを表示します。このツールでは、システム、およびシステム上で実行されている個々のプロセスに関する CPU 使用率、メモリ使用量およびスワップ使用状況が表示されます。
- iostat。データベースサーバーに取り付けられているすべてのディスクに対するロード操作を監視します。iostat には、ディスクが物理的にアクティブになっている時間の割合が表示されます。ディスクアレイを使用する場合は、iostat は使用しないで、ディスクアレイに同梱されているユーティリティを使用します。
- vmstat。ディスクのスワップ動作を監視します。
- sar。CPU 使用率、メモリ使用量、およびディスク使用状況に関する詳細システムアクティビティレポートを表示します。このツールは、CPU のロードの監視に使用できます。ユーザ、システム、アイドル時間および待ち時間の使用率を知ることができます。このツールは、ディスクのスワップ動作の監視にも使用できます。

ランタイムのボトルネック

パフォーマンス機能を有効にし、マッピングのパフォーマンスが最適になるようにデータ統合サービスのプロパティをチューニングします。データ統合サービスおよびモデルリポジトリサービスの最適化設定を Administrator ツールで設定します。

システムパフォーマンスが最適になるようにメモリを割り当て、データ統合サービスでのマッピングの実行時に生成されるログイベントの数が少なくなるようにエラートレースレベルを設定します。

すべての同時要求の実行用にデータ統合サービスによって割り当てられるメモリの最大量を設定できます。また、特定の要求用にデータ統合サービスによって割り当てられるメモリの最大量を制限することもできます。

結果セットキャッシュを設定すると、各 SQL データサービスクエリおよび Web サービス要求に関連付けられている DTM プロセスの結果をデータ統合サービスでキャッシュできます。

SQL データサービスの最適化のボトルネック

エンドユーザーがサードパーティのクライアントツールを使用して SQL クエリを実行する際のパフォーマンスが向上するように、SQL データサービスを最適化することができます。SQL データサービスで仮想テーブルマッピングを使用する場合は、トランスフォーメーションやマッピングを最適化します。

SQL データサービスのクエリのパフォーマンスを向上させるには、JDBC ドライバを最適化します。また、マッピングや SQL クエリのパフォーマンスを向上させるには、Data Integration Service のデータオブジェクトキャッシュを設定します。

Web サービスの最適化のボトルネック

Data Integration Service で Web サービス要求を実行する際のパフォーマンスが向上するように、Web サービスを最適化することができます。メモリを管理したり、同時 Web サービス要求を処理したり、DTM プロセスをアクティブな状態にして複数の Web サービス要求を処理できるように、Data Integration Service をチューニングします。

Web サービスのパフォーマンスを向上させる方法として、Web サービスメッセージの圧縮、HTTP 要求の最適化、データオブジェクトキャッシュの設定などがあります。

接続ボトルネック

パフォーマンスが向上するように接続を最適化することができます。データベース接続のアイドル状態の接続インスタンスをプールで管理することが可能です。ネットワークパケットのサイズを拡張することで、大きなデータパケットが一度にネットワークを流れるようにすることができます。

第 2 章

ターゲットの最適化

この章では、以下の項目について説明します。

- [ターゲットの最適化の概要, 14 ページ](#)
- [フラットファイルターゲットの最適化, 14 ページ](#)
- [データベースのチェックポイント間隔, 15 ページ](#)
- [バルクロード, 15 ページ](#)
- [データベースターゲットの最適化, 15 ページ](#)

ターゲットの最適化の概要

データ統合サービスによるターゲットへの書き込みを効率よく行えるように、ターゲットを最適化します。マッピングの実行前にインデックスおよびキー制約を削除する、データベースのチェックポイント間隔を長くする、データオブジェクトの書き込みプロパティでバルクロードを設定する、Oracle ターゲットデータベースを最適化するなどの方法があります。

ターゲットを最適化する方法は次のとおりです。

- フラットファイルターゲットを最適化する。
- データベースのチェックポイント間隔を長く設定する。
- バルクロードを使用する。
- Oracle ターゲットデータベースを最適化する。

フラットファイルターゲットの最適化

フラットファイルターゲットを最適化すると、マッピングのパフォーマンスを向上させることができます。また、コマンドにトランスフォーメーションタスクをプッシュしてもパフォーマンスが向上します。

フラットファイルターゲットに関するボトルネックを減らすには、以下の方法を検討します。

データ統合サービスではなくコマンドにトランスフォーメーションタスクをプッシュする。

データ統合サービスではなくコマンドにトランスフォーメーションタスクをプッシュすれば、マッピングのパフォーマンスを向上させることができます。コマンドを使用してターゲットデータのソートや圧縮を行うこともできます。Developer ツールで、フラットファイルターゲットのランタイムプロパティのコマンドプロパティを設定します。

UNIX では、有効な任意の UNIX コマンドまたはシェルスクリプトを使用します。Windows では、有効な任意の DOS コマンドまたはバッチファイルを使用します。フラットファイルライタは、フラットファイルターゲットではなくコマンドにデータを送信します。

例えば、ターゲットデータから圧縮ファイルを生成するには、以下のコマンドを使用します。

```
compress -c - > MyTargetFiles/MyCompressedFile.Z
```

サービスのプロセスノードに対してローカルなフラットファイルターゲットに書き込む。

単一ノードで実行されているデータ統合サービスでフラットファイルターゲットに書き込みを行う場合は、サービスのプロセスノードに対してローカルなフラットファイルターゲットに書き込むと、マッピングのパフォーマンスを最適化できます。

データベースのチェックポイント間隔

Data Integration Service がデータベースでのチェックポイント実行を待つたびに、パフォーマンスが低下します。

データベースのチェックポイントに関するボトルネックを減らすには、以下の方法を検討します。

データベースのチェックポイント間隔を長く設定する。

チェックポイント数を減らしてパフォーマンスを向上させるには、データベースのチェックポイント間隔を大きくします。

チェックポイント数を減らすとパフォーマンスは向上しますが、データベースが予期せずシャットダウンした場合のリカバリ時間も長くなります。

バルクロード

バルクロードを使用すると、Data Integration Service でデータベースログが無視されるため、パフォーマンスが向上します。

バルクロードに関するボトルネックを減らすには、以下の方法を検討します。

データオブジェクトの書き込みプロパティでバルクロードを設定する。

バルクロードの使用により、DB2、Sybase ASE、Oracle、または Microsoft SQL Server のデータベースに大量データを挿入するマッピングのパフォーマンスを向上させることができます。

データベースログへの書き込みは行われないので、ターゲットデータベースではロールバックを実行できません。その結果、リカバリを実行できない場合があります。バルクロードを実行する場合は、マッピングのパフォーマンスを向上させる機能と、不完全なマッピングをリカバリする機能のどちらを重視するかを検討する必要があります。

データベースターゲットの最適化

ストレージ句、スペース割り当て、ロールバックまたは取り消しセグメントのチェックによりターゲットデータベースを最適化できます。

データベースターゲットに関するボトルネックを減らすには、次の方法を検討します。

データベースでロールバックまたは取り消しセグメントがそれぞれ適切なテーブルスペース（別々のディスクが望ましい）に保存されていることを確認します。

データベースへの書き込みを行うとき、データベースではロード時にロールバックまたは取り消しセグメントを使用します。データベース管理者に依頼して、データベースによってロールバックまたは取り消しセグメントがそれぞれ適切なテーブルスペース（別々のディスクが望ましい）に格納されていることを確認します。また、ロールバックまたは取り消しセグメントには、適切なストレージ句が必要です。

データベースの REDO ログをチューニングします。

データベースを最適化するには、データベースの REDO ログをチューニングします。データベースでは、REDO ログを使用してロード操作を記録します。REDO ログおよびバッファのサイズが最適であることを確認します。Oracle データベースでは、init.ora ファイルで REDO ログのプロパティを表示できます。

IPC プロトコルの Oracle データベースに接続します。

データ統合サービスが単一ノードで実行されており、Oracle インスタンスがサービスのプロセスノードに対してローカルである場合、IPC プロトコルを使用して Oracle データベースに接続することによって、パフォーマンスを最適化できます。Oracle データベースへの接続は、listener.ora および tnsnames.ora で設定できます。

第 3 章

ソースの最適化

この章では、以下の項目について説明します。

- [ソースの最適化の概要, 17 ページ](#)
- [フラットファイルソースの最適化, 18 ページ](#)
- [クエリの最適化, 18 ページ](#)
- [条件フィルタ, 19 ページ](#)
- [個別に選択, 19 ページ](#)
- [ヒント, 19 ページ](#)
- [制約, 21 ページ](#)
- [カスタマイズデータオブジェクトの最適化, 22 ページ](#)
- [データベースソースの最適化, 23 ページ](#)

ソースの最適化の概要

データ統合サービスによるソースデータの読み取りを効率よく行えるように、フラットファイルソース、リレーショナルソース、およびカスタムデータソースを最適化します。

ソースを最適化する方法は次のとおりです。

- ソースデータを効率よく読み取る。
- クエリの最適化の方法を使用する。
- SQL クエリで条件フィルタを使用する。
- ソースから一意の値を選択する。
- SQL クエリにヒントを適用する。
- 論理データオブジェクト、物理データオブジェクト、および仮想テーブルに制約を設定する。
- カスタマイズデータオブジェクトの設定を最適化する。
- Oracle、Sybase、および Microsoft SQL Server データベースの設定を最適化する。

フラットファイルソースの最適化

Data Integration Service によるソースデータの読み取りを効率よく行えるように、フラットファイルソースのフォーマットのプロパティを設定します。

フラットファイルソースに関するボトルネックを減らすには、以下の方法を検討します。

区切りフラットファイルのフォーマットのプロパティで引用符やエスケープ文字を使用しない。

エスケープ文字を指定すると、Data Integration Service は、文字列に含まれる通常の文字として区切り文字を読み取ります。ソースファイルに引用符やエスケープ文字が含まれていないと、マッピングのパフォーマンスをある程度向上できます。

Data Integration Service によって読み取られる行ごとのバイト数を設定する。

マッピングでフラットファイルソースからの読み取りを行う場合は、Data Integration Service によって読み取られる行ごとのバイト数を設定することにより、マッピングのパフォーマンスを向上させることができます。フラットファイルソースのランタイムプロパティで「連続行のバッファ長」プロパティを設定します。

デフォルトでは、Data Integration Service は 1 行につき 1024 バイトを読み取ります。ソースファイル内の各行のバイト数がデフォルト設定の値より少ない場合、マッピングのプロパティで連続行のバッファ長の値を低くすることができます。

クエリの最適化

マッピングで 1 つのカスタマイズデータオブジェクト内にある複数のソーステーブルを結合する場合、最適化のヒントを利用してクエリを最適化することによって、パフォーマンスを向上させることがあります。また、ORDER BY 句または GROUP BY 句を使用した単独テーブル選択文に対しては、インデックスの追加などの最適化が役立ちます。

クエリに関するボトルネックを減らすには、以下の方法を検討します。

ソーステーブルの特定のセットに対するクエリの実行方法をデータベースに指示するためのオプティマイザヒントを作成する。

通常、データベースオプティマイザでは、ソースデータを処理する際の最も効率的な方法を判定します。ただし、ユーザはデータベースのオプティマイザが認識できないソーステーブルの属性を知っている場合があります。データベース管理者は、ソーステーブルの特定のセットに対するクエリの実行方法をデータベースに指示するためのオプティマイザヒントを作成できます。

オプティマイザヒントが、すべてのレコードを一度に返すのではなく、できるだけ早くレコードを返すように設定する。

クエリの実行を開始してからデータ統合サービスがデータの最初の行を受け取るまでの遅延が長い場合は、最適化のヒントを使用します。オプティマイザヒントが、すべてのレコードを一度に返すのではなく、できるだけ早くレコードを返すように設定する。これにより、データ統合サービスは、クエリの実行と並行してレコードを処理できるようになります。

ORDER BY カラムまたは GROUP BY カラムにインデックスを作成する。

ORDER BY カラムまたは GROUP BY カラムにインデックスを作成することにより、ORDER BY 句または GROUP BY 句を含むクエリの効率が向上します。クエリを最適化したら、SQL オーバーライドオプションを使用して、最適化の際の修正が最大限に活用されるようにします。

データベースで並行クエリを実行できるように設定する。

また、ソースデータベースで並行クエリを実行できるように設定して、パフォーマンスを向上させることもできます。平行クエリの設定の詳細については、データベースのマニュアルを参照してください。

データ統合サービスでデータの読み取りに使用されるクエリは、SQL データサービスの仮想テーブルに表示されます。また、カスタマイズデータオブジェクトでもクエリを確認できます。データベース管理者にクエリの実行を依頼し、オプティマイザヒントやソーステーブルのインデックスを作成します。

条件フィルタ

インデックスが設定されていないことが原因で、ソースデータベース上の簡単なソースフィルタが、パフォーマンスに悪影響を及ぼすことがあります。カスタマイズデータオブジェクトで条件フィルタを使用すると、パフォーマンスを向上させることができます。

条件フィルタに関するボトルネックを減らすには、以下の方法を検討します。

複数のマッピングで同一のソースから同時に読み取りを行う場合は条件フィルタを使用する。

複数のマッピングで同一のソースから同時に読み取りを行う場合、条件フィルタでパフォーマンスを向上させることができます。

ただし、ソースデータベース上のソースデータにフィルタをかける方が、実行速度が速くなるマッピングもあります。どの方法でパフォーマンスが向上するかを判定するには、データベースフィルタおよび条件フィルタの両方でマッピングをテストします。

個別に選択

〔個別選択〕 オプションを使用して、カスタマイズデータオブジェクトのソースから一意の値を選択できます。〔個別選択〕 を使用すると、Data Integration Service は SELECT DISTINCT 文をデフォルトの SQL クエリに追加します。

個別選択に関するボトルネックを減らすには、以下の方法を検討します。

〔個別選択〕 オプションを使用して、データフローの初期段階で不必要なデータにフィルタをかける。

Data Integration Service がソースから一意な値を選択するようにしたい場合は、カスタマイズデータオブジェクトに対して〔個別選択〕 オプションを使用します。〔個別選択〕 オプションを使用して、データフローの初期段階で不必要なデータにフィルタをかける。これによってパフォーマンスが向上します。

例えば、〔個別選択〕 オプションを使用して、総売上高の一覧のテーブルから一意の顧客 ID を抽出できます。カスタマイズデータオブジェクトをマッピングで使用すると、Data Integration Service はデータフローの初期段階で不必要なデータを除外するため、パフォーマンスが向上します。

ヒント

データベースオプティマイザに対する指示として、ソースの SQL クエリにヒントを追加することができます。オプティマイザはヒントを使用して、ソースにアクセスするためのクエリ実行プランを選択します。

〔ヒント〕 フィールドは、リレーショナルデータオブジェクトインスタンスまたはカスタマイズデータオブジェクトの〔クエリ〕ビューに表示されます。ソースデータベースは、Oracle、Sybase、IBM DB2、または Microsoft SQL Server でなければなりません。それ以外のデータベースタイプでは、〔ヒント〕 フィールドは表示されません。

Data Integration Service によるソースクエリの生成時に、Developer ツールで入力した SQL ヒントがクエリに追加されます。ヒントは Data Integration Service では解析されません。ソースを含むマッピングを実行したときに、マッピングログにクエリとヒントが表示されます。

Data Integration Service によって SQL ヒントが挿入されるクエリ内の位置は、データベースタイプによって異なります。ヒントの構文については、データベースのマニュアルを参照してください。

Oracle

ヒントが SELECT/UPDATE/INSERT/DELETE キーワードのすぐ後に追加されます。

```
SELECT /*+ <hints> */ FROM ...
```

'+'はヒントの先頭を示します。

ヒントはコメント（/* ... */または--...行の末尾まで）に格納されます。

Sybase

ヒントがクエリのすぐ後に追加されます。ヒントでプラン名を設定します。

```
SELECT ... PLAN <plan>
```

```
select avg(price) from titles plan "(scalar_agg (i_scan type_price_ix titles ))"
```

IBM DB2

ヒントとして optimize-for 句を入力できます。この句はクエリの末尾に追加されます。

```
SELECT ... OPTIMIZE FOR <n> ROWS
```

optimize-for 句では、クエリで処理する行数をデータベースオプティマイザに示します。これは行数の制限ではありません。データベースで処理する行数が<n>行を超えると、パフォーマンスが低下する可能性があります。

Microsoft SQL Server

ヒントがクエリの末尾に OPTION 句の一部として追加されます。

```
SELECT ... OPTION ( <query_hints> )
```

ヒントに関するルールとガイドライン

SQL クエリのヒントを設定するときは、次のルールとガイドラインに従います。

- プッシュダウンの最適化を有効にする場合やリレーショナルデータオブジェクトで準結合を使用する場合は、元のソースクエリが変わります。変更後のクエリにはヒントは適用されません。
- ヒントはジョインやフィルタのオーバーライドと組み合わせることができますが、SQL のオーバーライドを設定した場合は SQL のオーバーライドが優先され、他のオーバーライドは適用されません。
- **【クエリ】** ビューには、簡易ビューと詳細ビューがあります。簡易ビューでフィルタ、ソート、またはジョインのオーバーライドを使用してヒントを入力した場合、完全なクエリオーバーライドは詳細ビューに表示されます。

ヒントの作成

データベースオプティマイザにクエリプランの決定に関する指示を伝えるには、ヒントを作成します。

1. カスタマイズデータオブジェクトまたはリレーショナルデータオブジェクトインスタンスを開きます。
2. **【読み取り】** ビューを選択します。
3. **【出力トランスフォーメーション】** を選択します。

4. **【クエリ】** プロパティを選択します。
5. 単純クエリを選択します。
6. **【ヒント】** フィールドの横にある **【編集】** をクリックします。
【ヒント】 ダイアログボックスが表示されます。
7. **【SQL クエリ】** フィールドにヒントを入力します。
Developer ツールでは、ヒントは検証されません。
8. **【OK】** をクリックします。
9. データオブジェクトを保存します。

制約

データ統合サービスは、リレーショナルソース、フラットファイルソース、論理データオブジェクト、または仮想テーブルから制約を読み取ることができます。制約は、データ行上の値が合致する必要がある条件式です。

制約を読み取るときにデータ統合サービスは、適用されている最適化方式に基づき、データ行に関して TRUE に評価されない行を削除することがあります。

制約を設定する前に、その制約で設定される条件をソースデータが満たすことを確認する必要があります。

例えば、「AGE < 70」の行を含んでいると思われる AGE 列がソースデータベースに存在するとします。このソースデータベースに、「AGE < 70」という式で制約を設定できます。データ統合サービスは、ソースデータベースから制約「AGE < 70」に合致するレコードを読み取ります。「AGE >= 70」のレコードを読み取る場合、データ統合サービスはそれらを削除することがあります。

データベースでは、SQL コマンドを使用して、データベースに接続する際のデータベース環境に制約を設定できます。データ統合サービスは、データベースに接続するたびに、接続環境 SQL を実行します。

論理データオブジェクト、物理データオブジェクト、および仮想テーブルへの制約の設定には、Developer ツールを使用します。制約を設定する場合は、各データ行に関して TRUE に評価される式を入力する必要があります。

制約の設定

リレーショナルデータオブジェクト、フラットファイルデータオブジェクト、カスタマイズされたデータオブジェクト、論理データオブジェクト、および仮想テーブルに対して制約を追加できます。制約を追加した後で、その制約を編集したり、削除したりできます。

1. **【オブジェクトエクスプローラ】** ビューから、トランスフォーメーションの読み取りとして追加されたリレーショナルデータオブジェクトが含まれているマッピングを開きます。あるいは、フラットファイルデータオブジェクト、カスタマイズされたデータオブジェクト、論理データオブジェクト、または仮想テーブルを開きます。
 - トランスフォーメーションの読み取りとしてマッピングに追加されたリレーショナルデータオブジェクトに制約を設定するには、マッピングでトランスフォーメーションの読み取りを選択します。**【プロパティ】** ビューで **【詳細】** タブを選択します。
 - フラットファイルデータオブジェクトに制約を設定するには、**【詳細】** ビューを選択して **【ランタイム: 読み取り】** セクションを展開します。

- カスタマイズされたデータオブジェクトに制約を設定するには、**【読み込み】** ビューを選択し、ソーストランスフォーメーションの **【出力】** ポートを選択します。**【プロパティ】** ビューで **【詳細】** タブを選択します。
 - 論理データオブジェクトに制約を設定するには、論理データモデルを選択し、論理データオブジェクトを選択します。**【プロパティ】** ビューで **【詳細】** タブを選択します。
 - 仮想テーブルに制約を設定するには、SQL エンドポイントから仮想テーブルを開きます。**【プロパティ】** ビューで **【詳細】** タブを選択します。
2. 制約の値フィールドをクリックします。
【制約】 ダイアログボックスが表示されます。
 3. **【新規】** をクリックし、式エディタを開きます。
 4. 制約ロジックを設定し、式関数とカラムをパラメータとして使用します。
 5. **【検証】** をクリックします。
 6. **【OK】** をクリックします。

カスタマイズデータオブジェクトの最適化

カスタマイズデータオブジェクトの設定によってパフォーマンスを向上させることができます。カスタマイズデータオブジェクトで、SQL クエリを最適化する、条件フィルタを使用する、ソースから個別の値を選択する方法があります。

カスタマイズデータオブジェクトに関するボトルネックを減らすには、以下の方法を検討します。

特殊な SELECT 文を発行するカスタムクエリを作成して、データ統合サービスにソースデータの読み取りを行わせる。

カスタムクエリは、データ統合サービスがソースからデータを読み取るために使用するデフォルトクエリに置き換わります。

データ統合サービスがソースデータを読み取るときに行をフィルタする。

フィルタ条件を含めると、データ統合サービスはデフォルトクエリに WHERE 句を追加します。

ソースから重複しない値を選択する。

【個別選択】 を選択すると、データ統合サービスは SELECT DISTINCT 文をデフォルトの SQL クエリに追加します。

データベースのヒントを適用する。

データベースオブティマイザに対する指示として、ソースの SQL クエリにヒントを追加することができます。

ソースデータに制約を設定する。

カスタマイズされたデータオブジェクト内のフラットファイルおよびリレーショナルテーブルに制約を設定する場合、データ統合サービスはデータ行に対して TRUE に評価されない行を削除します。

データベースソースの最適化

ソースデータベースが Oracle の場合、Oracle データベースへの接続に IPC プロトコルを使用すると、データ統合サービスのパフォーマンスを向上させることができます。一時データベースをディスクアレイに移動してパフォーマンスを向上させることもできます。

データベースソースに関するボトルネックを減らすには、以下の方法を検討します。

Oracle データベースへの接続には IPC プロトコルを使用する。

データ統合サービスが単一ノードで実行されており、Oracle インスタンスがサービスのプロセスノードに対してローカルである場合、IPC プロトコルを使用して Oracle データベースに接続することによって、パフォーマンスを最適化できます。Oracle データベースへの接続は、listener.ora および tnsnames.ora で設定できます。

一時データベースおよび REDO ログをディスクアレイまたはより速いドライブに移動します。

データベース上の大きいテーブルを結合する場合は、キャッシュの場所として RAID（Redundant Array of Inexpensive Disks）を使用できます。または、他のディスクのプライマリファイルグループにより多くのファイルを追加してディスク間のロードを配分することができます。

第 4 章

トランスフォーメーションの最適化

この章では、以下の項目について説明します。

- [トランスフォーメーションの最適化, 24 ページ](#)
- [アグリゲータトランスフォーメーションの最適化, 25 ページ](#)
- [式の最適化, 25 ページ](#)
- [Java トランスフォーメーションの最適化, 27 ページ](#)
- [ジョイナトランスフォーメーションの最適化, 30 ページ](#)
- [ルックアップトランスフォーメーションの最適化, 30 ページ](#)
- [ソータトランスフォーメーションの最適化, 33 ページ](#)
- [SQL トランスフォーメーションの最適化, 33 ページ](#)
- [トランスフォーメーションのキャッシュ, 35 ページ](#)
- [トランスフォーメーションエラーの除去, 36 ページ](#)
- [トランスフォーメーションの副次作用, 36 ページ](#)
- [Web サービスコンシューマトランスフォーメーションの最適化, 37 ページ](#)

トランスフォーメーションの最適化

Data Integration Service によるマッピングのトランスフォーメーションの処理を効率よく行えるように、トランスフォーメーションを最適化します。

トランスフォーメーションを最適化する方法は次のとおりです。

- トランスフォーメーションの設定を最適化する。
- トランスフォーメーションエラーを除去する。
- トランスフォーメーションのキャッシュを設定する。

アグリゲータトランスフォーメーションの最適化

Aggregator トランスフォーメーションではデータを処理する前にそれをグループ化しなければならないため、パフォーマンスが低下することがよくあります。Aggregator トランスフォーメーションは、中間のグループ結果を格納するための追加メモリを必要とします。

アグリゲータトランスフォーメーションに関するボトルネックを減らすには、以下の方法を検討します。

単純なカラム別にグループ化する。

単純なカラム別のグループ化を行う際は、Aggregator トランスフォーメーションを最適化することができます。可能な場合には、GROUP BY で使用するカラムでは、文字列や日付ではなく数値を使用します。Aggregator の式が複雑にならないようにします。

ソート済み入力を使用する。

マッピングのパフォーマンスを向上させるには、アグリゲータトランスフォーメーションのデータをソートします。データのソートには、Sorted Input オプションを使用します。

Sorted Input オプションは、集計キャッシュの使用頻度を低くします。[ソート済み入力] オプションを使用すると、データ統合サービスはすべてのデータがグループ別にソートされているものと想定します。データ統合サービスは1つのグループに対する行を読み取るときに、読み取りながら集計計算を実行します。必要に応じて、メモリにグループ情報を格納します。

[ソート済み入力] オプションは、マッピング中にキャッシュに格納されるデータ量を減らして、パフォーマンスを向上させます。[ソート済み入力] オプションまたはソータトランスフォーメーションを使用して、ソート済みデータをアグリゲータトランスフォーメーションに渡します。

複数のパーティションが含まれるマッピングで [ソート済み入力] オプションを使用すると、パフォーマンスが向上します。

集計する前にデータをフィルタリングする。

マッピングでフィルタトランスフォーメーションを使用する場合、そのあとにアグリゲータトランスフォーメーションを置いて、不要な集計を減らしてください。

ポート接続数を制限する。

接続される入出力ポートまたは出力ポートの数を制限することにより、アグリゲータトランスフォーメーションがデータキャッシュに格納するデータ量を減らしてください。

式の最適化

トランスフォーメーションで使用される一部の式によってパフォーマンスが低下することがあります。

式に関するボトルネックを減らすには、以下の方法を検討します。

パフォーマンスを低下させている式を切り離す。

一部の式によってマッピングのパフォーマンスが低下することがあります。パフォーマンスを低下させている式を切り離すには、マッピングからそれらの式を1度に1つずつ削除してマッピングを実行し、式がない場合のマッピングの実行所要時間を調べます。マッピングの実行時間が著しく異なる場合は、パフォーマンスを低下させている式を最適化する方法を探します。

式のパフォーマンスを評価する次のステップを実行します。

1. 元の式を使用してマッピングの時間を計測します。
2. マッピングをコピーし、複雑な式の半分を定数に置き換えます。

3. 編集したマッピングを実行して時間を計測します。
4. マッピングのコピーをもう1つ作成し、複雑な式の残りの半分を定数で置き換えます。
5. 編集したマッピングを実行して時間を計測します。

共通のロジックを減らす。

マッピングによって複数の箇所で同じ処理が実行される場合、その処理をマッピングの初期段階に移すことによって、マッピングによる処理の実行回数を減らします。たとえば、マッピングに5つのターゲットテーブルがある場合を検討しましょう。各ターゲットには、社会保障番号のルックアップが必要です。この場合、ルックアップを5回実行する代わりに、データフローが分かれる前のマッピングに Lookup トランスフォーメーションを配置します。次に、ルックアップの結果を5つのターゲットすべてに渡します。

集計関数の呼び出しを最小限にする。

式を記述するときには、集計関数の呼び出しをできる限り少なくします。集計関数が呼び出されるたびに、Data Integration Service ではデータの検索とグループ化を行う必要が生じます。例えば、次の式では、Data Integration Service は最初に COLUMN_A を読み込んでその合計を求めます。次に、COLUMN_B を読み込んでその合計を求め、最後に2つの合計の和を求めます。

```
SUM(COLUMN_A) + SUM(COLUMN_B)
```

次の例のように集計関数の呼び出しを減らした場合、Data Integration Service は COLUMN_A を COLUMN_B に足した後でその合計を求めます。

```
SUM(COLUMN_A + COLUMN_B)
```

共通の式をローカル変数で置き換える。

1つのトランスフォーメーションにおいて同じ式が複数回使用されている場合は、その部分式をローカル変数にすることができます。ローカル変数はトランスフォーメーション内でのみ使用できます。ただし、変数の計算は一度で済むので、パフォーマンスを向上させることができます。

数値演算と文字列演算のどちらを使用するかを選択する。

Data Integration Service では、数値演算の方が文字列演算よりも高速に処理されます。例えば、2つの列 (EMPLOYEE_NAME と EMPLOYEE_ID) で大量のデータを検索する場合、EMPLOYEE_ID にルックアップを設定することでパフォーマンスが改善されます。

CHAR 対 CHAR および CHAR 対 VARCHAR の比較を最適化する。

Data Integration Service が CHAR 列と VARCHAR 列の比較を行う場合、行の末尾に空白が検出されるたびに処理速度が低下します。Informatica Administrator で Data Integration Service を設定する場合に、TreatCHARasCHARonRead オプションを使用すると、Data Integration Service は Char 型ソースフィールドの末尾の空白を削除しません。

DECODE と LOOKUP のどちらを使用するかを選択する。

LOOKUP 関数を使用すると、Data Integration Service ではデータベース内のテーブルを検索する必要が生じます。DECODE 関数を使用するとルックアップ値が式自体に取り込まれ、Data Integration Service が個々のテーブルを検索する必要がなくなります。したがって、ルックアップする値が変化せず、その数も少ない場合は、DECODE を使用するとパフォーマンスが向上します。

関数の代わりに演算子を使用する。

Data Integration Service では、関数を使用して書かれた式よりも、演算子を使用して書かれた式の方が高速に読み込まれます。したがって、できるだけ演算子を使用して式を記述してください。たとえば、CONCAT 関数をネストした次のような式があるとします。

```
CONCAT( CONCAT( CUSTOMERS.FIRST_NAME, ' ') CUSTOMERS.LAST_NAME)
```

|| 演算子を使用すると、上記の式を次のように表すことができます。

```
CUSTOMERS.FIRST_NAME || ' ' || CUSTOMERS.LAST_NAME
```

IIF 関数を最適化する。

IIF 関数では、値およびアクションを返すことができるため、式をより簡潔にできます。たとえば、FLG_A、FLG_B、FLG_C という 3 つの Y/N フラグを含むソースがあり、各フラグの値に基づく値を返したいとします。

次の式を使用します。

```
IIF( FLG_A = 'Y' and FLG_B = 'Y' AND FLG_C = 'Y',  
VAL_A + VAL_B + VAL_C,  
IIF( FLG_A = 'Y' and FLG_B = 'Y' AND FLG_C = 'N',  
VAL_A + VAL_B ,  
IIF( FLG_A = 'Y' and FLG_B = 'N' AND FLG_C = 'Y',  
VAL_A + VAL_C ,  
IIF( FLG_A = 'Y' and FLG_B = 'N' AND FLG_C = 'N',  
VAL_A ,  
IIF( FLG_A = 'N' and FLG_B = 'Y' AND FLG_C = 'Y',  
VAL_B + VAL_C ,  
IIF( FLG_A = 'N' and FLG_B = 'Y' AND FLG_C = 'N',  
VAL_B ,  
IIF( FLG_A = 'N' and FLG_B = 'N' AND FLG_C = 'Y',  
VAL_C ,  
IIF( FLG_A = 'N' and FLG_B = 'N' AND FLG_C = 'N',  
0.0 ,  
))))))
```

Java トランスフォーメーションの最適化

マッピングの一部の Java トランスフォーメーションにより、パフォーマンスが低下することがあります。

Java トランスフォーメーションのパフォーマンスを向上させるには、以下の方法を検討します。

Java トランスフォーメーションで初期選択または最適化にプッシュインのフィルタ最適化方式を有効にする。

Java トランスフォーメーションでは、初期選択の最適化または最適化にプッシュインを有効にできます。

Java トランスフォーメーションの **【最適化インタフェース】** タブで、コードスニペットを更新します。

Java トランスフォーメーションによる初期選択の最適化

Java トランスフォーメーションに副次作用がない場合は、初期選択の最適化のアクティブまたはパッシブな Java トランスフォーメーションを有効化できます。オプティマイザは、Java トランスフォーメーションを介してフィルタロジックを渡し、必要に応じてフィルタ条件を変更します。

初期選択の最適化のコードスニペットを表示するには、**【最適化インタフェース】** タブのナビゲータで、PredicatePushOptimization を選択します。

allowPredicatePush

ブール型。初期選択を有効にします。初期選択を有効にするために True の結果とメッセージを返すようにこの関数を変更します。デフォルトは False で、最適化がサポートされていないというメッセージが返されません。

```
public ResultAndMessage allowPredicatePush(boolean ignoreOrderOfOp) {  
    // To Enable PredicatePushOptimization, this function should return true  
    //return new ResultAndMessage(true, "");  
    return new ResultAndMessage(false, "Predicate Push Optimization Is Not Supported");  
}
```

canGenerateOutputFieldEvalError

ブール型。Java トランスフォーメーションがゼロ除算エラーなどの出力フィールドエラーを返すことができるかどうかを示します。Java トランスフォーメーションが出力フィールドエラーを生成しない場合は False

を返すようにこの関数を変更します。Java トランスフォーメーションがフィールドエラーを生成できるとき、Data Integration Service では初期選択の最適化は使用できません。

```
public boolean canGenerateOutputFieldEvalError() {
    // If this Java transformation can never generate an output field evaluation error,
    // return false.
    return true;
}
```

getInputExpr

入力フィールドのどの入力値が出力フィールドを構成しているかを示す Informatica の式を返します。オプティマイザは、Java トランスフォーメーションを介してフィルタロジックをプッシュするために、出力フィールドを構成している入力フィールドを認識する必要があります。

```
public InfaExpression getInputExpr(TransformationField field,
    TransformationDataInterface group) {
    // This should return an Informatica expression for output fields in terms of input fields
    // We will only push predicate that use fields for which input expressions are defined.
    // For example, if you have two input fields in0 and in1 and three output fields out0, out1, out2
    // out0 is the pass-through of in1, out2 is sum of in1 and in2, and out3 is unknown, the code should be:
    //if (field.getName().equals("out0"))
    //    return new InfaExpression("in0", instance);
    //else if (field.getName().equals("out1"))
    //    return new InfaExpression("in0 + in1", instance);
    //else if (field.getName().equals("out2"))
    //    return null;
    return null;
}
```

例えば、マッピングにフィルタ式が含まれている場合は、"out0 > 8"です。out0 は、Java トランスフォーメーションの out0 出力ポートの値です。out0 の値を in0 入力ポート+5 の値として定義できます。オプティマイザは、式"(in0 + 5) > 8"を初期選択の最適化の Java トランスフォーメーション以降にプッシュできます。出力フィールドに入力フィールドの式がない場合は、NULL を返すことができます。オプティマイザでは、入力式がない出力フィールド以降にフィルタ式がプッシュされることはありません。

次のコードを含めることができます。

```
if (field.getName().equals("out0"))
    return new InfaExpression("in0 + 5", instance);
else if (field.getName().equals("out2"))
    return null;
```

inputGroupsPushPredicateTo

フィルタロジックを受け取ることができるグループのリストを返します。Java トランスフォーメーションには入力グループが 1 つあります。Java トランスフォーメーションのこの関数は変更しないでください。

```
public List<TransformationDataInterface> inputGroupsPushPredicateTo(
    List<TransformationField> fields) {
    // This functions returns a list of input data interfaces to push predicates to.
    // Since JavaTx only has one input data interface, you should not have to modify this function
    AbstractTransformation tx = instance.getTransformation();
    List<DataInterface> dis = tx.getDataInterfaces();
    List<TransformationDataInterface> inputDIs = new ArrayList<TransformationDataInterface>();
    for (DataInterface di : dis){
        TransformationDataInterface tdi = (TransformationDataInterface) di;
        if (tdi.isInput())
            inputDIs.add(tdi);
    }
    if(inputDIs.size() == 1)
        return inputDIs;
    else
        return null;
}
```

Java トランスフォーメーションによるプッシュイン最適化

副次作用がなく、最適化がマッピング結果に影響しない場合は、最適化にプッシュインでアクティブ Java トランスフォーメーションを有効にできます。

Java トランスフォーメーションで最適化にプッシュインを設定するときは、Java トランスフォーメーションがオブティマイザから受け取るフィルタ条件を格納する方法を定義します。フィルタ条件を調べるコードを追加します。Java トランスフォーメーションがフィルタロジックを吸収できる場合、Java トランスフォーメーションは True の条件をオブティマイザに戻します。オブティマイザは、最適化されたマッピングからフィルタトランスフォーメーションを削除します。

Java トランスフォーメーションを設定するときは、最適化中にフィルタ条件をトランスフォーメーションのメタデータとして格納するコードを記述します。実行時にフィルタ条件を取得するコードや、フィルタロジックに従って行を削除するコードも記述します。

Java トランスフォーメーションを定義するときは、Java トランスフォーメーションの【最適化インタフェース】タブで、最適化にプッシュインのコードを追加します。最適化にプッシュインのコードスニペットにアクセスするには、【最適化インタフェース】タブのナビゲータで、FilterPushdownOptimization を選択します。

最適化にプッシュインを有効にするコードスニペットや、オブティマイザからフィルタ条件を取得するコードスニペットが表示されます。最適化を有効にしたり、フィルタロジックをトランスフォーメーションのメタデータとして保存したりするには、コードスニペットを更新します。

isFilterSupported

最適化にプッシュインを有効にする場合は、True を返します。最適化にプッシュインを無効にする場合は、False を返します。

最適化にプッシュインを有効にするために True を返すようにこの関数を変更します。

```
public ResultAndMessage isFilterSupported() {
    // To enable filter push-into optimization this function should return true
    // return new ResultAndMessage(true, "");
    return new ResultAndMessage(false, "Filter push-into optimization is not supported");
}
```

pushFilter

オブティマイザからフィルタ条件を受け取ります。

フィルタを調べ、トランスフォーメーションでフィルタロジックが使用可能かどうかを判断するコードを追加します。Java トランスフォーメーションがフィルタを吸収できる場合は、次のメソッドを使用してフィルタ条件をトランスフォーメーションのメタデータとして格納します。

storeMetadata (String key、String data)

このキーは、メタデータの識別子です。任意の文字列をキーとして定義できます。このデータは、実行時に削除する行を指定するために格納するデータです。例えば、このデータは、Java トランスフォーメーションがオブティマイザから受け取るフィルタ条件である可能性があります。

```
public ResultAndMessage pushFilter(InfraExpression condition) {
    // Add code to absorb the filter
    // If filter is successfully absorbed return new ResultAndMessage(true, ""); and the optimizer
    // will remove the filter from the mapping
    // If the filter is not absorbed, return new ResultAndMessage(false, msg);
    return new ResultAndMessage(false, "Filter push-into optimization is not supported");
}
```

ジョイナトランスフォーメーションの最適化

ジョイナトランスフォーメーションでは、中間結果を格納するための追加領域を実行時に必要とするため、パフォーマンスが低下することがあります。

ジョイナトランスフォーメーションに関するボトルネックを減らすには、以下の方法を検討します。

重複キー値が少ない方のソースをマスタとして指定します。

データ統合サービスは、ソート済みジョイナトランスフォーメーションを処理するときに、一度に 100 個の一意なキーの行をキャッシュに格納します。マスタソースに同じキー値を持つ多数の行が含まれる場合、データ統合サービスはより多くの行をキャッシュに格納する必要があり、それによってパフォーマンスが低下することがあります。

行数が少ない方のソースをマスタとして指定します。

ジョイナトランスフォーメーションは詳細ソースの各行をマスタソースと比較します。マスタ内の行が少なければ、結合のための比較が繰り返される回数も少なくなり、その結果、結合プロセスが高速になります。

可能な場合は、データベース内で結合を実行します。

データベース内で結合を実行すると、マッピングの実行中に実行する場合よりも処理が高速になります。パフォーマンスは、使用するデータベース結合の種類によっても変わってきます。ノーマル結合は、外部結合よりも高速で、結果的にレコード数が少なくて済みます。場合によっては、例えば 2 つの異なるデータベースまたはフラットファイルシステムとテーブルを結合する場合は、これが不可能なこともあります。

可能な場合は、ソート済みデータを結合します。

ソート済み入力を使用するようにジョイナトランスフォーメーションを設定してください。データ統合サービスは、ディスクの入出力を最小化することによってパフォーマンスを向上させます。パフォーマンスは、大量のデータセットを扱う場合に最大限に向上させることができます。未ソートジョイナトランスフォーメーションの場合、行の比較的少ないソースをマスタソースとして指定します。

結合条件を最適化する。

データ統合サービスは、小さい方のグループから行を読み取り、大きい方のグループで一致する行を見つけて結合操作を実行することで、1 つの結合オペランドのデータセットのサイズを小さくしようとします。データセットのサイズを小さくすると、データ統合サービスで大きい方のグループソースから不要な行が読み取られなくなるため、マッピングのパフォーマンスが向上します。データ統合サービスによって、結合条件が大きい方のグループソースに移動され、小さい方のグループと一致する行のみが読み取られます。

準結合最適化方式を使用する。

一方の入力グループに他方よりも多くの行が含まれており、結合条件に基づいて、小さい方のグループに一致するものがない行が大きい方のグループに多数含まれている場合は、準結合最適化方式を使用するとマッピングのパフォーマンスが向上します。

ルックアップトランスフォーメーションの最適化

ルックアップトランスフォーメーションでは、ルックアップキャッシュのタイプやルックアップ条件によって、パフォーマンスが低下することがあります。

ルックアップトランスフォーメーションに関するボトルネックを減らすには、以下の方法を検討します。

最適なデータベースドライバを使用する。

データ統合サービスは、ネイティブデータベースドライバまたは ODBC ドライバを使用してルックアップテーブルに接続できます。ネイティブデータベースドライバは、ODBC ドライバよりも高いパフォーマンスを得ることができます。

リレーショナルまたはフラットファイルのルックアップ用にルックアップテーブルをキャッシュする。

リレーショナルソースまたはフラットファイルソースのルックアップパフォーマンスを向上させるには、トランスフォーメーションのルックアップキャッシュを有効にします。キャッシュを有効にするとルックアップテーブルがキャッシュされます。マッピングを実行すると、データ統合サービスは、ルックアップテーブルではなく、ルックアップキャッシュに問い合わせを行います。このオプションを有効にしていない場合、データ統合サービスは行ごとにルックアップテーブルを参照します。

ルックアップテーブルをキャッシュに格納するかどうかに関わらず、ルックアップクエリの結果および処理は同じです。ただし、ルックアップキャッシュを使用するとマッピングのパフォーマンスを向上させることができます。通常、必要なサイズが 300MB 未満のルックアップテーブルは、キャッシュに入れます。

論理データオブジェクトのルックアップ用にルックアップテーブルをキャッシュする。

論理データオブジェクトのルックアップパフォーマンスを向上させるには、データ統合サービスでデータオブジェクトのキャッシュを有効にします。データオブジェクトのキャッシュを有効にすると、論理データオブジェクトがキャッシュされます。データオブジェクトのキャッシュを有効にするには、マッピングをアプリケーションにデプロイして、論理データオブジェクトのキャッシュを有効にし、`infacmd ms runmapping` コマンドでマッピングを実行します。マッピングを実行すると、データ統合サービスは、論理データオブジェクトではなく、データオブジェクトキャッシュに問い合わせを行います。

Developer tool からマッピングを実行すると、ルックアップトランスフォーメーションが論理データオブジェクトに行単位での問い合わせを行います。

適切なキャッシュタイプを使用する。

次のキャッシュのタイプを使用して、パフォーマンスを向上させることができます。

- 共有キャッシュ。複数のトランスフォーメーション間でルックアップキャッシュを共有できます。名前なしキャッシュを同じマッピング内のトランスフォーメーション間で共有することができます。名前付きキャッシュを同じマッピング内、または異なるマッピング内のトランスフォーメーション間で共有することができます。
- 永続キャッシュ。キャッシュファイルを保存して再使用するには、永続キャッシュを使用できるようにトランスフォーメーションを設定します。この機能は、マッピングの実行間でルックアップテーブルが変更されていないことが明らかな場合に使用します。データ統合サービスはメモリキャッシュをデータベースではなくキャッシュファイルから作成するので、永続キャッシュを使用するとパフォーマンスが向上します。

コンカレントキャッシュを有効にする。

ルックアップトランスフォーメーションを含むマッピングを処理する場合、データ統合サービスはキャッシュを使用するルックアップトランスフォーメーションでデータの最初の行を処理するときに、メモリにキャッシュを構築します。マッピングに複数のルックアップトランスフォーメーションがある場合、ルックアップトランスフォーメーションによってデータの最初の行が処理されるたびに、データ統合サービスは順番にキャッシュを作成します。これにより、Lookup トランスフォーメーションの処理は遅くなります。

コンカレントキャッシュを有効にすると、パフォーマンスを向上させることができます。並行処理する追加パイプライン数が 1 つまたは複数に設定されると、データ統合サービスは順番ではなく同時にキャッシュを作成します。アグリゲータ、ジョイナ、またはソータトランスフォーメーションなど、処理するのに時間のかかるアクティブなトランスフォーメーションがセッションに多数存在している場合、この方法でパフォーマンスを大幅に向上させることができます。複数のコンカレントパイプラインを有効にすると、データ統合サービスはキャッシュを構築するまでアクティブなマッピングを待たなくなります。パイプライン内の他のルックアップトランスフォーメーションも、同時にキャッシュを作成します。

ルックアップ条件の一致を最適化する。

Lookup トランスフォーメーションでルックアップキャッシュデータと Lookup 条件が一致すると、最初に一致した値と最後に一致した値を判断するため、データをソートし並べ替えます。Lookup 条件に一致する値を何か返すように、トランスフォーメーションを設定できます。Lookup トランスフォーメーションが一致する値を何か返すよう設定すると、トランスフォーメーションは Lookup 条件に一致する最初の値を返します。一致する先頭の値を返すか、または最後の値を返すようにトランスフォーメーションを設定する場合と異なり、必ずしもすべてのポートのインデックスを行いません。

一致する任意の値を使用すると、パフォーマンス低下の原因でもある、トランスフォーメーションによるすべてのポートのインデックスを行わないため、パフォーマンスを改善することができます。

キャッシュに入れるレコード数を減らす。

キャッシュに入れる行数を減らして、パフォーマンスを向上させることができます。ルックアップ SQL 上書きというオプションを使用すると、デフォルトの SQL 文に WHERE 句を追加できます。動的キャッシュを使用するルックアップトランスフォーメーションに WHERE 句を追加するときは、ルックアップトランスフォーメーションの前にフィルタトランスフォーメーションを使用して、WHERE 句に一致する動的キャッシュに行を渡します。

ORDER BY 文を上書きする。

デフォルトでは、データ統合サービスはキャッシュされたルックアップに対して ORDER BY 文を生成します。ORDER BY 文にはすべての Lookup ポートが含まれます。パフォーマンスを向上させるには、デフォルトの ORDER BY 文を使用しないで、カラム数の少ない ORDER BY を上書きで入力します。

上書きで ORDER BY 文を入力しても、データ統合サービスは常に ORDER BY 文を生成します。ORDER BY 上書きのあとに 2 個のダッシュ (--) を挿入して、生成した ORDER BY 文を抑止します。

たとえば、Lookup トランスフォーメーションが次の Lookup 条件を使用するとします。

```
ITEM_ID = IN_ITEM_ID  
PRICE <= IN_PRICE
```

Lookup トランスフォーメーションには、マッピングで使用される 3 つの Lookup ポート (ITEM_ID、ITEM_NAME、および PRICE) が含まれます。ORDER BY 文を入力した場合、Lookup 条件にポートを入力したのと同じ順番でカラムを入力します。また、すべてのデータベース予約語は引用符で囲みます。

Lookup SQL 上書きで次の Lookup クエリを入力します。

```
SELECT ITEMS_DIM.ITEM_NAME, ITEMS_DIM.PRICE, ITEMS_DIM.ITEM_ID FROM ITEMS_DIM ORDER BY  
ITEMS_DIM.ITEM_ID, ITEMS_DIM.PRICE --
```

メモリの多いマシンを使用する。

マッピングのパフォーマンスを向上させるには、大容量のメモリを搭載したデータ統合サービスノード上でマッピングを実行します。マシンに負荷をかけない範囲で、インデックスキャッシュサイズおよびデータキャッシュサイズをできる限り増やします。データ統合サービスノードに十分なメモリがある場合は、ディスクにページングしなくてもすべてのデータをメモリ内に保存できるよう、キャッシュを増やします。

ルックアップ条件を最適化する。

複数のルックアップ条件を入れる場合は、ルックアップのパフォーマンスを最適化するために、次の順序で条件を指定します。

- 等しい (=)
- より小さい (<)、より大きい (>)、より小さいまたは等しい (<=)、より大きいまたは等しい (>=)
- 等しくない (!=)

ルックアップ行をフィルタする。

パフォーマンスを向上させるには、フィルタ条件を作成して、ルックアップキャッシュの作成時にソースから取得されるルックアップ行の数を少なくします。

ルックアップテーブルをインデックスに入れる。

データ統合サービスは、ルックアップ条件カラムの値に対してクエリ、ソート、および比較を行う必要があります。インデックスには、Lookup 条件で使用する各カラムを含めなければなりません。

次の種類の Lookup において、パフォーマンスを向上させることができます。

- キャッシュを使用するルックアップ。パフォーマンスを向上させるには、ルックアップの ORDER BY 文のカラムにインデックスを付けます。マッピングログファイルには ORDER BY 文が含まれます。
- キャッシュを使用しないルックアップ。パフォーマンスを向上させるには、ルックアップ条件のカラムにインデックスを付けます。データ統合サービスは、ルックアップトランスフォーメーションに入る各行に対して SELECT 文を発行します。

複数のルックアップを最適化する。

マッピングに複数のルックアップが含まれている場合、キャッシングが有効になっていてヒープメモリが十分にあっても、ルックアップによりパフォーマンスが低下することがあります。一番多くのデータを問い合わせるルックアップトランスフォーメーションを探して、全体のパフォーマンスを向上させることができます。

ルックアップテーブルがマッピングにおいてソーステーブルと同じデータベースにあり、キャッシュが使用できない場合は、ルックアップトランスフォーメーションを使用する代わりに、ソースデータベース内でテーブル同士を結合します。

ソータトランスフォーメーションの最適化

データ統合サービスノードの物理 RAM にデータをソートするのに十分なメモリが割り当てられていない場合は、ソータトランスフォーメーションによってパフォーマンスが低下することがあります。

ソータトランスフォーメーションに関するボトルネックを減らすには、以下の方法を検討します。

十分なメモリを割り当てる。

最適なパフォーマンスを実現するには、データ統合サービスノードで利用可能な物理 RAM 量以下の値をソータキャッシュサイズに設定します。ソータトランスフォーメーションを使用してデータをソートするには、最低 16MB の物理メモリを割り当てます。デフォルトでは、ソータキャッシュサイズは 16,777,216 バイトに設定されます。データ統合サービスがデータをソートするのに十分なメモリを割り当てられない場合、マッピングは失敗します。

入力されるデータの量がソータキャッシュサイズよりも大きい場合、データ統合サービスはデータを一時的にソータトランスフォーメーションのワークディレクトリに保存します。データを作業ディレクトリに保存する場合、データ統合サービスは最低でも入力されるデータの量の 2 倍のディスク領域を必要とします。

SQL トランスフォーメーションの最適化

データ統合サービスは、マッピング内で新しいクエリを処理するたびに、SQLPrepare という関数を呼び出し、SQL プロシージャを作成してデータベースに渡します。入力行ごとにクエリが変更されると、パフォーマンスに影響する場合があります。

SQL トランスフォーメーションに関するボトルネックを減らすには、以下の方法を検討します。

SQL トランスフォーメーションクエリではトランザクション文を使用しない。

SQL クエリにコミットおよびロールバックのクエリ文が含まれる場合、データ統合サービスでは、コミットまたはロールバックするたびに SQL プロシージャを再作成する必要があります。パフォーマンスを最適化するには、SQL トランスフォーメーションクエリでトランザクション文を使用しないでください。

SQL トランスフォーメーションで初期選択または最適化にプッシュインのフィルタ最適化方式を有効にする。

パフォーマンスを向上させるには、SQL トランスフォーメーションでのプッシュイン最適化方式または初期選択の有効化を検討します。

SQL トランスフォーメーションを使用した初期選択の最適化

フィルタ条件がパススルーポートのみを参照し、SQL トランスフォーメーションに副次作用がない場合、Data Integration Service は SQL トランスフォーメーションを使用して初期選択の最適化を実行することができます。

SQL トランスフォーメーションには、次の状況で副次作用があります。

- SQL クエリがデータベースを更新する。SQL クエリには、CREATE、DROP、INSERT、UPDATE、GRANT、REVOKE などの文が含まれます。
- SQL トランスフォーメーションが結果を返さない SELECT 文に対して NULL 行を返す。行には、パススルーポート値、SQL エラー情報、または NUMRowsAffected フィールドが含まれます。

SQL トランスフォーメーションによる初期選択の最適化の有効化

SQL トランスフォーメーションに副次作用がない場合は、SQL トランスフォーメーションで初期選択の最適化を有効にします。

1. SQL トランスフォーメーションの **【詳細プロパティ】** で、**【データベース出力のみ返す】** オプションを有効にします。
2. SQL トランスフォーメーションの **【詳細プロパティ】** で、**【副次作用あり】** をクリアします。
3. SQL トランスフォーメーションに **NumAffectedRows** ポートがある場合は、このポートを削除します。

SQL トランスフォーメーションによるプッシュイン最適化

最適化にプッシュインを有効にすると、SQL トランスフォーメーションにおけるクエリへのマッピングで、Data Integration Service はフィルタトランスフォーメーションからフィルタロジックをプッシュします。

SQL トランスフォーメーションで最適化にプッシュインを有効にする場合は、以下のルールおよびガイドラインに従います。

- トランスフォーメーション SQL クエリは SELECT 文を含む必要があります。
- トランスフォーメーション SQL クエリは有効なサブクエリである必要があります。
- フィルタ条件は、[SQL エラー] フィールドまたは [NumRowsAffected] フィールドを参照できません。
- 出力ポートの名前は、SQL SELECT 文内のカラムの名前と一致している必要があります。フィルタ条件内で出力ポートを参照すると、Data Integration Service は対応するポート名を SQL クエリにプッシュします。クエリ内のカラムが出力ポート名と一致しない場合は、SQL にエイリアスを追加できます。次に例を示します。SELECT mycolname1 AS portname1, mycolname2 AS portname2
- トランスフォーメーションには副次作用がありません。

SQL トランスフォーメーションによるプッシュイン最適化の例

SQL トランスフォーメーションは顧客 ID 別に注文を取得します。SQL トランスフォーメーションの後に出現するフィルタトランスフォーメーションは、注文数が 1000 を超えた行のみを返します。

Data Integration Service は、次のフィルタを SQL トランスフォーメーションの SELECT 文にプッシュします。

```
orderAmount > 1000
```

SQL クエリ内の各文は、フィルタが含まれる SELECT 文の個々のサブクエリになります。

次のクエリ文は、元のクエリ文を SELECT 文内のサブクエリとして示しています。

```
SELECT <customerID>, <orderAmount>, ... FROM (original query statements) ALIAS WHERE <orderAmount> > 1000
```

SQL クエリに複数の文がある場合、各文は個々のサブクエリに含まれます。サブクエリの構文は、WHERE 句など、同じになります。

ポート *customerID* および *orderAmount* は、SQL トランスフォーメーションの出力ポートの名前です。サブクエリには、パススルーポート、SQL エラー、または SQL 統計ポートは含まれません。複数のフィルタを SQL トランスフォーメーションにプッシュする場合、WHERE 句にすべてのフィルタが含まれます。

SQL トランスフォーメーションによるプッシュイン最適化の有効化

最適化にプッシュインを有効にするには、SQL トランスフォーメーションの【詳細プロパティ】タブでプロパティを設定します。

1. 【副次作用あり】をクリアします。
2. 【データベース出力のみ返す】を有効にします。
3. 【最大出力行数】をゼロに設定します。
4. 【最適化にプッシュイン】を有効にします。

トランスフォーメーションのキャッシュ

アグリゲータ、ジョイナ、ルックアップ、ランク、またはソータトランスフォーメーションを使用するマッピングを実行すると、データ統合サービスはメモリにキャッシュを作成して、トランスフォーメーションを処理します。さらに多くの領域が必要になると、ディスク上のオーバーフローした値がキャッシュファイルに格納されます。

トランスフォーメーションのキャッシュに関するボトルネックを減らすには、以下の方法を検討します。

キャッシュをメモリに格納できるようにトランスフォーメーションの設定で十分な容量を割り当てる。

アグリゲータ、ジョイナ、ルックアップ、ランク、またはソータトランスフォーメーションの処理時間を向上させるには、キャッシュをメモリに格納するための十分な容量をトランスフォーメーションの設定で割り当てます。キャッシュメモリの量をデータとインデックスをキャッシュするのに必要な量以上に設定すると、システム I/O のオーバーヘッドが減り、パフォーマンスが向上します。データ統合サービスがキャッシュファイルをディスクに書き込むと、システム I/O のオーバーヘッドにより処理時間が長くなります。

デフォルトでは、キャッシュメモリ要件は実行時に自動的に設定されます。自動キャッシュモードでマッピングを実行した後、トランスフォーメーションのキャッシュサイズを調整できます。マッピングログでトランスフォーメーションの統計データを分析して、トランスフォーメーションをメモリで処理するのに必要なキャッシュサイズを決定します。マッピングログに指定された値を使用するようにキャッシュサイ

ズを設定することで、割り当てたメモリを無駄にしないようにすることができます。ただし、最適なキャッシュサイズは、ソースデータのサイズに応じて異なります。以降のマッピングの実行後にマッピングログを確認して、キャッシュサイズへの変更を監視します。再利用可能なトランスフォーメーションに対して特定のキャッシュサイズを設定した場合、マッピングでそのトランスフォーメーションを使用するたびに、キャッシュサイズが最適であるかどうか確認する必要があります。

トランスフォーメーションエラーの除去

トランスフォーメーションエラーが多数発生すると、Data Integration Service のパフォーマンスが低下します。それぞれのトランスフォーメーションエラーに対して、Data Integration Service は、一時停止してそのエラーを調べ、エラーの原因となっている行をデータフローから削除します。通常は、Data Integration Service のログのマッピングログファイルにその行が書き込まれます。

トランスフォーメーションエラーに関するボトルネックを減らすには、以下の方法を検討します。

トランスフォーメーションエラーが発生した箇所をマッピングログファイルで調べ、トランスフォーメーション制約を評価する。

Data Integration Service で変換エラー、競合しているマッピングロジック、およびエラーとして設定されている条件（Null 入力など）が検出されると、トランスフォーメーションエラーが発生します。トランスフォーメーションエラーが発生した箇所を調べるには、マッピングログファイルをチェックします。エラーが特定のトランスフォーメーションで発生している場合は、そのトランスフォーメーション制約を評価してください。

トレースレベルを低くする。

大量のトランスフォーメーションエラーを発生させるマッピングを実行する必要がある場合は、トレースレベルを低く設定することによってパフォーマンスを向上させることができます。ただし、この方法はトランスフォーメーションエラーに対する長期的な対策としてはお勧めできません。

トランスフォーメーションの副次作用

トランスフォーメーションが行を返し、オブジェクトを変更する場合、またはほかのオブジェクトや関数とやりとりする場合は、副次作用があります。例えば、データベースの変更、合計の加算、例外の生成、電子メールの作成、副次作用がある他の関数の呼び出しなどが行われることがあります。

マッピングを最適化する前に副次作用が発生するトランスフォーメーションはデータ統合サービスで特定されます。データ統合サービスは、トランスフォーメーションに副次作用があるかを判断できない場合に、トランスフォーメーションに副次作用があると想定します。

トランスフォーメーションに副次作用がある場合、データ統合サービスによるマッピングの最適化が制限されます。データ統合サービスが副次作用のあるトランスフォーメーションに適用される場合にマッピングの結果が変わるものとして、初期選択、ブランチ刈り込み、グローバル述部最適化、およびプッシュイン最適化があります。初期選択およびプッシュイン最適化では、フィルタトランスフォーメーションのフィルタロジックがソースのできるだけ近くに移動されます。副次作用の関数の前にフィルタが実行されると、マッピングの結果が変わります。

例えば、顧客 ID を受け取り、注文情報を含む行を返すトランスフォーメーションがあるとします。このトランスフォーメーションでは、さらに、ファイルに注文を書き込みます。ファイルに注文を書き込む前にデータ統合サービスはフィルタの最適化を適用すると、後でマッピングでフィルタを実行する場合よりも、ファイル

で受け取る行数が少なくなります。このトランスフォーメーションの場合、ファイルに注文レコードを書き込む関数が副次作用になります。

副次作用があるトランスフォーメーションは次のとおりです。

- SQL トランスフォーメーション、Web サービスコンシューマトランスフォーメーション、Java トランスフォーメーション（副次作用のプロパティを無効にしていない場合）。
- ABORT()関数または ERROR()関数の呼び出し、メールの送信、またはストアドプロシージャの呼び出しを実行するトランスフォーメーション。
- ファイルやデータベースへの書き込みを行うトランスフォーメーション。
- 変数ポートを使用してカウントを維持するトランスフォーメーションたとえば、COUNT=COUNT+1 です。

SQL トランスフォーメーション、Web サービスコンシューマトランスフォーメーション、および Java トランスフォーメーションには、デフォルトで副次作用があります。副次作用なしで行を処理するようにトランスフォーメーションを設定するには、**【詳細プロパティ】**で**【副次作用あり】**プロパティを無効にします。トランスフォーメーションに副次作用がない場合は、それらのトランスフォーメーションで追加プロパティを設定して最適化を有効にできます。

Web サービスコンシューマトランスフォーメーションの最適化

マッピングが Web サービスを複数回にわたって呼び出す場合、Web サービスコンシューマトランスフォーメーションによってパフォーマンスが低下することがあります。

Web サービスコンシューマトランスフォーメーションに関するボトルネックを減らすには、以下の方法を検討します。

クッキー認証を使用するよう Web サービスコンシューマトランスフォーメーションを設定する。

リモート Web サービスサーバーは、クッキーに基づいて、Web サービスコンシューマユーザーを追跡します。マッピングで Web サービスの呼び出しを繰り返し行う場合に、パフォーマンスが向上します。

クッキーポートを Web サービス要求メッセージに対して投影する場合、Web サービスプロバイダは応答メッセージでクッキー値を返します。クッキー値は、マッピング内の他のトランスフォーメーションダウンストリームに渡すことも、ファイル内に保存することもできます。クッキー値をファイルに保存する場合、そのクッキー値を Web サービスコンシューマトランスフォーメーションに対する入力として設定できます。クッキー出力ポートは、Web サービスコンシューマトランスフォーメーションの任意の出力グループに投影できます。

Web サービスコンシューマトランスフォーメーションで初期選択または最適化にプッシュインのフィルタ最適化方式を有効にする。

パフォーマンスを向上させるために、データ統合サービスは Web サービスコンシューマトランスフォーメーションでのプッシュイン最適化方式または初期選択を適用できます。初期選択の最適化を適用するには、Web サービスに副次作用があってはならず、フォルトをエラーとして扱ってもなりません。最適化にプッシュインを適用するには、Web サービスに副次作用があってはならず、フォルトをエラーとして扱ってもならず、またフィルタ条件がパススルーポートを参照する必要があります。

Web サービスコンシューマトランスフォーメーションに応答を返す以外に、その他の関数を実行する場合、その Web サービスには副次作用があります。Web サービスの副次作用には、データベースの変更、ファイルへの書き込み、電子メールの作成、カウントの更新、副次作用がある他の Web サービスの呼び出しなどがあります。

複数の要求を並行して送信するように Web サービスコンシューマトランスフォーメーションを設定します。

マッピングのパフォーマンスを向上させるには、複数の要求を並行して送信するように、Web サービスコンシューマトランスフォーメーションを設定します。Web サービスコンシューマトランスフォーメーションを有効にして、Web サービスに対して複数の同時接続を作成するときには、メモリ消費量の制限と同時接続制限を設定することができます。

Web サービスコンシューマトランスフォーメーションによる初期選択の最適化

Data Integration Service によって Web サービスコンシューマトランスフォーメーションで初期選択最適化方式が適用されると、フィルタ条件がマッピング内の Web サービスコンシューマトランスフォーメーションの前、ソース側に移動されます。

Web サービスコンシューマトランスフォーメーションでの初期選択の最適化の有効化

トランスフォーメーションに副次作用がなく、フォルトをエラーとして扱うこともない場合は、Web サービスコンシューマトランスフォーメーションで初期選択の最適化を有効にします。

1. Web サービスコンシューマトランスフォーメーションの【詳細プロパティ】ビューを開きます。
2. 【フォルトをエラーとして扱う】をクリアします。
3. 【副次作用あり】をクリアします。

Web サービスコンシューマトランスフォーメーションによるプッシュイン最適化

トランスフォーメーションが SQL データサービスの仮想テーブルにある場合は、プッシュイン最適化を Web サービスコンシューマトランスフォーメーションで設定できます。

データオブジェクトマッピングは、Web サービスを呼び出して、エンドユーザーの SQL クエリの文に基づいてデータのセットまたはデータのサブセットを取得します。エンドユーザーの SQL クエリには、オプションのフィルタ条件が含まれます。

Web サービスコンシューマトランスフォーメーションは、最適化にプッシュインによって、フィルタポートのフィルタ値を受け取ります。フィルタポートは、最適化にプッシュインを設定するときにフィルタポートとして特定する接続されていない入力ポートです。フィルタポートには、エンドユーザーのクエリにフィルタが含まれていない場合に Web サービスがすべての行を返すようにするデフォルト値があります。フィルタポートはパススルーポートではありません。

注: フィルタフィールドは、Web サービス要求のルートグループの一部である必要があります。

フィルタポートの設定時に、Web サービスコンシューマトランスフォーメーションで Web サービス応答からカラムデータを受け取る出力ポートを特定します。例えば、フィルタポートが EmployeeID という名前の入力ポートである場合、応答の出力ポートのポート名は EmployeeNum になります。Developer ツールは、読み込まれた仮想テーブルから Web サービスコンシューマ要求にフィルタロジックをプッシュするために、入力フィルタポートと出力ポートを関連付ける必要があります。Web サービス要求の入力ポートは、通常、Web サービス応答の出力ポートとは異なります。

フィルタフィールドをパススルーポートにすることはできません。フィルタポートの設定時、ポートのデフォルト値がフィルタ条件の値に変わるため、パススルー出力ポートの値が変わります。出力パススルーポートに基づくフィルタは、予期しない結果を返します。

Web サービスコンシューマトランスフォーメーションには複数の式をプッシュすることができます。各フィルタ条件は次の形式である必要があります。

<Field> = <Constant>

フィルタ条件は AND で結合する必要があります。フィルタ条件を OR で結合することはできません。

Web サービスコンシューマトランスフォーメーションによるプッシュイン最適化の例

SQL データサービスは、すべての顧客の注文を返すか、ユーザーから受け取る SQL クエリに基づいて特定の顧客の注文を返します。

SQL データサービスには、次のコンポーネントから構成される論理データオブジェクトが含まれています。

顧客テーブル

顧客情報が含まれる Oracle データベーステーブル。

Web サービスコンシューマトランスフォーメーション

Web サービスを呼び出して顧客の最新の注文を取得するトランスフォーメーション。Web サービスコンシューマトランスフォーメーションには、customerID と orderNum の入力ポートがあります。このトランスフォーメーションには、顧客テーブルから受け取る顧客データが含まれるパススルーポートがあります。orderNum ポートはフィルタポートで、接続されていません。OrderNum には、デフォルト値 "*" があります。Web サービスが Web サービス要求でこの値を受け取ると、すべての注文を返します。

注文仮想テーブル

Web サービスから顧客データと注文データを受け取る仮想テーブル。エンドユーザーはこのテーブルを参照します。注文には、顧客カラム、orderID カラム、および顧客データと注文データが含まれます。

エンドユーザーは次の SQL クエリを SQL データサービスに渡します。

```
SELECT * from OrdersID where customer = 23 and orderID = 56
```

クエリはマッピングを最適化するために分割されます。初期選択の最適化が使用され、フィルタロジック、customer = 23 が読み込まれた顧客テーブルに移動します。最適化にプッシュインが使用され、フィルタロジック、orderID = 56 が Web サービスコンシューマトランスフォーメーションのフィルタポートにプッシュされます。Web サービスコンシューマトランスフォーメーションでは、顧客 23 の ordersID 56 が取得されます。

Web サービスコンシューマトランスフォーメーションによるプッシュイン最適化の有効化

トランスフォーメーションに副次作用がなく、フォルトをエラーとして扱うこともない場合は、Web サービスコンシューマトランスフォーメーションで最適化にプッシュインを有効にします。

1. Web サービスコンシューマトランスフォーメーションの【詳細プロパティ】ビューを開きます。
2. 【フォルトをエラーとして扱う】をクリアします。
3. 【副次作用あり】をクリアします。
4. 【最適化にプッシュイン】プロパティの【開く】ボタンをクリックします。
5. 【最適化された入力】ダイアログボックスで、フィルタポート名を選択します。
複数のフィルタポートを選択できます。
6. 【出力】カラムをクリックします。
7. フィルタポートごとに、Web サービス応答にフィルタリングされたカラムを含める出力ポートを選択します。

8. フィルタポートのデフォルト値を入力します。

注: Web サービスコンシューマポートは、フィルタポートではない限り、デフォルト値を設定できません。

第 5 章

マッピングの最適化

この章では、以下の項目について説明します。

- [マッピングの最適化の概要, 41 ページ](#)
- [最適化方式, 42 ページ](#)
- [プッシュダウンの最適化, 48 ページ](#)
- [Single-Pass 読み込み, 51 ページ](#)
- [フィルタの最適化, 51 ページ](#)
- [データ型変換の最適化, 52 ページ](#)
- [エラートレース, 52 ページ](#)

マッピングの最適化の概要

データ統合サービスによるデータの変換や移動を効率よく行えるように、マッピングを最適化します。マッピングレベルの最適化はその実装に時間がかかりますが、マッピングのパフォーマンスを大幅に向上させることができます。

最適化タスクは、標準のマッピング、論理データオブジェクト読み取りマッピングおよび書き込みマッピング、仮想テーブルマッピング、操作のマッピングに適用されます。マッピングレベルの最適化は、ターゲットおよびソースの最適化が完了した後にのみ実施できます。

マッピングを最適化するには、以下のタスクを実行できます。

- 処理量を最大化するには、トランスフォーメーションと式をできるだけ少なくしてマッピングを設定します。
- トランスフォーメーション間の不要なリンクを削除して、移動するデータの量を最小限に抑えます。
- 最適化レベルを選択します。この選択により、データ統合サービスがマッピングに適用できる最適化方式が決まります。データ統合サービスによるマッピングの最適化では、処理するデータの量を減らすように試行されます。例えば、データ統合サービスでは、初期選択最適化を使用してフィルタをソースに近づけることができます。コストベースの最適化方式を使用して結合の処理順序を変更できます。
- トランスフォーメーションロジックの一部または全部をソースデータベースにプッシュダウンできるかどうかをデータ統合サービスで判断できるようにするには、プッシュダウンタイプを選択します。
- データ統合サービスキャッシュ論理データオブジェクトを有効にし、マッピングを実行するときに事前作成された論理データオブジェクトにアクセスできるように、データオブジェクトのキャッシュを設定します。デフォルトでは、データ統合サービスがマッピングを実行するときに、ソースデータを抽出し、必要なデータオブジェクトを構築します。データ統合サービスが事前作成されたデータオブジェクトにアクセスできるようにしておくと、マッピングのパフォーマンスが向上します。

- SQL トランスフォーメーション、Web サービスコンシューマトランスフォーメーション、および Java トランスフォーメーションで副次作用がない場合は、それらのトランスフォーメーションを設定するときにそのことを示します。トランスフォーメーションの中には、副次作用によって最適化が制限されるものがあります。トランスフォーメーションの副次作用は、例えば、ファイルやデータベースへの書き込み、カウントの加算、例外の生成、電子メールの作成などをトランスフォーメーションで行う場合に発生する可能性があります。ほとんどの場合、副次作用によって最適化が制限されるトランスフォーメーションはデータ統合サービスで特定されます。

関連項目：

- [「データオブジェクトのキャッシュ」 \(ページ 65\)](#)

最適化方式

各種の最適化方式を適用することによって、マッピングで処理される行数が削減されます。マッピングの最適化レベルを設定して、適用される最適化方式を制限できます。

データ統合サービスでは、以下の最適化方式を適用できます。

- プッシュダウンの最適化
- 初期プロジェクション最適化
- 初期選択最適化
- ブランチ刈り込み最適化
- 最適化にプッシュイン
- 述部最適化
- グローバル述部最適化
- コストベース最適化
- データシップ結合最適化
- 準結合最適化

データ統合サービスでは、マッピングに複数の最適化方式を同時に適用できます。例えば、ノーマル最適化レベルを選択すると、データ統合サービスは、初期プロジェクション最適化、述部最適化、グローバル述部最適化、ブランチ刈り込み最適化、および初期選択最適化またはプッシュイン最適化方式を適用します。

最適化レベル

設定した最適化レベルに基づいて、データ統合サービスによりマッピングが最適化されます。マッピングでデフォルト以外の最適化レベルを使用する場合は、最適化レベルのプロパティを設定します。

以下の最適化レベルのいずれかを選択できます。

自動

データ統合サービスは、実行モードとマッピングコンテンツに基づいて最適化を適用します。

なし

データ統合サービスは最適化は適用されません。

最小

データ統合サービスは初期プロジェクション最適化方式を適用します。

ノーマル

データ統合サービスは、初期プロジェクション、初期選択、ブランチ刈り込み、プッシュイン、グローバル述部、述部の最適化方式を適用します。

完全

データ統合サービスは、コストベース、初期プロジェクション、初期選択、ブランチ刈り込み、述部、プッシュイン、準結合、データシップ結合の最適化方式を適用します。

デフォルトは「自動」です。

マッピングを Developer tool の【実行】メニューまたはマッピングエディタから実行した場合、ノーマルの最適化レベルが適用されます。マッピングを【実行】メニューから実行した場合は、マッピング設定に従って最適化レベルが適用されます。マッピングをコマンドラインから実行した場合は、アプリケーションで設定したマッピングのデプロイメントのプロパティに従って最適化レベルが適用されます。

注: データ統合サービスはプッシュダウンの最適化方式に最適化レベルを適用しません。マッピングランタイムプロパティでマッピングのプッシュダウンの最適化を設定できます。

フィルタの最適化

フィルタを最適化すると、マッピングを通過する行数が減り、パフォーマンスが向上します。Data Integration Service は、初期選択の最適化と最適化にプッシュインを適用することができます。

Data Integration Service でフィルタ最適化方式が適用されると、マッピング内でフィルタができるだけソースの近くに移動されます。Data Integration Service がマッピング内のトランスフォーメーションの前にフィルタを移動できない場合、フィルタロジックをトランスフォーメーションにプッシュできる可能性があります。

初期プロジェクション最適化方法

データ統合サービスで初期プロジェクション最適化方式を適用すると、未使用のポートが特定され、それらのポート間のリンクが削除されます。

初期プロジェクション最適化方式では、トランスフォーメーション間を移動するデータの量を少なくすることによってパフォーマンスを向上させます。データ統合サービスは、マッピングを処理するときに、マッピング内のすべての接続ポートのデータをトランスフォーメーション間で移動します。大きく複雑なマッピング、またはネストされたマプレットを使用するマッピングには、データをターゲットに提供しないポートがある場合があります。データ統合サービスにより、データをターゲットに提供しないポートが特定されます。未使用のポートが特定されたら、すべての未使用のポート間のリンクがマッピングから削除されます。

すべてのリンクが削除されるわけではありません。例えば、以下のリンクは削除されません。

- 副次効果のあるトランスフォーメーションに接続されているリンク。
- ABORT()関数または ERROR()関数の呼び出し、メールの送信、またはストアードプロシージャの呼び出しを実行するトランスフォーメーションに接続されているリンク。

トランスフォーメーション内のすべてのポートが未使用であると判断されたら、データが最も少ないポートへのリンクを除くすべてのトランスフォーメーションリンクが削除されます。未使用のトランスフォーメーションはマッピングから削除されません。

この最適化方式は、Developer ツールではデフォルトで有効になっています。

述部最適化方式

データ統合サービスで述部最適化方式を適用すると、マッピングで生成された述部式が調べられ、マッピングのパフォーマンス向上のために式を簡略化するか書き直すことができるかどうかが判断されます。

データ統合サービスは、マッピングを実行するときに、マッピングソースに対するクエリを生成し、マッピングログロジックおよびマッピング内のトランスフォーメーションに基づいてクエリ結果に対して操作を実行します。クエリおよび操作には、多くの場合、述部式が含まれます。述部式は、データが満たす必要がある条件を表します。述部式の例としては、フィルタトランスフォーメーションとジョイナトランスフォーメーションのフィルタ条件と結合条件があります。

また、述部最適化方式では、マッピングのパフォーマンス向上のために、マッピングで述部式をできる限り早期に適用するように試行されます。

データ統合サービスでは、既存の述部式からリレーションを推測し、新しい述部式を作成します。例えば、マッピングに結合条件「A=B」を持つジョイナトランスフォーメーションおよびフィルタ条件「A>5」を持つフィルタトランスフォーメーションが含まれているとします。データ統合サービスは、「B>5」を結合条件に追加できます。

データ統合サービスでは、述部最適化方式と初期選択最適化方式の両方をマッピングに適用できる場合、それらの方式が適用されます。例えば、データ統合サービスで述部最適化方式を使用して新しいフィルタ条件が作成されるとき、初期選択方式を使用してマッピングのアップストリームへの条件の移動も試行されます。両方の最適化方式を適用すると、一方の方式のみを適用したときと比べてマッピングのパフォーマンスが向上します。

データ統合サービスで述部最適化方式が適用されるのは、それによってパフォーマンスが向上する場合です。適用することでマッピングの結果が変わったりマッピングのパフォーマンスが低下したりする場合は、この方式は適用されません。データ統合サービスでは、デフォルトでこの最適化方式が適用されます。

述部最適化に関するルールとガイドライン

データ統合サービスによって述部式が書き直される場合、最適化のために式に算術ロジックが適用されます。

データ統合サービスは以下のアクションのいずれかまたはすべてを実行することがあります。

- マッピングの述部式間で同等の変数が特定され、同値に基づいて簡略化された式が生成されます。
- マッピングの述部式間で冗長な述部が特定され、削除されます。
- 隣接する句からサブ式が抽出され、サブ式に基づいて複数の簡略化された式が生成されます。
- 述部式が正規化されます。
- マッピングで述部式ができる限り早期に適用されます。

接続ポート間でデータ型が一致しないトランスフォーメーションがマッピングに含まれている場合は、データ統合サービスはマッピングに述部最適化を適用しない可能性があります。

以下の条件のいずれかに該当する場合、データ統合サービスはトランスフォーメーションに述部最適化が適用されない可能性があります。

- トランスフォーメーションに接続ポートの明示的なデフォルト値が含まれている。
- トランスフォーメーションには副次作用があります。
- トランスフォーメーションで述部の移動が許可されていない。たとえば、副次作用があるトランスフォーメーションにはこの制限がある可能性があります。

述部最適化方式は、Developer ツールではデフォルトで有効になっています。

コストベースの最適化方式

コストベースの最適化では、データ統合サービスによって、マッピングが評価されて意味的に同等のマッピングが生成され、最適なパフォーマンスでマッピングが実行されます。コストベースの最適化では、隣接する内部結合や完全な外部結合の操作を実行するマッピングの実行時間が短縮されます。

意味的に同等のマッピングとは、同じ関数を実行して同じ結果になるマッピングです。意味的に同等のマッピングを生成するために、データ統合サービスでは、元のマッピングがフラグメントに分割されます。次に、最適化できるマッピングのフラグメントが特定されます。

最適化中に、フラグメント内のトランスフォーメーションの追加、削除、または順序変更が行われる場合があります。最適化したフラグメントが元のフラグメントと同じ結果になることが検証され、最適化したフラグメントを使用する代替マッピングが形成されます。

データ統合サービスは、ソートされたマージ結合のパフォーマンスが、ネストされたループ結合のパフォーマンスよりも高いと判断した場合に、ソートされたマージ結合を適用することもできます。ソートされたマージ結合では、結合を実行する前に、ソート順を使用して2つのデータセットを整列します。ネストされたループ結合では、ネストされたループを使用して2つのデータセットを結合します。データをソートするためのコストが、ネストされたループ結合を処理するためのコストより低い場合、データ統合サービスは、ソースのソート情報を使用したり、ソータートランスフォーメーションを作成したりできます。

元のマッピングと意味的に同等のすべてのマッピングまたはほぼすべてのマッピングが生成されます。プロファイリング統計またはデータベース統計を使用して、元のマッピングおよび各代替マッピングのコストを計算します。次に、最も早く実行するマッピングを特定します。最適な代替マッピングに対して検証チェックが実行され、そのマッピングが有効で元のマッピングと同じ結果になることが確認されます。

最適な代替マッピングがメモリにキャッシュされます。マッピングを実行するときに、代替マッピングが取得されて元のマッピングの代わりに実行されます。

この方式は、Developer tool ではデフォルトで無効になっています。

データシップ結合最適化方式

データシップ結合最適化方式では、大きいデータセットに隣接する小さいデータセットを特定し、結合処理の時間を短縮しようとします。データ統合サービスは、2つのテーブルのサイズが大幅に異なる場合、データシップ結合最適化方式を適用しようとします。

例えば、データ統合サービスは、データシップ結合最適化方式を適用して、10,000 行のマスタテーブルと 1,000,000 行の詳細テーブルを結合できます。データ統合サービスは、データシップ結合を実行するために、大きい詳細テーブルを含むデータベースに一時ステージングテーブルを作成します。次に、データ統合サービスは、小さいマスタテーブルを一時テーブルにコピーし、一時テーブルのデータと大きい詳細テーブルのデータを結合します。データ統合サービスが結合操作を実行すると、データベースでジョイナートランスフォーメーションロジックが処理されます。

データシップ結合最適化方式を適用する前に、データ統合サービスは分析を実行し、データシップ結合最適化が可能か、また実行する価値があると考えられるかを判断します。分析によって、この方式でパフォーマンスが向上する可能性が高いと判断されたら、この方式がマッピングに適用されます。その後、マッピングが再分析されて、データシップ結合最適化を行う機会が他にもあるかどうか判断されます。必要に応じて、最適化がさらに実行されます。

この方式は、Developer ツールではデフォルトで無効になっています。

データシップ結合でパフォーマンスを向上させるための要件

データシップ結合最適化方式では、パフォーマンスが常に向上するとは限りません。データシップ結合最適化によるマッピングのパフォーマンスに影響する要因を次に示します。

- ジョイナトランスフォーメーションのマスタソースの行が明細ソースよりも大幅に少なくなければならぬ。
- 明細ソースが最適化の正当性を保証できるほど大幅に大きくなければならぬ。明細ソースが十分に大きくない場合、データ統合サービスは、データシップ結合最適化方式を適用せずに、マスタソースおよび明細ソースからすべてのデータを読み取る方が速いと判断します。

データシップ結合最適化に関するルールとガイドライン

データ統合サービスでは、次の要件を満たすジョイナトランスフォーメーションにデータシップ結合最適化を適用できます。

- 結合タイプがノーマル、マスタ外部、または明細外部である。
- 明細パイプラインがリレーショナルソースから生成されている。
- ジョイナトランスフォーメーション範囲が「すべての入力」である（マッピングでターゲットベースのコミットが使用される場合）。
- マスタパイプラインと明細パイプラインでトランスフォーメーションが共有されていない。
- 明細ソースとジョイナトランスフォーメーションの間のブランチがマッピングに含まれていない。
- 結合の詳細サイドを含むデータベースが Unicode エンコーディングをサポートしない IBM DB2 データベースの場合、データ統合サービスはデータシップ結合最適化方式の適用に失敗します。

準結合最適化方式

準結合最適化方式では、マッピングで結合操作を変更することで、ソースから抽出されるデータの量の削減が試行されます。

データ統合サービスでは、一方の入力グループに他方よりも多くの行が含まれている場合、結合条件に基づいて、小さい方のグループに一致するものがない行が大きい方のグループに多数含まれているときに、準結合最適化方式がジョイナトランスフォーメーションに適用されます。データ統合サービスは、小さい方のグループから行を読み取り、大きい方のグループで一致する行を見つけて結合操作を実行することで、1つの結合オペランドのデータセットのサイズを小さくしようとします。データセットのサイズを小さくすると、データ統合サービスで大きい方のグループソースから不要な行が読み取られなくなるため、マッピングのパフォーマンスが向上します。データ統合サービスによって、結合条件が大きい方のグループソースに移動され、小さい方のグループと一致する行のみが読み取られます。

準結合最適化方式を適用する前に、データ統合サービスは分析を実行し、準結合最適化が可能か、また実行する価値があると考えられるかを判断します。分析によって、この方式でパフォーマンスが向上する可能性が高いと判断されたら、この方式がマッピングに適用されます。その後、マッピングが再分析されて、準結合最適化を行う機会が他にもあるかどうか判断されます。必要に応じて、最適化がさらに実行されます。

この方式は、Developer ツールではデフォルトで無効になっています。

準結合最適化でパフォーマンスを向上させるための要件

準結合最適化方式では、パフォーマンスが常に向上するとは限りません。準結合最適化によるマッピングのパフォーマンスに影響する要因を次に示します。

- ジョイナトランスフォーメーションのマスタソースの行が明細ソースよりも大幅に少なくなければならぬ。

- 明細ソースが最適化の正当性を保証できるだけの大きさでなければならない。準結合最適化方式を適用すると、マッピング処理にオーバーヘッド時間が加わります。明細ソースが小さい場合は、準結合最適化の適用にかかる時間が、明細ソース内のすべての行の処理にかかる時間を超える可能性があります。
- Data Integration Service で、通常の結合操作と準結合操作にかかる時間を正確に比較できるように、ジョイナトランスフォーメーションのソース行数の統計を取得できなければならない。

準結合最適化に関するルールとガイドライン

Data Integration Service では、以下の要件を満たすジョイナトランスフォーメーションに準結合最適化を適用できます。

- 結合タイプがノーマル、マスタ外部、または明細外部である。ジョイナトランスフォーメーションでは、完全外部結合は実行できません。
- 明細パイプラインがリレーショナルソースから生成されている。
- 結合条件が有効なソート-マージ-結合条件である。つまり、各句が1つのマスタポートと1つの明細ポートの等式である必要があります。複数の句がある場合は、AND で結合する必要があります。
- ジョイナトランスフォーメーション範囲が [すべての入力] である（マッピングでターゲットベースのコミットが使用されない場合）。
- マスタパイプラインと明細パイプラインでトランスフォーメーションが共有されていない。
- 明細ソースとジョイナトランスフォーメーションの間のブランチがマッピングに含まれていない。

初期選択最適化方式

Data Integration Service で初期選択最適化方式を適用すると、マッピングでフィルタトランスフォーメーションの分割、移動、削除が実行されます。フィルタはマッピングのソースに近づくように移動されます。

フィルタ条件が論理積の場合、フィルタトランスフォーメーションは分割されることがあります。例えば、フィルタ条件「A>100 AND B<50」は、2つのより単純な条件「A>100」および「B<50」に分割されることがあります。フィルタを分割する場合、Data Integration Service は、簡略化されたフィルタをマッピングパイプラインの上に移動してソースに近づけます。分割されたフィルタはパイプラインで個別に移動されます。

初期選択最適化方式は、最適化レベルをノーマルまたは完全に設定した場合に Developer tool でデフォルトで有効になります。フィルタトランスフォーメーションより前に出てくるトランスフォーメーションに副次作用がある場合、データ統合サービスは初期選択最適化を無視します。データ統合サービスは、SQL トランスフォーメーション、Web サービスコンシューマトランスフォーメーション、Java トランスフォーメーションに副次作用があるかを判断できません。これらのトランスフォーメーションに副次作用がない場合は、これらのトランスフォーメーションの初期選択最適化を設定できます。

初期選択最適化でパフォーマンスが向上しない場合は無効にしてください。データ統合サービスでは、デフォルトでこの最適化方式が有効化されます。

グローバル述部最適化方式

データ統合サービスでグローバル述部最適化方式を使用すると、マッピング内で除外可能な行ができるだけ初期に削除されます。これにより、マッピングで処理する必要がある行数が減少します。グローバル述部最適化方式には、述部最適化方式と初期選択方式が含まれます。

例えば、マッピングに結合条件「A=B」を持つジョイナトランスフォーメーションおよびフィルタ条件「A>5」を持つフィルタトランスフォーメーションが含まれているとします。データ統合サービスは、「B>5」を結合条件に追加し、フィルタトランスフォーメーションをソースに近づけることができる可能性があります。

グローバル述部最適化方式では、述部式が述部最適化方式よりも効果的に適用されます。グローバル述部最適化方式では、式の簡素化または書き直してマッピングのパフォーマンスを向上できるかどうか判断されます。

また、マッピングパフォーマンスを向上するための述部式をマッピング内にできるだけ初期に適用するための試みが行われます。

グローバル述部最適化方式では、マッピングに、ネストされたジョイナやブランチが含まれており、各ブランチにフィルタが存在する場合、フィルタが推測されソースの近くにプッシュされます。データ統合サービスでグローバル述部最適化方式を使用すると、データ統合サービスはフィルタを分割し、フィルタをソースに近づけるか、マッピング内でフィルタを削除します。

ブランチ刈り込み最適化方式

データ統合サービスでは、マッピングにおいてターゲットに行をまったく渡さないトランスフォーメーションにブランチ刈り込み最適化方式を適用できます。

データ行に関してフィルタ条件が FALSE に評価されると、データ統合サービスはフィルタトランスフォーメーションを削除することがあります。例えば、マッピングに、2つのリレーショナルソースのデータをフィルタリングする2つのフィルタトランスフォーメーションが存在するとします。一方のフィルタトランスフォーメーションには「Country=US」というフィルタ条件が設定されており、もう一方のフィルタトランスフォーメーションには「Country=Canada」というフィルタ条件が設定されているとします。共有体トランスフォーメーションは、これら2つのリレーショナルソースを結合するもので、「Country=US」というフィルタ条件が設定されているとします。データ統合サービスは、フィルタ条件「Country=Canada」が設定されたフィルタトランスフォーメーションをマッピングから削除することがあります。

ブランチ刈り込み最適化方式は、最適化レベルをノーマルまたは完全に設定した場合に Developer ツールでフォルトで有効になります。この最適化でパフォーマンスが向上しない場合は、最適化レベルを最小またはなしに設定してブランチ刈り込みを無効にすることができます。

プッシュイン最適化方法

最適化にプッシュインでは、データ統合サービスはフィルタトランスフォーメーションロジックを、マッピングのフィルタトランスフォーメーションのすぐ上にあるトランスフォーメーションに移動します。プッシュイン最適化を適用すると、マッピングを通過する行数が減り、パフォーマンスが向上します。

データ統合サービスによる別のトランスフォーメーションへのフィルタロジックの移動は、トランスフォーメーションに副次作用がある場合は行われません。データ統合サービスは、SQL トランスフォーメーション、Web サービスコンシューマトランスフォーメーション、Java トランスフォーメーションに副作用があるかを判断できません。ただし、プッシュイン最適化に対して、SQL トランスフォーメーション、Web サービスコンシューマトランスフォーメーション、および Java トランスフォーメーションを設定できます。

プッシュダウンの最適化

データ統合サービスでプッシュダウンの最適化を適用すると、トランスフォーメーションロジックがソースにプッシュされます。データ統合サービスはトランスフォーメーションロジックを SQL クエリに変換し、その SQL クエリをデータベースに送信します。トランスフォーメーションを処理する SQL クエリはソースデータベースで実行されます。

プッシュダウンの最適化は、ソースデータベースがデータ統合サービスよりも高速にトランスフォーメーションロジックを処理できる場合、マッピングのパフォーマンスを向上します。データ統合サービスがソースから読み取るデータも少なくなります。

データ統合サービスがソースデータベースにプッシュするトランスフォーメーションロジックの量は、データベース、トランスフォーメーションロジック、およびマッピング設定によって決まります。データ統合サービスは、データベースにプッシュできないすべてのトランスフォーメーションロジックを処理します。

プッシュダウンの最適化を適用すると、ソースからターゲットまで、またはソースデータベースにプッシュできないダウンストリームトランスフォーメーションまで、最適化されたマッピングが分析されます。データ統合サービスは、トランスフォーメーションロジックをプッシュダウンしたソースごとに、SELECT クエリを生成して実行します。ターゲットがデータベースにプッシュされた場合、データ統合サービスは INSERT クエリを生成することもできます。データ統合サービスはこの SQL クエリの結果を読み取り、マッピングの残りのトランスフォーメーションを処理します。

マッピングのランタイムプロパティでプッシュダウンタイプを選択した場合、データ統合サービスはプッシュダウンの最適化をマッピングに適用します。

以下のプッシュダウンタイプを選択することができます。

- なし。マッピングのプッシュダウンタイプを選択しません。
- ソース。データ統合サービスは、ソースデータベースに、できるだけ多くのトランスフォーメーションロジックのプッシュダウンを試みます。
- 全体。データ統合サービスは、トランスフォーメーションロジック全体をソースデータベースにプッシュします。

プッシュダウンタイプの文字列パラメータを作成し、以下のパラメータ値を使用することもできます。

- なし
- ソース
- 全体

完全なプッシュダウンの最適化

データ統合サービスは、完全なプッシュダウンの最適化を適用するときに、マッピングのすべてのトランスフォーメーションロジックをソースデータベースにプッシュします。マッピングランタイムプロパティで完全なプッシュダウンを設定できます。

完全なプッシュダウンの最適化は、ソースとターゲットが同じデータベースにある場合や、アグリゲータトランスフォーメーションやフィルタトランスフォーメーションなどのトランスフォーメーションをソースデータベースで処理してデータの移動量を減らす場合に最適です。例えば、マッピングに Teradata ソースと Teradata ターゲットが含まれている場合、完全なプッシュダウンの最適化を設定して、処理するすべてのトランスフォーメーションロジックを Teradata ソースデータベースから Teradata ターゲットデータベースにプッシュします。

完全なプッシュダウン用にアップデートストラテジトランスフォーメーションを含むマッピングを設定する場合、マッピングのプッシュダウン互換性を判断する必要があります。

次の場合、データ統合サービスはアップデートストラテジトランスフォーメーションを含むマッピングをプッシュダウンできます。

- アップデートストラテジトランスフォーメーションに接続されているターゲットトランスフォーメーションが、異なるキーを持つ複数の行を受信する場合。
- アップデートストラテジトランスフォーメーションに接続されているターゲットトランスフォーメーションが、並べ替えできる同じキーを持つ複数の行を受信する場合。

次の場合、データ統合サービスはアップデートストラテジトランスフォーメーションを含むマッピングをプッシュダウンできません。

- アップデートストラテジトランスフォーメーションに接続されているターゲットトランスフォーメーションが、並べ替えできない同じキーを持つ複数の行を受信する場合。

マッピングでプッシュダウン互換性パラメータを使用することもできます。次のパラメータ値を使用できます。

- noMultipleRowsWithSameKeyOnTarget

- `reorderAllowedForMultipleRowsWithSameKey`
- `reorderNotAllowedForRowsWithSameKey`

データ統合サービスで完全なプッシュダウンの最適化を使用できるソースは次のとおりです。

- Amazon Redshift
- Greenplum
- IBM DB2
- Microsoft SQL Server
- Netezza
- Oracle
- SAP HANA
- Snowflake
- Teradata

ソースプッシュダウン

データ統合サービスは、ソースプッシュダウンを適用するときにソースからターゲットへのマッピングを分析します。これを行わないと、ダウンストリームトランスフォーメーションに達するまで、ソースデータベースにプッシュできません。

データ統合サービスは、データベースにプッシュできる各トランスフォーメーションのトランスフォーメーションロジックに基づいて、SELECT 文を生成、実行します。次に、統合サービスではこの SQL クエリの結果が読み込まれ、残りのトランスフォーメーションが処理されます。

ソースとターゲットが異なるデータベースに存在している場合、ソースプッシュダウンを使用するようにマッピングを設定できます。例えば、マッピングに Teradata ソースと Oracle ターゲットが含まれている場合、ソースプッシュダウンを設定して、処理する一部のトランスフォーメーションロジックを Teradata ソースにプッシュできます。

プッシュダウンの最適化に関するルールとガイドライン

データ統合サービスは、以下のトランスフォーメーションロジックをソースデータベースにプッシュできます。

プッシュダウンの最適化には、次のルールとガイドラインが適用されます。

- データ統合サービスがルックアップおよびジョイナのトランスフォーメーションロジックをソースデータベースにプッシュできるのは、ソースが同じデータベース管理システムにあり、同じ接続を使用する場合です。
- データ統合サービスは Binary データ型のソースにトランスフォーメーションロジックをプッシュすることはできません。
- IBM DB2 データソースがあり、10 進データ型のカラム精度が 28～31 桁である場合、データ統合サービスはプッシュダウンの最適化を無効にします。
- データ統合サービスは、SQL データサービスまたは Web サービスに対してデフォルトでプッシュダウンの最適化を有効にします。SQL データサービスまたは Web サービスに対するプッシュダウンの最適化を無効にすることはできません。
- データ統合サービスは、group by ではないポートの集計関数または非集計関数を持つ式を含むアグリゲータトランスフォーメーションをプッシュできません。

Single-Pass 読み込み

Single-Pass 読み込みでは、1つのカスタマイズデータオブジェクトを使用して複数のターゲットに値を入れることができます。同じソースを使用するマッピングが複数ある場合は、Single-Pass 読み込みの使用を検討します。

Single-Pass 読み込みに関するボトルネックを減らすには、以下の方法を検討します。

各マッピングのトランスフォーメーションロジックを1つのマッピングに結合し、各ソースで1つのカスタマイズデータオブジェクトを使用する。

Data Integration Service は、各ソースを一度読み込んでからデータを異なるパイプラインに送信します。状況に応じて、パイプラインの組み合わせや、パイプライン以外の組み合わせにより、すべてのパイプラインで特定の行を使用することもできます。

たとえば、Purchasing ソーステーブルがあり、そのソースを毎日使用して集計およびランク付けを行うとします。アグリゲータトランスフォーメーションとランクトランスフォーメーションを別個のマッピングに分けた場合、Data Integration Service に同じソーステーブルを2回読み込ませることになります。ただし、1つのソース修飾子を有する1つのマッピングに集計およびランキングロジックを含めると、Data Integration Service では、一度 Purchasing ソーステーブルを読み込んでから、異なるパイプラインに適切なデータを送信します。

マッピングの共通の処理を減らす。

Single-Pass 読み込みを利用するようにマッピングを変更する場合は、マッピングの共通の処理を減らすことによって、この機能を最適化します。例えば、アグリゲータとランクの両方のトランスフォーメーションにおいて Price ポートからパーセンテージを減算する必要がある場合は、パイプラインを分割する前にパーセンテージの減算処理を行うことによって処理を最小限に抑えることができます。式トランスフォーメーションを使用してパーセンテージを減算してから、トランスフォーメーション実行後にマッピングを分割します。

フィルタの最適化

カスタマイズデータオブジェクト内でフィルタリングを行うか、マッピングの初期段階でのフィルタの使用によって、マッピングを最適化することができます。

フィルタに関するボトルネックを減らすには、以下の方法を検討します。

カスタマイズデータオブジェクトのフィルタを使用してソースの行を削除します。

マッピングの行にフィルタを適用する場合、データフローの初期段階でフィルタリングを行うことによって効率を上げることができます。カスタマイズデータオブジェクトのフィルタを使用してソースの行を削除します。カスタマイズデータオブジェクトはリレーショナルソースから抽出した行セットを制限します。

カスタムデータオブジェクトのフィルタを使用できない場合は、フィルタトランスフォーメーションを使用し、できるだけカスタムデータオブジェクトに近づけて、データフローの初期段階で不要なデータを削除します。フィルタトランスフォーメーションは、ターゲットに送信される行セットを制限します。

拒否された行を保持する必要がある場合はアップデートストラテジトランスフォーメーションのフィルタを使用する。

拒否された行を保持する必要がある場合は、マッピングのパフォーマンスを向上させるために、フィルタトランスフォーメーションを使用して、拒否された行をアップデートストラテジトランスフォーメーションから削除することもできます。

フィルタ条件で複雑な式を使用しないようにする。

フィルタ条件では、複雑な式を使用しないようにします。Filter トランスフォーメーションを最適化するには、フィルタ条件で簡単な整数式または真偽式を使用します。

フィルタトランスフォーメーションはマッピング内のデータをフィルタリングします。フィルタトランスフォーメーションは、すべての種類のソースからの行をフィルタリングします。カスタマイズデータオブジェクトはリレーショナルソースの行をフィルタリングします。フィルタトランスフォーメーションは、すべての種類のソースからの行をフィルタリングします。

データ型変換の最適化

不要なデータ型変換を除去することにより、パフォーマンスが向上します。たとえば、Integer カラムから Decimal カラムへデータを移動した後に再び Integer カラムにそのデータを戻すようなマッピングの場合、不必要なデータ型を変換することによってパフォーマンスが低下します。可能な限り、不必要なデータ型の変換をマッピングから取り除きます。

データ型変換に関するボトルネックを減らすには、以下の方法を検討します。

ルックアップトランスフォーメーションとフィルタトランスフォーメーションを使用して比較を実行する際は、Integer 以外のデータ型についても Integer 値を使用する。

例えば、数多くのデータベースで、米国の郵便番号情報は Char データ型または Varchar データ型として格納されます。郵便番号のデータを Integer データ型に変換した場合、ルックアップデータベースでは郵便番号の 94303-1234 を 943031234 という値で格納します。これで、郵便番号に基づいたルックアップの比較条件の処理が高速化します。

マッピングのパフォーマンスを向上させるためにポート対ポート変換を介してソースの日付を文字列に変換します。

ターゲットのポートを文字列のままにすることも、ポートを Date/Time ポートに変更することもできます。

エラートレース

パフォーマンスを向上させるには、Data Integration Service でのマッピングの実行時に生成されるログイベントの数を少なくします。マッピング設定またはマッピングのデプロイメントのプロパティを介してマッピングの最適化レベルを更新することにより、マッピングのパフォーマンスを向上させることができます。マッピングを最適化するには、コストベースの最適化方式を使用します。

エラートレースに関するボトルネックを減らすには、以下の方法を検討します。

マッピングのプロパティでトレースレベルを「簡易」に設定する。

マッピング内に大量のトランスフォーメーションエラーがあっても、修正する必要がない場合は、マッピングのプロパティでトレースレベルを「簡易」に設定します。このトレースレベルの場合、Data Integration Service はエラーメッセージや拒否データの行レベル情報を書き込みません。

マッピングをデバッグする必要がある、トレースレベルを「詳細」に設定すると、マッピングを実行した際にパフォーマンスの著しい低下が発生する可能性があります。パフォーマンスのチューニングを行っているときは、「詳細」を使用しないでください。マッピングのトレースレベルは、マッピング内のトランスフォーメーション固有のトレースレベルをオーバーライドします。ただし、この方法は多くのトランスフォーメーションエラーが発生する場合の長期的な対策としてはお勧めできません。

マッピングの最適化レベルを変更する。

マッピングの実行所要時間が非常に長い場合は、マッピングの最適化レベルを変更することをお勧めします。最適化レベルによって、Data Integration Service が実行時にマッピングに適用する最適化方式が決まります。

マッピングの最適化レベルは、マッピング設定またはマッピングのデプロイメントのプロパティで設定します。Data Integration Service では、マッピングの実行方法に応じて異なる最適化レベルがマッピングに適用されます。

コストベースの最適化方式を使用する。

コストベースの最適化方式では、Data Integration Service によって、マッピングが評価されて意味的に同等のマッピングが生成され、最適なパフォーマンスでマッピングが実行されます。この方式は、複数のジョイントランスフォーメーションが含まれているマッピングの場合に最も有効です。隣接する未ソートのインナージョイン操作を実行するマッピングの実行時間が短縮されます。

意味的に同等のマッピングとは、同じ関数を実行して同じ結果になるマッピングです。意味的に同等のマッピングを生成するために、Data Integration Service では、元のマッピングがフラグメントに分割されます。次に、最適化できるマッピングのフラグメントが特定されます。

第 6 章

パーティション化したマッピングの最適化

この章では、以下の項目について説明します。

- [パーティション化したマッピングの最適化の概要, 54 ページ](#)
- [複数の CPU の使用, 55 ページ](#)
- [最大並行処理の値の増加, 55 ページ](#)
- [パーティション化に対応するためのフラットファイルの最適化, 56 ページ](#)
- [パーティション化に対応するためのリレーショナルデータベースの最適化, 57 ページ](#)
- [パーティション化に対応するためのトランスフォーメーションの最適化, 58 ページ](#)

パーティション化したマッピングの最適化の概要

パーティション化が可能な場合は、データ統合サービスによるマッピング実行時の並行処理を最大化することができます。並行処理を最大化すると、データ統合サービスによって基になるデータが動的にパーティションに分割され、すべてのパーティションが同時に処理されます。

マッピングが大規模なデータセットを処理する場合、または複雑な計算を実行するトランスフォーメーションを含む場合は、マッピングの処理に時間がかかり、データのスループットが低下する可能性があります。これらのマッピングに対してパーティション化を有効にすると、データ統合サービスは追加のスレッドを使用してマッピングを処理します。

パーティション化したマッピングのパフォーマンスを最適化するには、以下のタスクを実行できます。

- マッピングを実行するノードで複数の CPU を使用します。
- データ統合サービスの最大並行処理の値を増加します。
- フラットファイルデータオブジェクトに関するプロパティを設定します。
- パーティション化が最適になるようにリレーショナルデータベースを設定します。
- トランスフォーメーションに関するプロパティを設定します。

複数の CPU の使用

処理スレッドの数が増加すると、マッピングを実行するノードの負荷が増加します。ノードに十分な CPU 帯域幅がある場合、マッピングのデータ行を同時処理することで、マッピングのパフォーマンスが最適化されます。

データ統合サービスは、複数の CPU を使用して、複数のパーティションを含むマッピングを処理することができます。サービスで使用する CPU の数は、パーティションポイントの数、各パイプラインステージに対して作成されたスレッドの数、マッピングの処理に必要なリソースの容量など、いくつかの要素で決まります。単純なマッピングは、2つのパーティションでより高速に実行されますが、通常、1つのパーティションで実行されるマッピングに比べて、CPU を 2 倍多く必要とします。

最大並行処理の値の増加

最大並行処理は、単一のパイプラインステージを処理できる並列スレッドの最大数を決定します。利用可能なハードウェアリソースに基づいて、データ統合サービスの **【最大並行処理】** プロパティを設定します。最大並行処理の値を増やすと、処理時間を減らすことができます。

最大並行処理の値を設定する場合は、次のガイドラインを考慮してください。

使用可能な CPU の数に応じて値を増やします。

マッピングを実行するノードで使用可能な CPU の数に基づいて最大並行処理値を増やします。最大並列処理値を増やすと、データ統合サービスは、マッピングの実行にさらに多くのスレッドを使用し、さらに多くの CPU を活用します。単純なマッピングを 2つのパーティションで実行すると時間を短縮できますが、通常、1つのパーティションで実行されるマッピングに比べて、CPU を 2 倍多く必要とします。

処理スレッドの総数に留意してください。

並行処理の最大値を設定する際、処理スレッドの総数に留意してください。複雑なマッピングによって、追加のパーティションポイントが複数生じた場合、データ統合サービスによって、CPU が処理できる以上の処理スレッドが使用されることがあります。

処理スレッドの合計数は最大並列処理値に等しくなります。

データ統合サービスで実行する必要がある他のジョブに留意してください。

各マッピングで大量のスレッドを使用するように最大並行処理値を設定すると、データ統合サービスが追加のジョブを実行する場合に使用できるスレッドが少なくなります。

必要に応じて、マッピングの値を変更します。

デフォルトでは、各マッピングの最大並行処理が自動に設定されています。各マッピングは、データ統合サービスに定義されている最大並行処理値を使用します。

開発者は、Developer tool でマッピングランタイムプロパティ内の最大並行処理値を変更することにより、特定のマッピングの最大値を定義できます。データ統合サービスとマッピングで設定されている最大並行処理の整数値が異なる場合、データ統合サービスではこれらの最小値が使用されます。

パーティション化に対応するためのフラットファイルの最適化

パーティション化を有効にしたマッピングによって、フラットファイルソースからの読み取り、またはフラットファイルターゲットへの書き込みが行われる場合、データ統合サービスは複数のスレッドを使用して、フラットファイルに対する読み取りまたは書き込みを行うことができます。

パーティション化に対応するためのフラットファイルソースの最適化

複数のスレッドを使用して1つのフラットファイルを読み取る場合に最適なパフォーマンスを達成するには、行の順番を保持するのではなくスループットを最適化するようにフラットファイルデータオブジェクトを設定します。

パーティション化したフラットファイルソースのボトルネックを軽減するには、以下の解決策を検討します。

スループットを最適化するようにフラットファイルデータオブジェクトの同時読み取りのパーティション化を設定します。

フラットファイルデータオブジェクトの詳細プロパティで、スループットを最適化するように**【同時読み取りのパーティション化】**プロパティを設定します。スループットを最適化すると、データ統合サービスによって行の順番が保持されることはありません。ファイルやファイルリストの行を順番に読み取ることがないためです。

パーティション化に対応するためのフラットファイルターゲットの最適化

複数のスレッドを使用してフラットファイルに書き込む場合に最適なパフォーマンスを達成するには、ターゲット出力を個別のファイルに書き込むようにパーティション化を設定し、複数のターゲットディレクトリを設定します。

パーティション化したフラットファイルターゲットのボトルネックを軽減するには、以下の解決策を検討します。

ターゲット出力を個別のファイルに書き込むようにパーティション化を設定します。

フラットファイルデータオブジェクトの詳細プロパティで、**【マージタイプ】**プロパティを**【マージなし】**に設定します。データ統合サービスは、ターゲット出力を各パーティションの個別のファイルに同時に書き込みます。マージしたターゲットデータが必要な場合、同時マージタイプを使用すると、シーケンシャルマージタイプを使用するよりもパフォーマンスが最適化されます。

複数のターゲットディレクトリを設定します。

複数のスレッドが単一のディレクトリに書き込む場合、入出力 (I/O) 競合によりマッピングにボトルネックが発生することがあります。複数のスレッドがデータをファイルシステムに同時に書き込む場合、I/O 競合が発生する可能性があります。複数のディレクトリを設定すると、データ統合サービスは各スレッドの出力ディレクトリをラウンドロビン方式で決定します。

フラットファイルデータオブジェクトの詳細プロパティで、出力ファイルディレクトリを設定します。Administrator ツールでデータ統合サービスの**【ターゲットディレクトリ】**プロパティ用に管理者がセミコロンで区切られた複数のディレクトリを入力した場合は、デフォルトの TargetDir システムパラメータ値を使用します。また、別の値を入力し、フラットファイルデータオブジェクトに固有の出力ファイルディレクトリを複数設定することもできます。

パーティション化に対応するためのリレーショナルデータベースの最適化

パーティション化を有効にしたマッピングによって、IBM DB2 for LUW または Oracle リレーショナルデータベースに対する読み取りまたは書き込みが行われる場合、データ統合サービスは複数のスレッドを使用して、リレーショナルソースを読み取ったり、リレーショナルターゲットに書き込んだりできます。

複数のスレッドを使用して DB2 for LUW または Oracle リレーショナルデータベースの読み書きを行う場合にパフォーマンスを最適化する場合、ソースおよびターゲットのテーブルをパーティション化できます。

注: マッピングによって DB2 for LUW または Oracle 以外のリレーショナルデータベースに対する読み取りまたは書き込みが行われる場合、データ統合サービスは 1 つの読み取りスレッドまたは 1 つの書き込みスレッドを使用します。

パーティション化に対応するためのソースデータベースの最適化

複数のスレッドを使用して DB2 for LUW または Oracle ソースデータベースを読み取る場合に最適なパフォーマンスを達成するには、ソーステーブルがパーティション化されており、並列クエリを許可するようにソーステーブルが設定されていることを確認します。

パーティション化に対応するためにソースデータベースを最適化するには、次のタスクを実行します。

データベースパーティションをソースに追加する。

ソースを読み取るデータ統合サービスの速度を増すために、リレーショナルソースにデータベースパーティションを追加します。ソースにデータベースパーティションがない場合、データ統合サービスは 1 つのスレッドを使用してソースから読み取ります。

並列クエリを有効にする。

リレーショナルデータベースには、データベースへの並列クエリを有効にするオプションが用意されていることがあります。これらのオプションについては、データベースのマニュアルを参照してください。これらのオプションが有効化されていない場合、データ統合サービスは複数のパーティションの SELECT 文を連続して実行します。

データを異なるテーブルスペースに分割する。

各データベースには、データを複数のテーブルスペースに分割するオプションが用意されています。各テーブルスペースは、一意のファイルシステムを参照できるため、パーティション間の I/O 競合が回避されます。

データベースで許可されている最大セッション数を入力する。

データ統合サービスは、各パーティション用にソースデータベースへの個別の接続を作成します。データベースで大量の同時接続を処理できるように、許可するセッションの最大数を増やします。

パーティション化に対応するためのターゲットデータベースの最適化

複数のスレッドを使用して DB2 for LUW または Oracle ターゲットデータベースに書き込む場合に最適なパフォーマンスを達成するには、ターゲットテーブルがパーティション化されており、行を並列挿入するようにターゲットテーブルが設定されていることを確認します。

パーティション化に対応するためにターゲットデータベースを最適化するには、次のタスクを実行します。

データベースパーティションを DB2 for LUW ターゲットに追加する。

データ統合サービスは複数のスレッドを使用して、データベースパーティションのない DB2 for LUW ターゲットに書き込むことができます。ただし、ターゲットにデータベースパーティションがある場合は、口

ードパフォーマンスを最適化することができます。この場合、各 writer スレッドは、データベースパーティションを含む DB2 for LUW ノードに接続します。すべてのスレッドが単一のマスターノードに接続するのではなく、writer スレッドが別々の DB2 for LUW ノードに接続するため、パフォーマンスは向上します。

並列挿入を有効にする。

リレーショナルデータベースには、データベースへの並列挿入を有効にするオプションが用意されていることがあります。これらのオプションについては、データベースのマニュアルを参照してください。例えば、db_writer_processes オプションを Oracle データベースで設定し、max_agents オプションを DB2 for LUW データベースで設定して、並列挿入を有効にします。

データを異なるテーブルスペースに分割する。

各データベースには、データを複数のテーブルスペースに分割するオプションが用意されています。各テーブルスペースは、一意のファイルシステムを参照できるため、パーティション間の I/O 競合が回避されます。

データベースで許可されている最大セッション数を入力する。

データ統合サービスは、各パーティション用にターゲットデータベースへの個別の接続を作成します。データベースで大量の同時接続を処理できるように、許可するセッションの最大数を増やします。

データベースの拡張性を強化するオプションを設定する。

リレーショナルデータベースには、拡張性を強化するオプションが用意されていることがあります。たとえば、Oracle ではアーカイブログと実行時統計を無効にして、データベースを拡張します。

パーティション化に対応するためのトランスフォーメーションの最適化

データ統合サービスで複数のスレッドを使用してアグリゲータ、ジョイナ、ランク、またはソータのトランスフォーメーションを実行する場合、データ統合サービスはキャッシュのパーティション化を使用して、スレッド全体にキャッシュサイズを分割します。キャッシュのパーティション化に対応するようにパフォーマンスを最適化するには、複数のキャッシュディレクトリを設定します。

注: ルックアップトランスフォーメーションでは、1 つのキャッシュディレクトリしか使用できません。

パーティション化したアグリゲータ、ジョイナ、ランク、およびソータのトランスフォーメーションのボトルネックを軽減するには、以下の解決策を検討します。

複数のキャッシュディレクトリを設定します。

キャッシュのパーティション化では、アグリゲータ、ジョイナ、ランク、またはソータの各トランスフォーメーションを処理するパーティションごとに個別のキャッシュが作成されます。キャッシュのパーティション化中、各パーティションは異なるデータを別々のキャッシュに保存します。各キャッシュには、そのパーティションが必要とする行が含まれます。各スレッドが個々のキャッシュに並行してクエリを実行するため、キャッシュをパーティション化することでマッピングのパフォーマンスが最適化されます。

トランスフォーメーションの実行に必要なメモリ容量よりもキャッシュサイズが小さい場合、トランスフォーメーションスレッドはキャッシュディレクトリに書き込みを行って、オーバーフロー値をキャッシュファイルに保存します。複数のスレッドが単一のディレクトリに書き込む場合、I/O 競合によりマッピングにボトルネックが発生することがあります。複数のスレッドがデータをファイルシステムに同時に書き込む場合、I/O 競合が発生する可能性があります。複数のキャッシュディレクトリを設定すると、データ統合サービスは各トランスフォーメーションスレッドのキャッシュディレクトリをラウンドロビン方式で決定します。

アグリゲータ、ジョイナ、またはランクのトランスフォーメーションの【キャッシュディレクトリ】詳細プロパティでキャッシュディレクトリを設定します。Administrator ツールでデータ統合サービスの【キャッシュディレクトリ】プロパティ用に管理者がセミコロンで区切られた複数のディレクトリを入力した場合は、デフォルトの CacheDir システムパラメータ値を使用します。また、別の値を入力し、トランスフォーメーションに固有のキャッシュディレクトリを複数設定することもできます。

ソータトランスフォーメーションの【ワークディレクトリ】詳細プロパティでキャッシュディレクトリを設定します。Administrator ツールでデータ統合サービスの【一時ディレクトリ】プロパティ用に管理者がセミコロンで区切られた複数のディレクトリを入力した場合は、デフォルトの TempDir システムパラメータ値を使用します。また、別の値を入力し、トランスフォーメーションに固有のキャッシュディレクトリを複数設定することもできます。

第 7 章

実行時の最適化

この章では、以下の項目について説明します。

- [実行時の最適化の概要, 60 ページ](#)
- [アプリケーションサービスの最適化, 60 ページ](#)
- [監視統計, 62 ページ](#)
- [メモリ割り当て, 64 ページ](#)
- [データオブジェクトのキャッシュ, 65 ページ](#)
- [システムの最適化, 68 ページ](#)

実行時の最適化の概要

パフォーマンス機能を有効にし、マッピングのパフォーマンスが最適になるようにデータ統合サービスのプロパティをチューニングします。

要件に基づいて最適なパフォーマンス結果を得るには、Administrator ツールで次の最適化方法を使用します。

- アプリケーションサービスプロセスを最適化する。
- システムのボトルネックを監視するように監視統計を設定する。
- システムパフォーマンスが最適になるようにメモリを割り当てる。
- データオブジェクトのキャッシュの設定
- システムの遅延とディスクアクセス速度の低下を避けるためにシステムを最適化する。

アプリケーションサービスの最適化

パフォーマンスが影響を受けている場合は、アプリケーションサービスプロセスを最適化します。アナリストサービス、データ統合サービス、およびモデルリポジトリサービスを最適化できます。

アナリストサービスの最適化

アナリストサービスをチューニングしてパフォーマンスを最適化します。サービスパフォーマンスを向上させるには、アナリストサービスプロセスのメモリプロパティを設定し、ネットワーク待ち時間を最小限に抑え、Analyst ツールのフラットファイルのアップロードを設定します。

アナリストサービスのボトルネックに対する次の解決策を検討してください。

10MB を超えるフラットファイルをアップロードする場合はネットワークパスの場所に接続するように Analyst ツールを設定する。

Analyst ツールが実行されているマシンの Informatica インストールディレクトリに、10MB を超えるフラットファイルをアップロードすると、アナリストサービスプロセスのパフォーマンスが低下する場合があります。その結果、ディスク容量とネットワークパフォーマンスの両方が影響を受ける可能性があります。

Analyst ツールから Informatica インストールディレクトリに 10MB 未満のフラットファイルをアップロードする。

Analyst ツールから Informatica インストールディレクトリに 10MB を超えるフラットファイルをアップロードすると、アナリストサービスプロセスのパフォーマンスが低下する場合があります。その結果、ディスク容量とネットワークパフォーマンスの両方が影響を受ける可能性があります。

アナリストサービスプロセスの [最大ヒープサイズ] プロパティの値を増やす。

アナリストサービスプロセスは、多数の同時ログインユーザーの処理中に大量のメモリを使用する場合があります。これにより、アナリストサービスと他のサービス（データ統合サービスやモデルリポジトリサービスなど）の間で多数のネットワーク接続が開かれる可能性があります。

Administrator ツールを使用して、アナリストサービスプロセスの [詳細プロパティ] で [最大ヒープサイズ] プロパティの値を大きくしてください。

サイズの大きいマッピング仕様をテーブルにエクスポートするか、フラットファイルにエクスポートしてファイルを切り詰める。

サイズの大きいマッピング仕様を Analyst ツールからフラットファイルとしてエクスポートすると、アナリストサービスプロセスのパフォーマンスに影響する場合があります。

データ統合サービスの最適化

データ統合サービスプロセスをチューニングして、サービスパフォーマンスを向上させます。データ統合サービスプロセスのメモリプロパティを設定できます。また、同時要求を処理するためにデータ統合サービスで実行される各 Web サービスおよび SQL データサービスを設定できます。

データ統合サービスのボトルネックに対する次の解決策を検討してください。

データ統合サービスプロセスの [最大ヒープサイズ] プロパティを設定する。

データ統合サービスは、SQL データサービスおよび Web サービスの処理中に大量のメモリを使用する場合があります。

Administrator ツールを使用して、データ統合サービスプロセスの [詳細プロパティ] で [最大ヒープサイズ] プロパティの値を大きくしてください。

データ統合サービスの Web サービスの [DTM キープアライブ時間] プロパティを設定する。

データ統合サービスは、システムリソースを使用して、Web サービス要求ごとに DTM インスタンスを生成します。1 つの DTM インスタンスを使用して複数の Web サービス要求を処理するようにデータ統合サービスを設定してください。

Administrator ツールを使用して、データ統合サービスの Web サービスの [DTM キープアライブ時間] プロパティを設定してください。

Data Integration プロセスのプロパティと Web サービスおよび SQL データサービスのプロパティで、同時要求の実行オプションを設定する。

データ統合サービスと、データ統合サービスで実行される各 SQL データサービスおよび各 Web サービスは、同時要求ごとにシステムリソースとメモリリソースを使用します。

データ統合サービス、各 SQL データサービス、および各 Web サービスで受け入れることができる同時要求の数を設定するには、データ統合サービスのプロパティと Web サービスのプロパティを設定します。

Administrator ツールを使用して、データ統合サービス、Web サービス、および SQL データサービスの次のオプションとプロパティを設定してください。

- データ統合サービスの実行オプションを設定します。
- データ統合サービスプロセスの SQL プロパティの SQL データサービスごとに [最大同時接続数] プロパティを設定します。
- データ統合サービスプロセスの HTTP 設定プロパティで、Web サービスごとに [最大バックログリクエスト数] プロパティと [最大同時リクエスト数] プロパティを設定します。

Web サービスのトレースレベルをオフにする。

データ統合サービスで書き込まれ維持される Web サービスのログファイルの数によって、パフォーマンスが低下する場合があります。

Administrator ツールを使用して、データ統合サービスによってディスクに格納される Web サービスのランタイムログファイルの数を減らすように、Web サービスのトレースレベルを設定してください。

モデルリポジトリサービスの最適化

モデルリポジトリサービスをチューニングしてパフォーマンスを向上させます。モデルリポジトリサービスプロセスのメモリプロパティを設定し、ネットワーク待ち時間を最小限に抑えることができます。

モデルリポジトリサービスのボトルネックに対する次の解決策を検討してください。

モデルリポジトリサービスと同じマシンでモデルリポジトリデータベースをホストする。

モデルリポジトリデータベースがリモートサーバーでホストされている場合は、モデルリポジトリサービスプロセスのパフォーマンスが影響を受ける可能性があります。待ち時間の長いネットワークにおけるモデルリポジトリとモデルリポジトリサービス間の通信を必要とするモデルリポジトリサービス操作は、モデルリポジトリサービスのパフォーマンスを低下させる可能性があります。

モデルリポジトリサービスプロセスの [最大ヒープサイズ] プロパティの値を増やす。

モデルリポジトリサービスプロセスは、多数の同時ログインユーザーの処理中に大量のメモリを使用する場合があります。これにより、モデルリポジトリサービスと他のサービス（データ統合サービスやアナリストサービスなど）の間で多数のネットワーク接続が開かれる可能性があります。

Administrator ツールを使用して、モデルリポジトリサービスプロセスの [詳細プロパティ] で [最大ヒープサイズ] プロパティの値を大きくしてください。

監視統計

監視は、サービスマネージャで実行されるドメイン機能です。サービスマネージャは、モデルリポジトリに監視設定を格納します。Administrator ツールの [モニタ] タブを使用して、選択したサービスで実行されているジョブ、失敗したジョブ、キャンセルされたジョブ、および完了したジョブの総数などのシステムのボトルネックを監視します。

監視統計のボトルネックに対する次の解決策を検討してください。

監視を設定するようにドメインを設定する。

監視を設定すると、保持されている統計および監視レポートが、データ統合サービスによってモデルリポジトリに格納されます。保持されている統計とは、以前に実行された統合オブジェクトの履歴情報です。監視レポートには、統合オブジェクトに関する重要なメトリックが記載されます。

ドメインの監視設定を行うことで、データ統合サービスにデプロイされたオブジェクトに関するランタイム統計を格納するモデルリポジトリを指定します。監視設定は、ドメイン内のすべてのデータ統合サービスに適用されるため、サービスパフォーマンスに影響する可能性があります。

次の表に、サービスパフォーマンスに影響する可能性がある監視設定を示します。

オプション	説明
サマリ履歴データの保持	モデルリポジトリで平均のデータが保存される日数。パージを無効にしている場合、モデルリポジトリはデータを無期限に保存します。 デフォルトは 180 です。最小値は 0 です。最大値は 366 です。
詳細履歴データの保持	モデルリポジトリで分単位のデータが保存される日数。パージを無効にしている場合、モデルリポジトリはデータを無期限に保存します。 デフォルトは 14 です。最小値は 1 です。最大値は 14 です。
パージ統計の頻度	【履歴データの保持日数】 オプションに設定されている値より古いデータをモデルリポジトリサービスがパージする間隔（日単位）。 デフォルトは 1 日です。
日	モデルリポジトリサービスで統計をパージする時刻。デフォルトは午前 1 時です。
ソート可能な最大レコード数	【モニタ】 タブでソートできるレコードの最大数。 【モニタ】 タブのレコード数がこの値を超えると、ソートの基準にできるのは 【開始時刻】 と 【終了時刻】 のみになります。デフォルトは 3,000 です。
更新通知の最大遅延	データ統合サービスで統計をバッファする最大時間（秒）。この時間を経過すると、モデルリポジトリ内に統計が保存され、 【モニタ】 タブに表示されます。データ統合サービスが、サービスがモデルリポジトリ内の統計を保存する前に予期せずシャットダウンした場合、統計は失われます。デフォルトは 10 です。
ミリ秒の表示	【モニタ】 タブの日付や時刻のフィールドにミリ秒まで表示します。

メモリ割り当て

マッピングパフォーマンスを最適化するには、Administrator ツールでデータ統合サービスのメモリプロパティを設定します。

次の表に、マッピングサービスモジュールに対する要求ごとの最大メモリプロパティを示します。

プロパティ	説明
要求ごとの最大メモリ	<p>要求ごとの最大メモリは、データ統合サービスの次の設定によって異なります。</p> <ul style="list-style-type: none">- 個別のローカルまたはリモートプロセスでジョブが実行されるか、またはサービスプロパティの最大メモリサイズが 0 の場合（デフォルト）。 この場合、要求ごとの最大メモリは、データ統合サービスが、1 回の要求内の、自動キャッシュモードを使用するすべてのトランスフォーメーションに割り当てることができる、最大メモリサイズ（バイト）です。データ統合サービスは、特定のキャッシュサイズの複数のトランスフォーメーションにメモリを別々に割り当てます。要求によって使用されるメモリ n 合計は、要求ごとの最大メモリの値を超えることができます。- データ統合サービスプロセスでジョブが実行され、かつ、サービスプロパティの最大メモリサイズが 0 を超えている場合。 この場合、要求ごとの最大メモリは、データ統合サービスが単一の要求に割り当てることができる、最大メモリサイズ（バイト）です。要求によって使用されるメモリ合計は、要求ごとの最大メモリの値を超えることはできません。 デフォルトは 536,870,912 です。

以下の表に、データ統合サービスの実行オプションを示します。

プロパティ	説明
最大メモリサイズ	<p>データ統合サービスプロセスでジョブを実行する場合、すべての要求を同時に実行するためにデータ統合サービスが割り当てることができる最大メモリサイズ（バイト）。データ統合サービスがジョブを個別のローカルまたはリモートプロセスで実行する場合、この値は無視されません。データ統合サービスで割り当てることができるメモリの量を制限しない場合は、このプロパティを 0 に設定します。</p> <p>値が 0 を超えていた場合、データ統合サービスは、このプロパティを使用して、すべての要求を同時に実行できる最大メモリサイズを計算します。データ統合サービスでは、次の式に基づいて最大合計メモリが計算されます。</p> <p>最大メモリサイズ + 最大ヒープサイズ + プログラムのコンポーネントのロードに必要なメモリ デフォルトは 0 です。</p> <p>注: プロファイルやデータ品質のマッピングを実行する場合は、このプロパティを 0 に設定します。</p>

以下の表に、データ統合サービスプロセスの最大ヒープサイズのプロパティを示します。

プロパティ	説明
最大ヒープサイズ	データ統合サービスを実行する Java Virtual Machine (JVM) に割り当てられる RAM サイズ。このプロパティを使用して、パフォーマンスの向上を図ることができます。単位を指定するには、次のいずれかの文字を値に付加します。 <ul style="list-style-type: none">- b はバイト。- k はキロバイト。- m はメガバイト。- g はギガバイト。 デフォルトは 640 メガバイトです。 注: データ統合サービスで大量のデータを処理する必要がある場合は、最大ヒープサイズを増やすことを検討します。

データオブジェクトのキャッシュ

データ統合サービスでは、データオブジェクトのキャッシュを使用して、事前作成された論理データオブジェクトと仮想テーブルにアクセスします。データオブジェクトのキャッシュを有効にして、論理データオブジェクトと仮想テーブルを含むマッピング、SQL データサービスクエリ、および Web サービスリクエストのパフォーマンスを向上させます。

デフォルトでは、データ統合サービスがマッピング、SQL データサービスクエリ、または Web サービス要求を実行するときに、ソースデータを抽出し、必要なデータオブジェクトを構築します。データオブジェクトのキャッシュを有効にすると、データ統合サービスは、キャッシュされた論理データオブジェクトと仮想テーブルを使用できるようになります。

アプリケーションの論理データオブジェクトおよび仮想テーブルに対してデータオブジェクトのキャッシュを設定するには、次の手順を実行します。

1. データ統合サービスのキャッシュのプロパティで、データオブジェクトキャッシュのデータベース接続を設定します。
2. アプリケーションの論理データオブジェクトまたは仮想テーブルのプロパティでキャッシュを有効にします。

デフォルトでは、データ統合サービスのデータオブジェクトキャッシュマネージャが、データオブジェクトキャッシュデータベース内の論理データオブジェクトと仮想テーブルのキャッシュテーブルを管理します。データオブジェクトキャッシュマネージャがキャッシュを管理する場合、リフレッシュごとに全データがキャッシュテーブルに挿入されます。キャッシュテーブルを差分更新するには、データベースクライアントまたはその他の外部ツールを使って手動でキャッシュテーブル管理します。データオブジェクトのキャッシュを有効にした後に、ユーザー管理のキャッシュテーブルを使用するように論理データオブジェクトまたは仮想テーブルを設定できます。

Timestamp with Time Zone データ型を使用し、IBM DB2 または Microsoft SQL Server に対してデータオブジェクトのキャッシュを有効にするには、デプロイ済みマッピングの日時形式を "YYYY-MM-DD HH24:MI:SS" 形式に設定します。データ統合サービスは最大数秒間でデータを書き込みます。

キャッシュテーブルのデータ型

データ統合サービスでは、キャッシュオブジェクトを含むマッピング、SQL データサービスクエリ、Web サービス要求を処理する場合にキャッシュテーブルのデータを使用します。データ統合サービスに必要なキャッシュテーブルのデータ型は、キャッシュオブジェクトのデータ型と異なることがあります。

データオブジェクトキャッシュマネージャは、データ統合サービスに必要なデータ型でキャッシュテーブルを作成します。ユーザー管理のキャッシュテーブルを使用する場合は、データ統合サービスに必要なデータ型がキャッシュテーブルで使用されていることを確認します。

仮想テーブルのキャッシュのデータ型

以下の表に、仮想テーブルのキャッシュテーブルのデータ型を示します。

仮想テーブルのデータ型	IBM DB2	Microsoft SQL Server	Oracle
Char	Vargraphic Dbclob、長さが 32672 より大きい場合	Nvarchar Ntext、長さが 4000 より大きい場合	Nvarchar2 Nclob、長さが 2000 より大きい場合
Bigint	Bigint	Bigint	Number
Boolean	Integer	Int	Number
Date	Time stamp	Datetime2	Time stamp
ダブル	ダブル	Float	Time stamp
Decimal	Decimal	Decimal	Number
Int	Integer	Int	Number
Time	Time stamp	Datetime2	Time stamp
Time stamp	Time stamp	Datetime2	Time stamp
Varbinary	Blob	Binary Image、長さが 8000 より大きい場合	RAW Blob、長さが 2000 より大きい場合
Varchar	Vargraphic Dbclob、長さが 32672 より大きい場合	Nvarchar Ntext、長さが 4000 より大きい場合	Nvarchar2 Nclob、長さが 2000 より大きい場合

論理データオブジェクトのキャッシュのデータ型

以下の表に、論理データオブジェクトのキャッシュテーブルのデータ型を示します。

論理データオブジェクトデータ型	DB2	Microsoft SQL Server	Oracle
Bigint	Bigint	Bigint	Number
Binary	Blob	Binary Image、長さが 8000 より大きい場合	RAW Blob、長さが 2000 より大きい場合
Date/Time	Time stamp	Datetime2	Time stamp
ダブル	ダブル	Float	Number
Decimal	Decimal	Decimal	Number
Integer	Integer	Int	Number
String	Vargraphic Dbclob、長さが 32672 より大きい場合	Nvarchar Ntext、長さが 4000 より大きい場合	Nvarchar2 Nclob、長さが 2000 より大きい場合
Text	Vargraphic Dbclob、長さが 32672 より大きい場合	Nvarchar Ntext、長さが 4000 より大きい場合	Nvarchar2 Nclob、長さが 2000 より大きい場合

データオブジェクトキャッシュの最適化

キャッシュのパフォーマンスは、キャッシュデータベースのパフォーマンスと、マッピング、SQL データサービス、および Web サービス内のオブジェクト構成によって異なります。

キャッシュのパフォーマンスを向上させるには、以下の方法を検討します。

キャッシュデータベースを最適化する。

キャッシュの最適なパフォーマンスは、キャッシュデータベースの速度とパフォーマンス、およびキャッシュサイズによって異なります。キャッシュデータベース内のキャッシュサイズを設定します。

データオブジェクトキャッシュマネージャでは更新操作のために古いキャッシュを維持する必要があるため、キャッシュには 2 セットのデータを格納できるだけの大きさが必要です。下記の式で必要な最小キャッシュサイズを見積もります。

$$2 * \text{average data object size} * \text{number of data objects}$$

例えば、20 個の論理データオブジェクトと 10 個の仮想テーブルをキャッシュするとします。平均オブジェクトサイズが 15MB の場合、必要なキャッシュサイズは、 $2 * 15\text{MB} * (20 + 10) = 900\text{MB}$ になります。

キャッシュテーブルは読み取り専用です。エンドユーザーは、SQL コマンドでキャッシュテーブルを更新できません。

論理データオブジェクトのプライマリキーと外部キーを定義する。

データ統合サービスでは、キーを含む論理データオブジェクトのキャッシュを生成する際、インデックスを作成します。このインデックスにより、キャッシュデータベースに対するクエリのパフォーマンスが向上します。

マッピングで結合する論理データオブジェクトをキャッシュする。

キャッシュされた論理データオブジェクトを結合すると、ソースデータが別のデータベースのものであっても、データ統合サービスはジョイナトランスフォーメーションロジックをキャッシュデータベースにプッシュダウンできます。

論理データオブジェクトまたは仮想テーブルのカラムに基づいてインデックスキャッシュを生成する。

論理データオブジェクトまたは仮想テーブルのカラムに基づいてインデックスキャッシュを生成するように、データ統合サービスを設定します。このインデックスにより、キャッシュデータベースに対するクエリのパフォーマンスが向上します。

システムの最適化

マッピングが非効率的な接続や過負荷のデータ統合サービスプロセスシステムに依存していると、パフォーマンスの低下が頻繁に発生します。システムの遅延は、ルータ、スイッチ、ネットワークプロトコル、および同時利用者数の過多が原因で生じることもあります。

ソースおよびターゲットのデータベース、ソースおよびターゲットのファイルシステム、およびドメイン内のノードが低速であると、マッピングのパフォーマンスが低下する場合があります。システム管理者に依頼して、使用しているマシンのハードディスクを評価してもらってください。

システムの最適化のボトルネックに対する次の解決策を検討してください。

ネットワーク速度を向上させる。

ネットワーク接続が低速であると、マッピングのパフォーマンスが低下する場合があります。ネットワークの応答性が良好かどうかをシステム管理者に調べてもらってください。データ統合サービスプロセスとデータベースの間のネットワークホップ数を減らします。

複数の CPU を使用する。

複数の CPU を使用すると、複数のマッピングを並列で実行できます。

ページングを削除する。

オペレーティングシステムの物理的なメモリが不足した場合、物理的なメモリを解放するためにディスクへのページングが開始されます。ディスクへのページングが最小限になるように、データ統合サービスプロセスマシン用の物理メモリを構成します。

プロセッサバインドを使用する。

マルチプロセッサ UNIX 環境では、データ統合サービスが大量のシステムリソースを使用する場合があります。プロセッサバインドを使用して、Integration Service プロセスによるプロセッサの使用を制御します。また、ソースデータベースとターゲットデータベースが同じマシン上にある場合は、プロセッサバインドを使って、データベースが使用するリソースを制限します。

第 8 章

SQL データサービスの最適化

この章では、以下の項目について説明します。

- [SQL データサービスの最適化の概要, 69 ページ](#)
- [サードパーティのクライアントツールの最適化, 70 ページ](#)
- [SQL データサービス最適化レベル, 70 ページ](#)
- [SQL データサービスのメモリおよび同時要求のプロパティ, 73 ページ](#)
- [SQL データサービスの結果セットキャッシュ, 75 ページ](#)
- [一時テーブルにおける仮想データの維持, 76 ページ](#)

SQL データサービスの最適化の概要

エンドユーザーがサードパーティのクライアントツールを使用して SQL クエリを実行する際のパフォーマンスが向上するように、SQL データサービスを最適化することができます。SQL データサービスで仮想テーブルマッピングを使用する場合は、ソース、トランスフォーメーション、およびマッピングを最適化できます。

SQL データサービスを最適化する方法は次のとおりです。

- サードパーティのクライアントツールを最適化する。
- SQL データサービス最適化レベルを設定する。
- SQL データサービスのデータ統合処理の並列処理とメモリのプロパティを設定する。
- SQL データサービスのデータオブジェクトのキャッシュを設定する。
- SQL データサービスの結果セットのキャッシュ処理を設定する
- SQL データサービスの仮想テーブルに制約を設定する。

関連項目：

- [「データオブジェクトのキャッシュ」 \(ページ 65\)](#)

サードパーティのクライアントツールの最適化

サードパーティのクライアントツールは、SQL データサービスに対する SQL クエリを処理および実行する際のパフォーマンスに影響します。エンドユーザーが SQL データサービスに対する SQL クエリの実行に使用できるサードパーティのクライアントツールを最適化します。

サードパーティのクライアントツールに関するボトルネックを減らすには、以下の方法を検討します。

大量のクエリ結果はディスク上のファイルに送信する。

サードパーティのクライアントツールで大量のクエリ結果をコンソールウィンドウに表示すると、パフォーマンスに影響することがあります。

サードパーティのクライアントツールでは、暗号化を無効に設定する。

サードパーティのクライアントツールでクエリ結果を取得または表示するときにデータを暗号化すると、パフォーマンスに影響することがあります。

サードパーティのクライアントツールでは、行のセットを事前に取得するように設定する。

サードパーティのクライアントツールで一度に 1 行ずつ行を取得すると、パフォーマンスに影響することがあります。

サードパーティのクライアントツールでは、最初のロード時にテーブルから内容を読み取るオプションを無効に設定する。

サードパーティのクライアントツールの BLOB データ型および CLOB データ型の設定で、BLOB データ型および CLOB データ型がクエリで使用されない場合に最初のロード時にテーブルから内容を読み取るように設定すると、パフォーマンスに影響することがあります。

サードパーティのクライアントツールでは、日付、時刻、およびタイムスタンプの形式や変換にデフォルトの設定を使用するように設定する。

サードパーティのクライアントツールの日付、時刻、およびタイムスタンプの形式や変換の設定で、デフォルトの形式ではなくユーザー指定の形式に設定すると、パフォーマンスに影響することがあります。

デバッグオプションは無効にするかデバッグなしに設定する。

サードパーティのクライアントツールでクエリの実行を追跡するようにデバッグオプションを設定すると、パフォーマンスに影響することがあります。この設定により、クエリの処理中にデバッグファイルに書き込まれるログメッセージが多くなると、それに応じてパフォーマンスが低下します。

SQL データサービス最適化レベル

データ統合サービスは、SQL データサービスを設定した最適化レベルに基づいて最適化します。SQL データサービスでノーマル以外の最適化レベルを使用する場合は、最適化レベルを設定します。デフォルトでは、各 SQL データサービスでノーマルの最適化レベルが使用されます。

最適化レベルが SQL データサービスに対して最適化されたクエリを作成する方法を理解するには、SQL データサービスのクエリプランを表示します。クエリプランを表示すると、Developer ツールにより、最適化レベルに基づいて最適化されたクエリのグラフィカル表現と、元のクエリのグラフィカル表現が表示されます。

以下の最適化レベルを設定できます。

0 (なし)

データ統合サービスは最適化を適用しません。

1 (最小)

データ統合サービスは初期プロジェクション最適化方式を適用します。

2 (ノーマル)

データ統合サービスは、初期プロジェクション、初期選択、ブランチ刈り込み、プッシュイン、グローバル述部、述部の最適化方式を適用します。

3 (完全)

データ統合サービスは、コストベース、初期プロジェクション、初期選択、ブランチ刈り込み、述部、プッシュイン、準結合、データシップ結合の最適化方式を適用します。

デフォルトは2 (ノーマル) です。

以下の方法のうちの1つ以上を使用してSQLデータサービスの最適化レベルを設定できます。

- SQLデータサービスのデータプレビューの最適化レベルを設定する。
- デプロイされているSQLデータサービスの最適化レベルを設定する。
- デプロイされたSQLデータサービスに対して実行するクエリの接続文字列で最適化レベルを設定する。

データプレビューのSQLデータサービス最適化レベルの設定

SQLデータサービスの出力をプレビューするときにData Integration ServiceがSQLクエリの実行に使用する最適化レベルを設定します。

1. Developer ツールで、**[実行]** > **[実行ダイアログを開く]** をクリックします。

[実行] ダイアログボックスが表示されます。

2. **[データビューアの設定]** をクリックします。
3. **[新規]** ボタンをクリックします。
4. データビューアの設定の名前を入力します。
5. **[詳細]** タブをクリックします。
6. 最適化レベルを選択します。
7. **[適用]** をクリックします。
8. **[閉じる]** をクリックします。

データビューアの設定が作成されます。

デプロイされているSQLデータサービスの最適化レベルの設定

デプロイされているSQLデータサービスに対してData Integration ServiceがSQLクエリの実行に使用するData Integration Serviceの最適化レベルを設定します。SQLデータサービス接続で最適化レベルを設定することにより、単一のクエリに対して最適化レベルをオーバーライドすることもできます。

1. 管理者ツールで、Data Integration Serviceを選択します。
2. **[アプリケーション]** ビューをクリックします。
3. 最適化レベルを設定するSQLデータサービスを含むアプリケーションを展開します。

4. SQL データサービスを選択し、次のプロパティを編集します。

プロパティ	説明
最適化レベル	<p>Data Integration Service がオブジェクトに適用する最適化レベルです。設定する最適化レベルに関連する数値を入力します。以下のいずれかの数値を入力できます。</p> <ul style="list-style-type: none"> - 0. データ統合サービスは最適化を適用しません。 - 1. データ統合サービスは初期プロジェクション最適化方式を適用します。 - 2. データ統合サービスは、初期プロジェクション、初期選択、プッシュイン、および述部の各最適化方式を適用します。 - 3. データ統合サービスは、コストベース、初期プロジェクション、初期選択、プッシュイン、述部、準結合の各最適化方式を適用します。

5. Data Integration Services がクエリの実行に使用する最適化レベルをオーバーライドするには、を JDBC URL または ODBC 接続文字列にエントリ `SQLDataServiceOptions.optimizeLevel= <numeric_optimizer_level>` 追加します。

SQL データサービスのクエリプラン

SQL データサービスのクエリプランを表示すると、元のクエリと最適化されたクエリのグラフィカル表現が表示されます。元の表現は Data Integration Service がクエリを処理する方法を説明しています。トランスフォーメーションと、Data Integration Service が各トランスフォーメーションを処理する順番が含まれます。

Developer ツールでは、最適化クエリを生成する Developer ツールに設定された最適化レベルが使用されます。最適化されたクエリは Data Integration Service が実行するとおりにクエリを表示します。

例えば、SQL データサービスの CUSTOMERS 仮想テーブルにクエリを実行する場合、**【データビューア】** ビューで、デフォルトのデータビューアの設定を選択します。この設定により、クエリの最適化レベルはノーマルに設定されます。

以下のクエリを **【データビューア】** ビューに入力します。

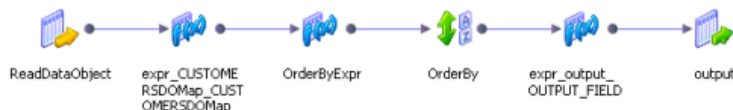
```
select * from CUSTOMERS where CUSTOMER_ID > 150000 order by LAST_NAME
```

クエリの SQL クエリプランを表示すると、クエリのグラフィカル表現が以下のように表示されます。



最適化されていないビューでは、入力したとおりにクエリが表示されます。WHERE 句はフィルタトランスフォーメーションとして、また ORDER BY 句はソータートランスフォーメーションとして表示されます。Developer ツールはパススルー式トランスフォーメーションを使用してポートの名前を変更します。

最適化されたクエリを表示すると、クエリのグラフィカル表現が以下のように表示されます。



最適化されたビューには、Data Integration Service が実行するクエリが表示されます。最適化レベルは [ノーマル] のため、Data Integration Service はフィルタ条件をソースデータオブジェクトにプッシュします。フィルタ条件をプッシュすると、Data Integration Service がソースデータオブジェクトから読み取る行数が減るため、クエリパフォーマンスが向上します。最適化されていないクエリと同様に、ORDER BY 句はソータートランスフォーメーションとして表示されます。パススルー式トランスフォーメーションを使用して、論理トランスフォーメーションで指定したデータ型を適用します。

SQL クエリプランの表示

SQL クエリプランを表示して、仮想テーブルデータをプレビューするときに入力した SQL クエリをマッピングのように表示します。

1. 少なくとも 1 つの仮想テーブルを含む SQL データサービスを開きます。
2. **【データビューア】** ビューをクリックします。
3. **【入力】** ウィンドウで SQL クエリを入力します。
4. 必要に応じて、クエリに適用する最適化レベルを含むデータビューア構成を選択します。
5. **【クエリプランの表示】** をクリックします。
【未最適化】 タブで入力したクエリの SQL クエリプランが表示されます。
6. 最適化されたクエリを表示するには、**【最適化済み】** タブをクリックします。
最適化された SQL クエリプランが表示されます。

SQL データサービスのメモリおよび同時要求のプロパティ

SQL データサービスのパフォーマンスを最適化するには、Administrator ツールでデータ統合サービスの並列処理とメモリのプロパティを設定します。

次の表に、SQL サービスモジュールの要求ごとの最大メモリプロパティを示します。

プロパティ	説明
要求ごとの最大メモリ	<p>要求ごとの最大メモリは、データ統合サービスの次の設定によって異なります。</p> <ul style="list-style-type: none">- 個別のローカルまたはリモートプロセスでジョブが実行されるか、またはサービスプロパティの最大メモリサイズが 0 の場合（デフォルト）。 <p>この場合、要求ごとの最大メモリは、データ統合サービスが、1 回の要求内の、自動キャッシュモードを使用するすべてのトランスフォーメーションに割り当てることができる、最大メモリサイズ（バイト）です。データ統合サービスは、特定のキャッシュサイズの複数のトランスフォーメーションにメモリを別々に割り当てます。要求によって使用されるメモリ n 合計は、要求ごとの最大メモリの値を超えることができます。</p> <ul style="list-style-type: none">- データ統合サービスプロセスでジョブが実行され、かつ、サービスプロパティの最大メモリサイズが 0 を超えている場合。 <p>この場合、要求ごとの最大メモリは、データ統合サービスが単一の要求に割り当てることができる、最大メモリサイズ（バイト）です。要求によって使用されるメモリ合計は、要求ごとの最大メモリの値を超えることはできません。</p> <p>デフォルトは 50,000,000 です。</p>

以下の表に、データ統合サービスプロセスの最大ヒープサイズのプロパティを示します。

プロパティ	説明
最大ヒープサイズ	<p>データ統合サービスを実行する Java Virtual Machine (JVM) に割り当てられる RAM サイズ。このプロパティを使用して、パフォーマンスの向上を図ることができます。単位を指定するには、次のいずれかの文字を値に付加します。</p> <ul style="list-style-type: none"> - b はバイト。 - k はキロバイト。 - m はメガバイト。 - g はギガバイト。 <p>デフォルトは 640 メガバイトです。</p> <p>注: データ統合サービスで大量のデータを処理する必要がある場合は、最大ヒープサイズを増やすことを検討します。</p>

以下の表に、データ統合サービスプロセスの SQL プロパティを示します。

プロパティ	説明
最大同時接続数	<p>データ統合サービスが SQL データサービスに対して確立できるデータベース接続の最大数です。デフォルトは 100 です。</p>

以下の表に、データ統合サービスの実行オプションを示します。

プロパティ	説明
オンデマンド実行プールの最大サイズ	<p>同時に実行できるオンデマンドジョブの最大数。ジョブには、データのプレビュー、プロファイリングジョブ、REST および SQL クエリ、Web サービス要求、および Developer ツールから実行されるマッピングが含まれます。データ統合サービスが受信するすべてのジョブは、オンデマンドプールサイズに関係します。データ統合サービスは、十分なりソースが利用可能であれば、即時にオンデマンドジョブを実行します。それ以外の場合、データ統合サービスはジョブを拒否します。デフォルトは 10 です。</p>
ネイティブバッチ実行プールの最大サイズ	<p>ネイティブ環境で同時に実行できるデプロイ済みジョブの最大数。データ統合サービスは、十分なりソースが利用可能な場合に、ネイティブのマッピングジョブをキューからネイティブのジョブプールに移動します。デフォルトは 10 です。</p>
Hadoop バッチ実行プールの最大サイズ	<p>Hadoop 環境で同時に実行できるデプロイ済みジョブの最大数。データ統合サービスは、十分なりソースが利用可能な場合に、Hadoop ジョブをキューから Hadoop ジョブプールに移動します。デフォルトは 100 です。</p>
最大メモリサイズ	<p>データ統合サービスプロセスでジョブを実行する場合、すべての要求を同時に実行するためにデータ統合サービスが割り当てることができる最大メモリサイズ (バイト)。データ統合サービスがジョブを個別のローカルまたはリモートプロセスで実行する場合、この値は無視されます。データ統合サービスで割り当てることができるメモリの量を制限しない場合は、このプロパティを 0 に設定します。</p> <p>値が 0 を超えていた場合、データ統合サービスは、このプロパティを使用して、すべての要求を同時に実行できる最大メモリサイズを計算します。データ統合サービスでは、次の式に基づいて最大合計メモリが計算されます。</p> <p>最大メモリサイズ + 最大ヒープサイズ + プログラムのコンポーネントのロードに必要なメモリ</p> <p>デフォルトは 0 です。</p> <p>注: プロファイルやデータ品質のマッピングを実行する場合は、このプロパティを 0 に設定します。</p>

SQL データサービスの結果セットキャッシュ

結果セットキャッシュを設定すると、各 SQL データサービスクエリおよび Web サービス要求に関連付けられている DTM プロセスの結果がデータ統合サービスによってキャッシュされます。また、設定した有効期間の結果がキャッシュされます。キャッシュが期限切れになる前にクライアントが同じクエリを行うと、データ統合サービスによってキャッシュされた結果が返されます。

結果セットキャッシュに関するボトルネックを減らすには、以下の方法を検討します。

SQL データサービスの結果セットキャッシュを設定する。

結果セットキャッシュによって、データ統合サービスで SQL データサービスクエリのキャッシュされた結果を使用できるようになります。短期間で同じクエリを複数のユーザーが実行する場合、結果セットキャッシュを使用して同じクエリのランタイムを減らすことができます。

キャッシュされた結果をデータ統合サービスで使用するようにすると、データサービスのパフォーマンスが向上します。データサービスによる同じクエリの処理時間をさらに短縮するには、キャッシュをメモリに格納できるように十分な容量を割り当てます。キャッシュメモリの量を結果をキャッシュするのに必要な量以上に設定すると、システム I/O のオーバーヘッドが減り、パフォーマンスが向上します。キャッシュファイルがディスクに書き込まれると、システム I/O のオーバーヘッドによりデータサービスの処理時間が長くなります。

SQL データサービスの結果セットキャッシュのプロパティ

パフォーマンスが向上するように、Data Integration Service の結果セットキャッシュのプロパティを設定することができます。SQL データサービスで結果セットキャッシュを使用できる時間（ミリ秒）を設定することもできます。

以下の表に、Data Integration Service の結果セットキャッシュのプロパティを示します。

プロパティ	説明
ファイル名のプレフィックス	ディスクに格納されるすべての結果セットキャッシュファイルの名前のプレフィックス。デフォルトは RSCACHE です。
暗号化を有効にする	結果セットキャッシュファイルを 128 ビットの AES 暗号化を使用して暗号化するかどうかを示します。有効な値は true または false です。デフォルトは true です。

以下の表に、SQL データサービスで結果セットキャッシュを使用できる時間（ミリ秒）を設定するプロパティを示します。

プロパティ	説明
結果セットキャッシュの有効期限	結果セットキャッシュを使用できる時間（ミリ秒）。-1 に設定した場合、キャッシュには期限がありません。0 に設定した場合、結果セットキャッシュは無効になります。有効期限の変更は、既存のキャッシュには適用されません。すべてのキャッシュで同じ有効期限を使用する場合は、有効期限を変更した後に結果セットキャッシュをバージします。デフォルトは 0 です。

SQL データサービスの結果セットキャッシュ処理の有効化

同じ SQL データサービスクエリに対してキャッシュされた結果を使用するには、結果セットキャッシュ処理を使用するように Data Integration Service を設定します。

1. Administrator ツールで、Data Integration Service を選択します。
2. **【プロセス】** ビューをクリックして、結果セットキャッシュのプロパティを設定します。
3. **【アプリケーション】** ビューをクリックし、SQL データサービスををクリックして、結果セットキャッシュの有効期限のプロパティを設定します。

一時テーブルにおける仮想データの維持

一時テーブルは、リレーショナルデータベースにあるテーブルで、中間データや一時データを格納します。複雑なクエリでは通常、結合からの情報などの、大量の中間データを格納する必要があります。一時テーブルを実装すると、ビジネスインテリジェンスツールは、SQL データサービスの代わりに一時テーブルからこのデータを取得できます。結果として、パフォーマンスが向上します。

また、一時テーブルにより 2 つの方法でセキュリティも向上します。まず、アクティブなセッションのユーザーのみがテーブルにアクセスできます。また、テーブルはセッションがアクティブな間維持され、データベースは接続が閉じられるとテーブルを削除します。

一時テーブルの実装

一時テーブルを使用して、サイズが大きい複雑なクエリのパフォーマンスを向上させることができます。リレーショナルデータベースの一時テーブルへのクエリは、同じデータセットに対して SQL データサービスに繰り返しクエリを実行するよりも高速なので、一時テーブルを使用することでパフォーマンスが向上します。

パフォーマンスを向上させるために一時テーブルを実装するには、Informatica 管理者とビジネスインテリジェンスツールの開発者による操作が必要です。

まず、Informatica 管理者がリレーショナルテーブル接続を作成し、その接続を使用するようにデータ統合サービスを設定します。

次に、ビジネスインテリジェントツール（IBM Cognos、SAP Business Objects など）の開発者が、ビジネスインテリジェンスツールと Informatica SQL データサービス間の接続を作成します。接続は、Informatica ODBC または JDBC ドライバを使用します。

この接続がアクティブな場合、ビジネスインテリジェンスツールは一時テーブルを作成して使用し、大量の中間データを処理できます。

第 9 章

Web サービスの最適化

この章では、以下の項目について説明します。

- [Web サービスの最適化の概要, 77 ページ](#)
- [HTTP 要求の最適化, 78 ページ](#)
- [Web サービスメッセージの圧縮, 78 ページ](#)
- [Web サービス最適化レベル, 78 ページ](#)
- [Web サービスのメモリおよび同時要求のプロパティ, 80 ページ](#)
- [アクティブな DTM インスタンスを設定する Web サービスプロパティ, 82 ページ](#)
- [Web サービスの結果セットキャッシュ処理, 83 ページ](#)
- [Web サービスのログ管理, 83 ページ](#)

Web サービスの最適化の概要

Data Integration Service で Web サービス要求を実行する際のパフォーマンスが向上するように、Web サービスを最適化することができます。メモリを管理したり、同時 Web サービス要求を処理したりできるように、Data Integration Service をチューニングします。Web サービスのパフォーマンスを向上させる方法として、Web サービスメッセージの圧縮、HTTP 要求の最適化、データオブジェクトと結果セットキャッシュの設定、エラーログレベルの設定などがあります。

Web サービスを最適化する方法は次のとおりです。

- HTTP 要求を最適化する。
- Web サービスメッセージを圧縮する。
- Web サービス最適化レベルを設定する。
- Web サービスのデータ統合処理の並列処理とメモリのプロパティを設定する。
- DTM プロセスをアクティブな状態にして複数の Web サービス要求を処理できるように Data Integration Service を設定する。
- Web サービスに対してデータオブジェクトのキャッシュを設定する。
- Web サービスに対して結果セットのキャッシュを設定する。
- Web サービスのランタイムエラーログレベルを設定する。

関連項目：

- [「データオブジェクトのキャッシュ」 \(ページ 65\)](#)

HTTP 要求の最適化

Web サーバーへの要求の数が少なくなるように、HTTP 要求を最適化します。

HTTP 要求に関するボトルネックを減らすには、以下の方法を検討します。

Web サービスクライアントの HTTP ソケットタイムアウトを小さくする。

ソケットタイムアウトは、クライアントで HTTP 要求がタイムアウトするまでの時間を示します。ソケットタイムアウトの値が大きいと、Web サービスクライアントがハングする可能性があります。

Web サービスメッセージの圧縮

プロバイダとの間でやり取りされる大きい Web メッセージを圧縮することで、Web サービスのパフォーマンスを最適化することができます。

Web サービスメッセージのボトルネックを減らすには、以下の方法を検討します。

Web サービスクライアントの SOAP メッセージ圧縮を有効にする。

SOAP メッセージ圧縮を使用すると、Web サービスメッセージを圧縮して、圧縮された Web サービスクライアントメッセージを受け取ることができます。Web サービスでは、Web サービスクライアントからの gzip 圧縮形式の SOAP メッセージを受け入れることができます。

Web サービスからの応答を受信すると、Data Integration Service は SOAP メッセージ内の Content-Encoding HTTP ヘッダを調べ、SOAP メッセージをデコードします。

Web サービス最適化レベル

データ統合サービスにより、設定した最適化レベルに基づいて Web サービスが最適化されます。Web サービスでノーマル以外の最適化レベルを使用する場合は、最適化レベルを設定します。デフォルトでは、各 Web サービスでノーマルの最適化レベルが使用されます。

以下の最適化レベルのいずれかを選択できます。

0 (なし)

データ統合サービスは最適化を適用しません。

1 (最小)

データ統合サービスは初期プロジェクション最適化方式を適用します。

2 (ノーマル)

データ統合サービスは、初期プロジェクション、初期選択、ブランチ刈り込み、ブッシュイン、グローバル述部、述部の最適化方式を適用します。

3 (完全)

データ統合サービスは、コストベース、初期プロジェクション、初期選択、ブランチ刈り込み、述部、プッシュイン、準結合、データシップ結合の最適化方式を適用します。

デフォルトは2 (ノーマル) です。

以下の方法のうちの1つ以上を使用して Web サービスの最適化レベルを設定できます。

- データ統合サービスにデプロイする前に、Web サービスのデータプレビューの最適化レベルを設定する。
- 特定のデータ統合サービスで動作するデプロイされた Web サービスの最適化レベルを設定する。
- デプロイされている Web サービスに対して Web サービス要求のヘッダで最適化レベルを設定する。

データプレビューの Web サービス最適化レベルの設定

Data Integration Services が Web サービスの出力のプレビューに使用する最適化レベルを設定します。

1. Developer ツールで、**[実行]** > **【実行ダイアログを開く】** をクリックします。
 【実行】 ダイアログボックスが表示されます。
2. **【Web サービス設定】** をクリックします。
3. **【新規】** ボタンをクリックします。
4. Web サービス設定の名前を入力します。
5. **【詳細】** タブをクリックします。
6. 最適化レベルを選択します。
7. **【適用】** をクリックします。
8. **【閉じる】** をクリックします。

Web サービス設定が作成されます。

データビューアを実行して操作マッピングの出力をプレビューするときは、使用する最適化レベルが含まれている Web サービス設定を選択します。

デプロイされている Web サービスの最適化レベルの設定

Data Integration Service がデプロイされている Web サービスの実行に使用する最適化レベルを設定します。Web サービスの SOAP リクエストの HTTP ヘッダで最適化レベルを設定することにより、単一のリクエストに対して最適化レベルをオーバーライドすることもできます。

1. 管理者ツールで、Data Integration Service を選択します。
2. **【アプリケーション】** ビューをクリックします。
3. 最適化レベルを設定する Web サービスを含むアプリケーションを展開します。

4. Web サービスを選択し、次のプロパティを編集します。

プロパティ	説明
最適化レベル	Data Integration Service がオブジェクトに適用する最適化レベルです。設定する最適化レベルに関連する数値を入力します。以下のいずれかの数値を入力できます。 <ul style="list-style-type: none">- 0. データ統合サービスは最適化を適用しません。- 1. データ統合サービスは初期プロジェクション最適化方式を適用します。- 2. データ統合サービスは、初期プロジェクション、初期選択、プッシュイン、および述部の各最適化方式を適用します。- 3. データ統合サービスは、コストベース、初期プロジェクション、初期選択、プッシュイン、述部、準結合の各最適化方式を適用します。

5. Web サービス要求に対して Web サービス最適化レベルをオーバーライドするには、Web サービス SOAP リクエストの HTTP ヘッダーにエントリ `WebServiceOptions.optimizeLevel= <numeric_optimizer_level>` を追加します。

Web サービスのメモリおよび同時要求のプロパティ

Web サービスのパフォーマンスを最適化するには、Administrator ツールで、データ統合サービスおよび各 Web サービスの並列処理とメモリのプロパティを設定します。

次の表に、Web サービスモジュールに対する要求ごとの最大メモリプロパティを示します。

プロパティ	説明
要求ごとの最大メモリ	<p>要求ごとの最大メモリは、データ統合サービスの次の設定によって異なります。</p> <ul style="list-style-type: none">- 個別のローカルまたはリモートプロセスでジョブが実行されるか、またはサービスプロパティの最大メモリサイズが 0 の場合（デフォルト）。 この場合、要求ごとの最大メモリは、データ統合サービスが、1 回の要求内の、自動キャッシュモードを使用するすべてのトランスフォーメーションに割り当てることができる、最大メモリサイズ（バイト）です。データ統合サービスは、特定のキャッシュサイズの複数のトランスフォーメーションにメモリを別々に割り当てます。要求によって使用されるメモリ n 合計は、要求ごとの最大メモリの値を超えることができます。- データ統合サービスプロセスでジョブが実行され、かつ、サービスプロパティの最大メモリサイズが 0 を超えている場合。 この場合、要求ごとの最大メモリは、データ統合サービスが単一の要求に割り当てることができる、最大メモリサイズ（バイト）です。要求によって使用されるメモリ合計は、要求ごとの最大メモリの値を超えることはできません。 デフォルトは 50,000,000 です。

以下の表に、データ統合サービスの実行オプションを示します。

プロパティ	説明
最大メモリサイズ	<p>データ統合サービスプロセスでジョブを実行する場合、すべての要求を同時に実行するためにデータ統合サービスが割り当てることができる最大メモリサイズ（バイト）。データ統合サービスがジョブを個別のローカルまたはリモートプロセスで実行する場合、この値は無視されます。データ統合サービスで割り当てることができるメモリの量を制限しない場合は、このプロパティを 0 に設定します。</p> <p>値が 0 を超えていた場合、データ統合サービスは、このプロパティを使用して、すべての要求を同時に実行できる最大メモリサイズを計算します。データ統合サービスでは、次の式に基づいて最大合計メモリが計算されます。</p> <p>最大メモリサイズ + 最大ヒープサイズ + プログラムのコンポーネントのロードに必要なメモリ デフォルトは 0 です。</p> <p>注: プロファイルやデータ品質のマッピングを実行する場合は、このプロパティを 0 に設定します。</p>

以下の表に、データ統合サービスプロセスの HTTP 設定プロパティを示します。

プロパティ	説明
最大バックログリクエスト数	現在のデータ統合サービスプロセスに対してキューで待機可能な HTTP 接続または HTTPS 接続の最大数。デフォルトは 100 です。
最大同時要求数	<p>現在のデータ統合サービスプロセスに対して確立できる HTTP 接続または HTTPS 接続の最大数。最小値は 4。デフォルトは 200 です。</p> <p>注: Web サービスのこのプロパティは、データ統合サービスのバックログに送信されずに受け入れられる Web サービス要求の数に影響します。</p>

以下の表に、データ統合サービスプロセスに対して設定できる最大ヒープサイズのプロパティを示します。

プロパティ	説明
最大ヒープサイズ	<p>データ統合サービスを実行する Java Virtual Machine (JVM) に割り当てられる RAM サイズ。このプロパティを使用して、パフォーマンスの向上を図ることができます。単位を指定するには、次のいずれかの文字を値に付加します。</p> <ul style="list-style-type: none"> - b はバイト。 - k はキロバイト。 - m はメガバイト。 - g はギガバイト。 <p>デフォルトは 640 メガバイトです。</p> <p>注: データ統合サービスで大量のデータを処理する必要がある場合は、最大ヒープサイズを増やすことを検討します。</p>

データ統合サービスの同時 Web サービス要求の設定例

データ統合サービスによる同時 Web サービス要求の処理方法を設定するときは、Web サービスとデータ統合サービスプロセスの同時要求の最大数の値が同じであることを確認してください。

例えば、次の設定では、同時 HTTP 要求は 200 個まで受け入れられますが、Web サービスの同時要求は 10 個しか受け入れられません。

プロパティタイプ	プロパティ名	設定
データ統合サービスプロセス	最大同時要求数	200
データ統合サービスプロセス	最大バックログリクエスト数	500
データ統合サービス	オンデマンド実行プールの最大サイズ	100
Web サービス	最大同時要求数	10

データ統合サービスで 20 個の Web サービス要求を受け取った場合、Web サービスでは同時要求が 10 個しか受け入れられないため、10 個の Web サービス要求が失敗します。

Web サービスの同時要求の最大数に達したときに Web サービス要求が失敗しないようにするには、データ統合サービスプロセスと Web サービスに同じ最大値を設定します。データ統合サービスに送信された要求の数が最大同時要求数の値を超えると、データ統合サービスプロセスで要求を処理できるようになるまで、以降の要求がバックログで保持されます。

アクティブな DTM インスタンスを設定する Web サービスプロパティ

パフォーマンスを向上させるために、DTM インスタンスをアクティブな状態にして複数の Web サービス要求を処理できるようにデータ統合サービスを設定できます。データ統合サービスの [DTM キープアライブ時間] プロパティは、Administrator ツールで設定できます。

以下の表に、[DTM キープアライブ時間] プロパティを示します。

プロパティ	説明
DTM キープアライブ時間	<p>DTM インスタンスが最後の要求の完了後にオープン状態を維持する期間（ミリ秒）。同じ操作に対して送信された Web サービス要求は、オープンインスタンスを再利用できます。要求の処理に必要な時間が、DTM インスタンスの初期化時間よりも短い場合は、キープアライブ時間を使用してパフォーマンスを向上させます。要求が失敗すると、DTM インスタンスは終了します。</p> <p>デフォルトは 5000 です。</p> <p>注: 既存の DTM インスタンスを使用することでパフォーマンスが向上します。要求のたびに DIS で DTM インスタンスを開始すると、余分なリソースが必要になるからです。DTM をアクティブな状態にしておく場合はメモリが消費されます。そのため、このオプションを設定するときはメモリ消費量に注意する必要があります。</p>

Web サービスの結果セットキャッシュ処理

結果セットキャッシュ処理を設定すると、各 Web サービス要求に関連付けられている DTM プロセスの結果が、Data Integration Service によってキャッシュされます。また、設定した有効期間の結果がキャッシュされます。キャッシュが期限切れになる前に外部クライアントが同じ要求を行うと、Data Integration Service によってキャッシュされた結果が返されます。

結果セットキャッシュに関するボトルネックを減らすには、以下の方法を検討します。

Web サービスの結果セットキャッシュを設定する。

結果セットキャッシュ処理によって、Data Integration Service で Web サービス要求のキャッシュされた結果を使用できるようになります。短期間で同じクエリを複数のユーザーが実行する場合、結果セットキャッシュ処理を使用して同じクエリのランタイムを減らすことができます。

Web サービスで WS-Security が使用される場合は、Data Integration Service でユーザーごとに Web サービスの結果セットキャッシュが格納されます。Data Integration Service では、Web サービス要求のユーザー名トークンに指定されているユーザー名ごとにキャッシュが格納されます。Data Integration Service でユーザーごとに結果がキャッシュされる場合は、Web サービス要求を送信したユーザーだけにキャッシュされた結果が返されます。

Web サービスの結果セットキャッシュ処理の有効化

同じ Web サービス要求に対してキャッシュされた結果を使用するには、結果セットキャッシュ処理を使用するように Data Integration Service を設定します。

1. Administrator ツールで、Data Integration Service を選択します。
2. **【プロセス】** ビューをクリックして、結果セットキャッシュのプロパティを設定します。
3. **【アプリケーション】** ビューをクリックし、Web サービスをクリックします。次に、操作をクリックして、Web サービス操作のプロパティでキャッシュの有効期間を設定します。Data Integration Service でユーザーごとに結果をキャッシュするには、Web サービスのプロパティで WS-Security を有効にします。
4. Web サービス操作が結果セットをキャッシュするように設定されている場合に、Web サービス要求に対して結果セットキャッシュ処理を無効にするには、SOAP リクエストの HTTP ヘッダに以下の構文を追加します。

```
WebServiceOptions.disableResultSetCache=true
```

Web サービスのログ管理

Data Integration Service で書き込みを行う管理対象のログファイルの数が多いと、システム I/O のパフォーマンスが低下することがあります。Data Integration Service では、設定したトレースレベルに基づいて Web サービスのランタイムログが生成されます。Data Integration Service で書き込みを行う管理対象のログファイルの数に注意するようにしてください。

Web サービスのログに関するボトルネックを減らすには、以下の方法を検討します。

Web サービスのトレースレベルをオフに設定する。

デプロイ済みの Web サービスのプロパティを設定するときに、ログのトレースレベルを指定することができます。トレースレベルによって、Data Integration Service でランタイムログの場所に書き込まれるログのタイプが決まります。Web サービスのデフォルトのトレースレベルは INFO です。トレースレベルを FINEST または ALL に設定すると、ログファイルに書き込まれるログが多くなり、パフォーマンスが

低下する可能性があります。トレースレベルを FINEST または ALL に設定した場合のパフォーマンスに対する影響は、Web サービスで HTTPS および WS-Security を使用する場合に最も大きくなります。

不要になったログファイルをアーカイブする。

保存するログファイルの数が多すぎると、システム I/O に影響します。デフォルトでは、Web サービスのランタイムログは次のディレクトリに書き込まれます。 <InformaticaInstallationDir>/tomcat/bin/dislogs/ws

注: ログを空にするために ws フォルダを削除する場合は、ws フォルダを作成し直す必要があります。ws フォルダを削除して作成し直すときは、Data Integration Service を停止してから行ってください。

データ統合サービスの [ログのスキップ] プロパティを有効にする

Web サービスの要求が正常に完了したときに、データ統合サービスでログファイルが生成されないようにするには、**[ログのスキップ]** プロパティを有効にします。Web サービスのトレースレベルは、データ統合サービスで INFO またはそれ以上に設定する必要があります。

第 10 章

接続の最適化

この章では、以下の項目について説明します。

- [接続の最適化の概要, 85 ページ](#)
- [接続プール, 85 ページ](#)
- [データベースのネットワークパケットサイズ, 86 ページ](#)

接続の最適化の概要

パフォーマンスが向上するように接続を最適化することができます。データベース接続のアイドル状態の接続インスタンスをプールで管理することが可能です。ネットワークパケットのサイズを拡張することで、大きなデータパケットが一度にネットワークを流れるようにすることができます。

接続を最適化する方法は次のとおりです。

- 接続プールを最適化する。
- データベースのネットワークパケットのサイズを最適化する。

接続プール

接続プールは、データ統合サービスによって使用されるデータベース接続情報をキャッシュするためのフレームワークです。キャッシュされた接続情報を再利用することによってパフォーマンスを向上させます。

接続のボトルネックには次の解決策を検討してください。

データベース接続の接続プールを有効にする。

接続プールを有効にして接続パフォーマンスを最適化します。データベース接続のアイドル状態の接続インスタンスを管理できます。接続プールは設定したプールプロパティに基づいてアイドル状態の接続インスタンスを保持します。アイドル状態の接続の最大数、最少数、最大待機時間を調整できます。

接続オブジェクトのプールのプロパティ

データベース接続の【プール】ビューで接続プールのプロパティを編集できます。

接続プールライブラリの数は、実行中のデータ統合サービスプロセスまたは DTM プロセスの数によって異なります。各データ統合サービスプロセスまたは DTM プロセスは、独自の接続プールライブラリを維持します。プールプロパティの値は接続プールライブラリごとです。

例えば、最大接続数を 15 に設定すると、各接続ライブラリはプール内に最大 15 の接続プールライブラリを持つことができます。データ統合サービスがジョブを別々のローカルプロセスで実行する場合、3 つの DTM プロセスが実行されると、最大 45 のアイドル接続インスタンスを持つことができます。

アイドル接続インスタンス数を減らすには、接続最小数を 0 に設定して、各データベース接続の最大アイドル時間を減らします。

次の表に、データベース接続の【プール】ビューで編集できるデータベース接続プールのプロパティを示します。

接続プールを有効にする

接続プールを有効にします。接続プールを有効にした場合、各接続プールはメモリ内にアイドル状態の接続インスタンスを保持します。アイドル状態の接続プールを削除するには、データ統合サービスを再起動する必要があります。

接続プールが無効になっている場合、DTM プロセスまたはデータ統合サービスプロセスはすべてのプールアクティビティを停止します。DTM プロセスまたはデータ統合サービスプロセスはジョブを処理するたびに接続インスタンスを作成します。ジョブの処理を終了するときにインスタンスを削除します。

デフォルトは i5/OS の DB2、z/OS の DB2、IBM DB2、Microsoft SQL Server、Oracle、および ODBC 接続で有効です。デフォルトは Adabas、IMS、シーケンシャル、および VSAM 接続で無効です。

最小接続数

最大アイドル時間に達した後、プールがデータベース接続で維持するアイドル接続インスタンスの最小数。この値をアイドル接続インスタンスの最大数以下に設定します。デフォルトは 0 です。

最大接続数

最大アイドル時間に達する前にプールがデータベース接続で維持するアイドル接続インスタンスの最大数。アイドル状態の接続インスタンスの最小数より大きな値を設定します。デフォルトは 15 です。

最大アイドル時間

接続プールでアイドル状態を保持できる接続インスタンスの最小数を超えた接続インスタンスが削除されるまでの秒数。接続インスタンスがアイドル状態の接続インスタンスの最小数を超えない場合、接続プールはアイドル時間を無視します。デフォルトは 120 です。

データベースのネットワークパケットサイズ

Oracle、Sybase ASE、または Microsoft SQL Server のデータの読み取りや書き込みを行う場合は、それらの対象のデータベースに基づいてネットワークパケットのサイズを拡張することで、パフォーマンスを向上させることができます。ネットワークパケットのサイズを拡張すると、大きなデータパケットが一度にネットワークを流れるようにすることができます。

データベースのネットワークパケットのサイズに関するボトルネックを減らすには、以下の方法を検討します。
Oracle データベースのネットワークパケットのサイズを拡張する。

データベースサーバーのネットワークパケットのサイズは、listener.ora と tnsnames.ora で拡張できます。必要に応じて、パケットサイズの拡張の詳細について、データベースのマニュアルを参照してください。

Sybase ASE データベースのネットワークパケットのサイズを拡張する。

パケットサイズの拡張方法の詳細については、データベースのマニュアルを参照してください。Sybase ASE については、Data Integration Service でリレーショナル接続オブジェクトのパケットサイズも変更して、データベースサーバーのパケットサイズに対応させる必要があります。

Microsoft SQL Server データベースのネットワークパケットのサイズを拡張する。

パケットサイズの拡張方法の詳細については、データベースのマニュアルを参照してください。Microsoft SQL Server については、Data Integration Service でリレーショナル接続オブジェクトのパケットサイズも変更して、データベースサーバーのパケットサイズに対応させる必要があります。

索引

記号

Windows

ボトルネック [11](#)

D

Data Integration Service

Web サービスの結果セットキャッシュ [83](#)

H

HTTP 要求の最適化

Web サービスの最適化 [78](#)

J

Java トランスフォーメーション

トランスフォーメーションの最適化 [27](#)

JDBC ドライバ

実行時間の最適化 [70](#)

O

Oracle データベースの最適化

ソースの最適化 [23](#)

ターゲットの最適化 [15](#)

S

Single-Pass 読み込み

マッピングの最適化 [51](#)

SQL クエリプラン

表示 [73](#)

Web サービスの結果セットキャッシュ

Data Integration Service [83](#)

SQL データサービス

メモリ割り当て [73](#)

SQL データサービスの結果セットキャッシュ

データ統合サービス [75](#)

SQL データサービスの結果セットキャッシュの有効化

結果セットキャッシュ [76](#)

SQL データサービスの最適化

JDBC ドライバ [70](#)

サードパーティクライアントツール [70](#)

SQL トランスフォーメーション

初期選択の最適化 [34](#)

トランスフォーメーションの最適化 [33](#)

プッシュイン最適化 [34](#)

最適化にプッシュインのプロパティ [35](#)

SQL ヒント

Developer ツールでの入力 [20](#)

U

UNIX

システムのボトルネック [12](#)

W

Web サービスコンシューマトランスフォーメーション

トランスフォーメーションの最適化 [37](#)

フィルタの最適化 [38](#)

プッシュイン最適化 [38](#)

最適化にプッシュインの有効化 [39](#)

初期選択の最適化 [38](#)

Web サービス

メモリ割り当て [82](#)

同時要求 [80](#)

Web サービスの最適化

HTTP 要求の最適化 [78](#)

Web サービスメッセージの圧縮 [78](#)

Web サービスのログ管理

エラートレースレベル [83](#)

Web サービスメッセージの圧縮

Web サービスの最適化 [78](#)

あ

アクティブ DTM インスタンス

Web サービス [82](#)

アグリゲータトランスフォーメーション

トランスフォーメーションの最適化 [25](#)

アナリストサービスの最適化

実行時間の最適化 [60](#)

い

一時テーブル

説明 [76](#)

え

エラートレース

マッピングの最適化 [52](#)

エラートレースレベル

Web サービスのログ管理 [83](#)

か

カスタマイズデータオブジェクト
ソースの最適化 [22](#)
監視統計
実行時間の最適化 [62](#)

く

クエリの最適化
ソースの最適化 [18](#)
クエリビュー
ヒントの設定 [20](#)

け

結果セットキャッシュ
SQL データサービスの結果セットキャッシュの有効化 [76](#)
結果セットキャッシュのプロパティ [75](#)
結果セットキャッシュのプロパティ
実行時間の最適化 [75](#)

こ

コストベース最適化
説明 [45](#)
個別に選択
ソースの最適化 [19](#)

さ

サードパーティクライアントツール
実行時間の最適化 [70](#)
最大並行処理
増加 [55](#)
最適化
コストベースの最適化方式 [45](#)
準結合最適化方式 [46](#)
初期選択の最適化方法 [47](#)
初期プロジェクション最適化方法 [43](#)
データシップ結合最適化方式 [45](#)
副次作用 [36](#)
プッシュイン最適化方式 [48](#)
プッシュダウンの最適化方式 [48](#)
ランチャリ込み最適化方式 [48](#)
マッピングのパフォーマンスの方式 [42](#)

し

システム
UNIX のボトルネック、特定 [12](#)
Windows のボトルネック、特定 [11](#)
システムの最適化
実行時間の最適化 [68](#)
実行時間の最適化
アナリストサービスの最適化 [60](#)
監視統計 [62](#)
システムの最適化 [68](#)
データ統合サービスの最適化 [61](#)
モデルリポジトリサービスの最適化 [62](#)
準結合最適化
説明 [46](#)

ジョイナトランスフォーメーション
トランスフォーメーションの最適化 [30](#)
条件フィルタ
ソースの最適化 [19](#)
初期選択の最適化
SQL トランスフォーメーション [34](#)
Web サービスコンシューマトランスフォーメーション [38](#)
説明 [47](#)
初期プロジェクション最適化
説明 [43](#)

せ

制約
制約の設定 [21](#)
ソースの最適化 [21](#)
接続の最適化
接続プール [85](#)
データベースのネットワークパケットサイズ [86](#)
接続プール
接続の最適化 [85](#)
プロパティ [85](#)

そ

ソースの最適化
Oracle データベースの最適化 [23](#)
カスタマイズデータオブジェクト [22](#)
クエリの最適化 [18](#)
個別に選択 [19](#)
条件フィルタ [19](#)
制約 [21](#)
フラットファイルソース [18](#)
ソータトランスフォーメーション
トランスフォーメーションの最適化 [33](#)

た

ターゲットの最適化
Oracle データベースの最適化 [15](#)
データベースのチェックポイント間隔 [15](#)
バルクロード [15](#)
フラットファイルターゲット [14](#)

て

データオブジェクトキャッシュ
インデックスキャッシュ [65](#)
最適化 [67](#)
設定 [65](#)
説明 [65](#)
テーブルデータ型 [66](#)
ユーザー管理テーブル [65](#)
データ型変換の最適化
マッピングの最適化 [52](#)
データシップ結合最適化
説明 [45](#)
データ統合サービス
SQL データサービスの結果セットキャッシュ [75](#)
データ統合サービスの最適化
実行時間の最適化 [61](#)
データベースのチェックポイント間隔
ターゲットの最適化 [15](#)

データベースのネットワークパケットサイズ

接続の最適化 [86](#)

データベースのヒント

Developer ツールでの入力 [20](#)

データベース

パーティション化に対応するためのソースの最適化 [57](#)

パーティション化に対応するためのターゲットの最適化 [57](#)

と

トランスフォーメーション

パーティション化に対応するための最適化 [58](#)

トランスフォーメーションエラーの除去

トランスフォーメーションの最適化 [36](#)

トランスフォーメーションのキャッシュ

トランスフォーメーションの最適化 [35](#)

トランスフォーメーションの最適化

Java トランスフォーメーション [27](#)

SQL トランスフォーメーション [33](#)

Web サービスコンシューマトランスフォーメーション [37](#)

アグリゲータトランスフォーメーション [25](#)

ジョイナトランスフォーメーション [30](#)

ソータトランスフォーメーション [33](#)

トランスフォーメーションエラーの除去 [36](#)

トランスフォーメーションのキャッシュ [35](#)

ルックアップトランスフォーメーション [30](#)

の

ノーマル最適化レベル

説明 [42](#)

は

パーティション化

最適化 [54](#)

ソースデータベースの最適化 [57](#)

ターゲットデータベースの最適化 [57](#)

トランスフォーメーションの最適化 [58](#)

複数の CPU [55](#)

フラットファイルソースの最適化 [56](#)

フラットファイルターゲットの最適化 [56](#)

パフォーマンスのチューニング

グローバル述部最適化方式 [47](#)

コストベースの最適化方式 [45](#)

最適化方式 [42](#)

述部最適化方式 [44](#)

準結合最適化方式 [46](#)

初期選択の最適化方式 [47](#)

初期プロジェクション最適化方法 [43](#)

データシップ結合最適化方式 [45](#)

プッシュイン最適化方式 [48](#)

プッシュダウンの最適化方式 [48](#)

ブランチ刈り込み最適化方式 [48](#)

プロセス概要 [10](#)

最適化レベル [42](#)

バルクロード

ターゲットの最適化 [15](#)

ひ

ヒント

クエリビュー [20](#)

ふ

フィルタの最適化

マッピングの最適化 [51](#)

フィルタポート

Web サービスコンシューマトランスフォーメーション [38](#)

副次作用

SQL トランスフォーメーション [34](#)

Web サービスコンシューマトランスフォーメーション [38](#)

説明 [36](#)

副次作用あり

トランスフォーメーションのプロパティの説明 [36](#)

プッシュイン最適化

SQL トランスフォーメーション [34](#)

SQL トランスフォーメーションでの有効化 [35](#)

Web サービスコンシューマトランスフォーメーション [38](#)

説明 [48](#)

プッシュダウンの最適化

説明 [48](#)

プッシュダウンの最適化方式

ソースプッシュダウン [50](#)

完全なプッシュダウン [49](#)

フラットファイル

パーティション化に対応するためのソースの最適化 [56](#)

パーティション化に対応するためのターゲットの最適化 [56](#)

フラットファイルソース

ソースの最適化 [18](#)

フラットファイルターゲット

ターゲットの最適化 [14](#)

ブランチ刈り込み最適化

説明 [48](#)

ほ

ボトルネック

UNIX [12](#)

Windows [11](#)

ま

マッピング

グローバル述部最適化方式 [47](#)

最適化方式 [42](#)

述部最適化方式 [44](#)

パーティション化の最適化 [54](#)

マッピングの最適化

Single-Pass 読み込み [51](#)

エラートレース [52](#)

データ型変換の最適化 [52](#)

フィルタの最適化 [51](#)

式の最適化 [25](#)

め

メモリ割り当て

SQL データサービス [73](#)

Web サービス [82](#)

アクティブ DTM インスタンス [82](#)

同時要求 [73](#)

も

モデルリポジトリサービスの最適化

実行時間の最適化 [62](#)

る

ルックアップトランスフォーメーション
トランスフォーメーションの最適化 [30](#)

ろ

論理データオブジェクト
データベースへのキャッシュ [65](#)